铁谱图像定量分析系统

**部分源代码（60页）**

[**中国矿业大学**](http://job.guolairen.com/archive/09/02/72546098.html)

**2017年2月**

# Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

using HalconDotNet;

namespace Halcon\_region\_sort

{

static class Program

{

/// <summary>

/// 应用程序的主入口点。

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

HALCON\_SORT begin = new HALCON\_SORT();

Application.Run(begin);

}

}

}

# HALCON\_SORT.Desigener.cs

using HalconDotNet;

namespace Halcon\_region\_sort

{

partial class HALCON\_SORT

{

/// <summary>

/// 必需的设计器变量。

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// 清理所有正在使用的资源。

/// </summary>

/// <param name="disposing">如果应释放托管资源，为 true；否则为 false。</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows 窗体设计器生成的代码

/// <summary>

/// 设计器支持所需的方法 - 不要

/// 使用代码编辑器修改此方法的内容。

/// </summary>

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(HALCON\_SORT));

this.hWindow\_main = new HalconDotNet.HWindowControl();

this.menuStrip1 = new System.Windows.Forms.MenuStrip();

this.文件ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.读取图像ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.保存图像ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.退出ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.连接相机ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.自动处理ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.打开自动处理窗口ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.帮助ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.关于ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.textBox1 = new System.Windows.Forms.TextBox();

this.groupBox1 = new System.Windows.Forms.GroupBox();

this.label11 = new System.Windows.Forms.Label();

this.label10 = new System.Windows.Forms.Label();

this.trackBar4 = new System.Windows.Forms.TrackBar();

this.trackBar3 = new System.Windows.Forms.TrackBar();

this.comboBox4 = new System.Windows.Forms.ComboBox();

this.comboBox3 = new System.Windows.Forms.ComboBox();

this.label1 = new System.Windows.Forms.Label();

this.trackBar2 = new System.Windows.Forms.TrackBar();

this.button6 = new System.Windows.Forms.Button();

this.button5 = new System.Windows.Forms.Button();

this.comboBox2 = new System.Windows.Forms.ComboBox();

this.button4 = new System.Windows.Forms.Button();

this.button3 = new System.Windows.Forms.Button();

this.labelthreshold = new System.Windows.Forms.Label();

this.trackBar1 = new System.Windows.Forms.TrackBar();

this.button2 = new System.Windows.Forms.Button();

this.comboBox1 = new System.Windows.Forms.ComboBox();

this.button1 = new System.Windows.Forms.Button();

this.groupBox2 = new System.Windows.Forms.GroupBox();

this.label15 = new System.Windows.Forms.Label();

this.label14 = new System.Windows.Forms.Label();

this.label13 = new System.Windows.Forms.Label();

this.label12 = new System.Windows.Forms.Label();

this.textBox8 = new System.Windows.Forms.TextBox();

this.label9 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.textBox5 = new System.Windows.Forms.TextBox();

this.label7 = new System.Windows.Forms.Label();

this.label8 = new System.Windows.Forms.Label();

this.textBox6 = new System.Windows.Forms.TextBox();

this.textBox7 = new System.Windows.Forms.TextBox();

this.label6 = new System.Windows.Forms.Label();

this.label5 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.textBox4 = new System.Windows.Forms.TextBox();

this.textBox3 = new System.Windows.Forms.TextBox();

this.button9 = new System.Windows.Forms.Button();

this.textBox2 = new System.Windows.Forms.TextBox();

this.button8 = new System.Windows.Forms.Button();

this.button7 = new System.Windows.Forms.Button();

this.button10 = new System.Windows.Forms.Button();

this.button11 = new System.Windows.Forms.Button();

this.saveImage = new System.Windows.Forms.SaveFileDialog();

this.button12 = new System.Windows.Forms.Button();

this.menuStrip1.SuspendLayout();

this.groupBox1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar4)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar3)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar2)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).BeginInit();

this.groupBox2.SuspendLayout();

this.SuspendLayout();

//

// hWindow\_main

//

this.hWindow\_main.BackColor = System.Drawing.Color.Black;

this.hWindow\_main.BorderColor = System.Drawing.Color.Black;

this.hWindow\_main.ImagePart = new System.Drawing.Rectangle(-15, -15, 2590, 1960);

this.hWindow\_main.Location = new System.Drawing.Point(379, 0);

this.hWindow\_main.Name = "hWindow\_main";

this.hWindow\_main.Size = new System.Drawing.Size(825, 576);

this.hWindow\_main.TabIndex = 0;

this.hWindow\_main.WindowSize = new System.Drawing.Size(825, 576);

this.hWindow\_main.HMouseMove += new HalconDotNet.HMouseEventHandler(this.hWindowControl1\_HMouseMove);

//

// menuStrip1

//

this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.文件ToolStripMenuItem,

this.自动处理ToolStripMenuItem,

this.帮助ToolStripMenuItem,

this.关于ToolStripMenuItem});

this.menuStrip1.Location = new System.Drawing.Point(0, 0);

this.menuStrip1.Name = "menuStrip1";

this.menuStrip1.Size = new System.Drawing.Size(1210, 25);

this.menuStrip1.TabIndex = 1;

this.menuStrip1.Text = "menuStrip1";

//

// 文件ToolStripMenuItem

//

this.文件ToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.读取图像ToolStripMenuItem,

this.保存图像ToolStripMenuItem,

this.退出ToolStripMenuItem,

this.连接相机ToolStripMenuItem});

this.文件ToolStripMenuItem.Name = "文件ToolStripMenuItem";

this.文件ToolStripMenuItem.Size = new System.Drawing.Size(44, 21);

this.文件ToolStripMenuItem.Text = "文件";

//

// 读取图像ToolStripMenuItem

//

this.读取图像ToolStripMenuItem.Name = "读取图像ToolStripMenuItem";

this.读取图像ToolStripMenuItem.Size = new System.Drawing.Size(124, 22);

this.读取图像ToolStripMenuItem.Text = "读取图像";

this.读取图像ToolStripMenuItem.Click += new System.EventHandler(this.读取图像ToolStripMenuItem\_Click);

//

// 保存图像ToolStripMenuItem

//

this.保存图像ToolStripMenuItem.Name = "保存图像ToolStripMenuItem";

this.保存图像ToolStripMenuItem.Size = new System.Drawing.Size(124, 22);

this.保存图像ToolStripMenuItem.Text = "保存图像";

this.保存图像ToolStripMenuItem.Click += new System.EventHandler(this.保存图像ToolStripMenuItem\_Click);

//

// 退出ToolStripMenuItem

//

this.退出ToolStripMenuItem.Name = "退出ToolStripMenuItem";

this.退出ToolStripMenuItem.Size = new System.Drawing.Size(124, 22);

this.退出ToolStripMenuItem.Text = "退出";

this.退出ToolStripMenuItem.Click += new System.EventHandler(this.退出ToolStripMenuItem\_Click);

//

// 连接相机ToolStripMenuItem

//

this.连接相机ToolStripMenuItem.Name = "连接相机ToolStripMenuItem";

this.连接相机ToolStripMenuItem.Size = new System.Drawing.Size(124, 22);

this.连接相机ToolStripMenuItem.Text = "连接相机";

//

// 自动处理ToolStripMenuItem

//

this.自动处理ToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.打开自动处理窗口ToolStripMenuItem});

this.自动处理ToolStripMenuItem.Name = "自动处理ToolStripMenuItem";

this.自动处理ToolStripMenuItem.Size = new System.Drawing.Size(68, 21);

this.自动处理ToolStripMenuItem.Text = "自动处理";

//

// 打开自动处理窗口ToolStripMenuItem

//

this.打开自动处理窗口ToolStripMenuItem.Name = "打开自动处理窗口ToolStripMenuItem";

this.打开自动处理窗口ToolStripMenuItem.Size = new System.Drawing.Size(172, 22);

this.打开自动处理窗口ToolStripMenuItem.Text = "打开自动处理窗口";

this.打开自动处理窗口ToolStripMenuItem.Click += new System.EventHandler(this.打开自动处理窗口ToolStripMenuItem\_Click);

//

// 帮助ToolStripMenuItem

//

this.帮助ToolStripMenuItem.Name = "帮助ToolStripMenuItem";

this.帮助ToolStripMenuItem.Size = new System.Drawing.Size(44, 21);

this.帮助ToolStripMenuItem.Text = "帮助";

this.帮助ToolStripMenuItem.Click += new System.EventHandler(this.帮助ToolStripMenuItem\_Click);

//

// 关于ToolStripMenuItem

//

this.关于ToolStripMenuItem.Name = "关于ToolStripMenuItem";

this.关于ToolStripMenuItem.Size = new System.Drawing.Size(44, 21);

this.关于ToolStripMenuItem.Text = "关于";

this.关于ToolStripMenuItem.Click += new System.EventHandler(this.关于ToolStripMenuItem\_Click);

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(119, 31);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(103, 21);

this.textBox1.TabIndex = 2;

this.textBox1.TextChanged += new System.EventHandler(this.textBox1\_TextChanged);

//

// groupBox1

//

this.groupBox1.Controls.Add(this.label11);

this.groupBox1.Controls.Add(this.label10);

this.groupBox1.Controls.Add(this.trackBar4);

this.groupBox1.Controls.Add(this.trackBar3);

this.groupBox1.Controls.Add(this.comboBox4);

this.groupBox1.Controls.Add(this.comboBox3);

this.groupBox1.Controls.Add(this.label1);

this.groupBox1.Controls.Add(this.trackBar2);

this.groupBox1.Controls.Add(this.button6);

this.groupBox1.Controls.Add(this.button5);

this.groupBox1.Controls.Add(this.comboBox2);

this.groupBox1.Controls.Add(this.button4);

this.groupBox1.Controls.Add(this.button3);

this.groupBox1.Controls.Add(this.labelthreshold);

this.groupBox1.Controls.Add(this.trackBar1);

this.groupBox1.Controls.Add(this.button2);

this.groupBox1.Controls.Add(this.comboBox1);

this.groupBox1.Controls.Add(this.button1);

this.groupBox1.Location = new System.Drawing.Point(29, 52);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(344, 267);

this.groupBox1.TabIndex = 3;

this.groupBox1.TabStop = false;

this.groupBox1.Text = "图像预处理与分割";

//

// label11

//

this.label11.AutoSize = true;

this.label11.Font = new System.Drawing.Font("宋体", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label11.Location = new System.Drawing.Point(289, 120);

this.label11.Name = "label11";

this.label11.Size = new System.Drawing.Size(49, 19);

this.label11.TabIndex = 270;

this.label11.Text = "size";

//

// label10

//

this.label10.AutoSize = true;

this.label10.Font = new System.Drawing.Font("宋体", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label10.Location = new System.Drawing.Point(289, 73);

this.label10.Name = "label10";

this.label10.Size = new System.Drawing.Size(49, 19);

this.label10.TabIndex = 269;

this.label10.Text = "size";

//

// trackBar4

//

this.trackBar4.Location = new System.Drawing.Point(189, 73);

this.trackBar4.Name = "trackBar4";

this.trackBar4.Size = new System.Drawing.Size(76, 45);

this.trackBar4.TabIndex = 268;

this.trackBar4.ValueChanged += new System.EventHandler(this.trackBar4\_ValueChanged);

//

// trackBar3

//

this.trackBar3.Location = new System.Drawing.Point(189, 114);

this.trackBar3.Maximum = 20;

this.trackBar3.Name = "trackBar3";

this.trackBar3.Size = new System.Drawing.Size(76, 45);

this.trackBar3.TabIndex = 255;

this.trackBar3.ValueChanged += new System.EventHandler(this.trackBar3\_ValueChanged);

//

// comboBox4

//

this.comboBox4.FormattingEnabled = true;

this.comboBox4.Items.AddRange(new object[] {

"高斯滤波",

"均值滤波",

"中值滤波",

"递归滤波",

"sigma非线性滤波"});

this.comboBox4.Location = new System.Drawing.Point(119, 73);

this.comboBox4.Name = "comboBox4";

this.comboBox4.Size = new System.Drawing.Size(64, 20);

this.comboBox4.TabIndex = 5;

//

// comboBox3

//

this.comboBox3.FormattingEnabled = true;

this.comboBox3.Items.AddRange(new object[] {

"Frei-Chen",

"Kirsch",

"Prewitt",

"灰度腐蚀",

"emphasize",

"illuminate"});

this.comboBox3.Location = new System.Drawing.Point(119, 114);

this.comboBox3.Name = "comboBox3";

this.comboBox3.Size = new System.Drawing.Size(64, 20);

this.comboBox3.TabIndex = 265;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("宋体", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label1.Location = new System.Drawing.Point(230, 217);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(47, 19);

this.label1.TabIndex = 264;

this.label1.Text = "系数";

//

// trackBar2

//

this.trackBar2.Location = new System.Drawing.Point(119, 215);

this.trackBar2.Minimum = 1;

this.trackBar2.Name = "trackBar2";

this.trackBar2.Size = new System.Drawing.Size(105, 45);

this.trackBar2.TabIndex = 100;

this.trackBar2.Value = 2;

this.trackBar2.ValueChanged += new System.EventHandler(this.trackBar2\_ValueChanged);

//

// button6

//

this.button6.Location = new System.Drawing.Point(6, 215);

this.button6.Name = "button6";

this.button6.Size = new System.Drawing.Size(85, 21);

this.button6.TabIndex = 262;

this.button6.Text = "形态处理";

this.button6.UseVisualStyleBackColor = true;

this.button6.Click += new System.EventHandler(this.button6\_Click);

//

// button5

//

this.button5.Location = new System.Drawing.Point(6, 114);

this.button5.Name = "button5";

this.button5.Size = new System.Drawing.Size(85, 21);

this.button5.TabIndex = 261;

this.button5.Text = "图像增强";

this.button5.UseVisualStyleBackColor = true;

this.button5.Click += new System.EventHandler(this.button5\_Click);

//

// comboBox2

//

this.comboBox2.FormattingEnabled = true;

this.comboBox2.Items.AddRange(new object[] {

"手动选择",

"Otsu法",

"smooth\_histo"});

this.comboBox2.Location = new System.Drawing.Point(119, 168);

this.comboBox2.Name = "comboBox2";

this.comboBox2.Size = new System.Drawing.Size(64, 20);

this.comboBox2.TabIndex = 260;

//

// button4

//

this.button4.Location = new System.Drawing.Point(6, 73);

this.button4.Name = "button4";

this.button4.Size = new System.Drawing.Size(85, 21);

this.button4.TabIndex = 259;

this.button4.Text = "图像平滑";

this.button4.UseVisualStyleBackColor = true;

this.button4.Click += new System.EventHandler(this.button4\_Click);

//

// button3

//

this.button3.Location = new System.Drawing.Point(6, 165);

this.button3.Name = "button3";

this.button3.Size = new System.Drawing.Size(85, 21);

this.button3.TabIndex = 258;

this.button3.Text = "阈值分割";

this.button3.UseVisualStyleBackColor = true;

this.button3.Click += new System.EventHandler(this.button3\_Click);

//

// labelthreshold

//

this.labelthreshold.AutoSize = true;

this.labelthreshold.Font = new System.Drawing.Font("宋体", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.labelthreshold.Location = new System.Drawing.Point(291, 169);

this.labelthreshold.Name = "labelthreshold";

this.labelthreshold.Size = new System.Drawing.Size(47, 19);

this.labelthreshold.TabIndex = 257;

this.labelthreshold.Text = "阈值";

this.labelthreshold.Click += new System.EventHandler(this.labelthreshold\_Click);

//

// trackBar1

//

this.trackBar1.Location = new System.Drawing.Point(189, 165);

this.trackBar1.Maximum = 255;

this.trackBar1.Name = "trackBar1";

this.trackBar1.Size = new System.Drawing.Size(105, 45);

this.trackBar1.TabIndex = 256;

this.trackBar1.Scroll += new System.EventHandler(this.trackBar1\_Scroll);

this.trackBar1.ValueChanged += new System.EventHandler(this.trackBar1\_ValueChanged);

//

// button2

//

this.button2.Location = new System.Drawing.Point(6, 34);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(85, 21);

this.button2.TabIndex = 6;

this.button2.Text = "放大倍率";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// comboBox1

//

this.comboBox1.FormattingEnabled = true;

this.comboBox1.Items.AddRange(new object[] {

"50X",

"100X",

"200X",

"500X",

"1000X"});

this.comboBox1.Location = new System.Drawing.Point(119, 35);

this.comboBox1.Name = "comboBox1";

this.comboBox1.Size = new System.Drawing.Size(105, 20);

this.comboBox1.TabIndex = 5;

this.comboBox1.SelectedIndexChanged += new System.EventHandler(this.comboBox1\_SelectedIndexChanged);

//

// button1

//

this.button1.Location = new System.Drawing.Point(253, 34);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(85, 21);

this.button1.TabIndex = 3;

this.button1.Text = "灰度变换";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// groupBox2

//

this.groupBox2.Controls.Add(this.label15);

this.groupBox2.Controls.Add(this.label14);

this.groupBox2.Controls.Add(this.label13);

this.groupBox2.Controls.Add(this.label12);

this.groupBox2.Controls.Add(this.textBox8);

this.groupBox2.Controls.Add(this.label9);

this.groupBox2.Controls.Add(this.label2);

this.groupBox2.Controls.Add(this.textBox5);

this.groupBox2.Controls.Add(this.label7);

this.groupBox2.Controls.Add(this.label8);

this.groupBox2.Controls.Add(this.textBox6);

this.groupBox2.Controls.Add(this.textBox7);

this.groupBox2.Controls.Add(this.label6);

this.groupBox2.Controls.Add(this.label5);

this.groupBox2.Controls.Add(this.label4);

this.groupBox2.Controls.Add(this.label3);

this.groupBox2.Controls.Add(this.textBox4);

this.groupBox2.Controls.Add(this.textBox3);

this.groupBox2.Controls.Add(this.button9);

this.groupBox2.Controls.Add(this.textBox2);

this.groupBox2.Controls.Add(this.button8);

this.groupBox2.Controls.Add(this.button7);

this.groupBox2.Controls.Add(this.textBox1);

this.groupBox2.Location = new System.Drawing.Point(29, 325);

this.groupBox2.Name = "groupBox2";

this.groupBox2.Size = new System.Drawing.Size(344, 318);

this.groupBox2.TabIndex = 4;

this.groupBox2.TabStop = false;

this.groupBox2.Text = "铁谱图像分析";

//

// label15

//

this.label15.AutoSize = true;

this.label15.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label15.Location = new System.Drawing.Point(266, 149);

this.label15.Name = "label15";

this.label15.Size = new System.Drawing.Size(28, 14);

this.label15.TabIndex = 286;

this.label15.Text = "μm";

//

// label14

//

this.label14.AutoSize = true;

this.label14.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label14.Location = new System.Drawing.Point(147, 150);

this.label14.Name = "label14";

this.label14.Size = new System.Drawing.Size(28, 14);

this.label14.TabIndex = 285;

this.label14.Text = "μm";

//

// label13

//

this.label13.AutoSize = true;

this.label13.Font = new System.Drawing.Font("宋体", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label13.Location = new System.Drawing.Point(247, 58);

this.label13.Name = "label13";

this.label13.Size = new System.Drawing.Size(0, 19);

this.label13.TabIndex = 284;

//

// label12

//

this.label12.AutoSize = true;

this.label12.Font = new System.Drawing.Font("宋体", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label12.Location = new System.Drawing.Point(228, 32);

this.label12.Name = "label12";

this.label12.Size = new System.Drawing.Size(19, 19);

this.label12.TabIndex = 283;

this.label12.Text = "%";

//

// textBox8

//

this.textBox8.Location = new System.Drawing.Point(61, 252);

this.textBox8.Name = "textBox8";

this.textBox8.Size = new System.Drawing.Size(78, 21);

this.textBox8.TabIndex = 282;

//

// label9

//

this.label9.AutoSize = true;

this.label9.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label9.Location = new System.Drawing.Point(12, 254);

this.label9.Name = "label9";

this.label9.Size = new System.Drawing.Size(35, 14);

this.label9.TabIndex = 281;

this.label9.Text = "片状";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label2.Location = new System.Drawing.Point(6, 291);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(49, 14);

this.label2.TabIndex = 280;

this.label2.Text = "总个数";

//

// textBox5

//

this.textBox5.Location = new System.Drawing.Point(61, 284);

this.textBox5.Name = "textBox5";

this.textBox5.Size = new System.Drawing.Size(78, 21);

this.textBox5.TabIndex = 279;

//

// label7

//

this.label7.AutoSize = true;

this.label7.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label7.Location = new System.Drawing.Point(12, 219);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(35, 14);

this.label7.TabIndex = 278;

this.label7.Text = "条状";

//

// label8

//

this.label8.AutoSize = true;

this.label8.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label8.Location = new System.Drawing.Point(12, 188);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(35, 14);

this.label8.TabIndex = 277;

this.label8.Text = "圆状";

//

// textBox6

//

this.textBox6.Location = new System.Drawing.Point(61, 217);

this.textBox6.Name = "textBox6";

this.textBox6.Size = new System.Drawing.Size(78, 21);

this.textBox6.TabIndex = 276;

//

// textBox7

//

this.textBox7.Location = new System.Drawing.Point(61, 182);

this.textBox7.Name = "textBox7";

this.textBox7.Size = new System.Drawing.Size(78, 21);

this.textBox7.TabIndex = 275;

//

// label6

//

this.label6.AutoSize = true;

this.label6.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label6.Location = new System.Drawing.Point(6, 153);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(49, 14);

this.label6.TabIndex = 274;

this.label6.Text = "长轴长";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label5.Location = new System.Drawing.Point(181, 151);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(28, 14);

this.label5.TabIndex = 273;

this.label5.Text = "MAX";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label4.Location = new System.Drawing.Point(58, 153);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(28, 14);

this.label4.TabIndex = 272;

this.label4.Text = "MIN";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Font = new System.Drawing.Font("宋体", 10.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label3.Location = new System.Drawing.Point(6, 189);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(0, 14);

this.label3.TabIndex = 271;

//

// textBox4

//

this.textBox4.Location = new System.Drawing.Point(215, 149);

this.textBox4.Name = "textBox4";

this.textBox4.Size = new System.Drawing.Size(49, 21);

this.textBox4.TabIndex = 268;

this.textBox4.KeyPress += new System.Windows.Forms.KeyPressEventHandler(this.textBox4\_KeyPress);

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(92, 150);

this.textBox3.Name = "textBox3";

this.textBox3.Size = new System.Drawing.Size(49, 21);

this.textBox3.TabIndex = 267;

this.textBox3.KeyPress += new System.Windows.Forms.KeyPressEventHandler(this.textBox3\_KeyPress);

//

// button9

//

this.button9.Location = new System.Drawing.Point(6, 112);

this.button9.Name = "button9";

this.button9.Size = new System.Drawing.Size(85, 21);

this.button9.TabIndex = 266;

this.button9.Text = "磨粒分类";

this.button9.UseVisualStyleBackColor = true;

this.button9.Click += new System.EventHandler(this.button9\_Click);

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(119, 73);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(103, 21);

this.textBox2.TabIndex = 265;

//

// button8

//

this.button8.Location = new System.Drawing.Point(6, 73);

this.button8.Name = "button8";

this.button8.Size = new System.Drawing.Size(85, 21);

this.button8.TabIndex = 264;

this.button8.Text = "磨粒个数计算";

this.button8.UseVisualStyleBackColor = true;

this.button8.Click += new System.EventHandler(this.button8\_Click);

//

// button7

//

this.button7.Location = new System.Drawing.Point(6, 30);

this.button7.Name = "button7";

this.button7.Size = new System.Drawing.Size(98, 21);

this.button7.TabIndex = 263;

this.button7.Text = "磨粒屏占比计算";

this.button7.UseVisualStyleBackColor = true;

this.button7.Click += new System.EventHandler(this.button7\_Click);

//

// button10

//

this.button10.Location = new System.Drawing.Point(496, 602);

this.button10.Name = "button10";

this.button10.Size = new System.Drawing.Size(116, 41);

this.button10.TabIndex = 5;

this.button10.Text = "保存数据";

this.button10.UseVisualStyleBackColor = true;

this.button10.Click += new System.EventHandler(this.button10\_Click);

//

// button11

//

this.button11.Location = new System.Drawing.Point(942, 602);

this.button11.Name = "button11";

this.button11.Size = new System.Drawing.Size(116, 41);

this.button11.TabIndex = 6;

this.button11.Text = "输出EXCEL";

this.button11.UseVisualStyleBackColor = true;

this.button11.Click += new System.EventHandler(this.button11\_Click);

//

// saveImage

//

this.saveImage.Filter = "图像(\*.jpg)|\*.jpg";

this.saveImage.InitialDirectory = "F:\\Projects\\C#\\Halcon\_region\_sort\_1";

//

// button12

//

this.button12.Location = new System.Drawing.Point(719, 602);

this.button12.Name = "button12";

this.button12.Size = new System.Drawing.Size(116, 41);

this.button12.TabIndex = 7;

this.button12.Text = "查看数据";

this.button12.UseVisualStyleBackColor = true;

this.button12.Click += new System.EventHandler(this.button12\_Click);

//

// HALCON\_SORT

//

this.AllowDrop = true;

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.AutoSize = true;

this.ClientSize = new System.Drawing.Size(1210, 654);

this.Controls.Add(this.button12);

this.Controls.Add(this.button11);

this.Controls.Add(this.button10);

this.Controls.Add(this.groupBox2);

this.Controls.Add(this.groupBox1);

this.Controls.Add(this.hWindow\_main);

this.Controls.Add(this.menuStrip1);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.MainMenuStrip = this.menuStrip1;

this.MaximizeBox = false;

this.Name = "HALCON\_SORT";

this.Text = "铁谱图像定量分析软件";

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.HALCON\_SORT\_FormClosing);

this.Load += new System.EventHandler(this.HALCON\_SORT\_Load);

this.menuStrip1.ResumeLayout(false);

this.menuStrip1.PerformLayout();

this.groupBox1.ResumeLayout(false);

this.groupBox1.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar4)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar3)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar2)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).EndInit();

this.groupBox2.ResumeLayout(false);

this.groupBox2.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private HalconDotNet.HWindowControl hWindow\_main;

private System.Windows.Forms.MenuStrip menuStrip1;

private System.Windows.Forms.ToolStripMenuItem 文件ToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem 读取图像ToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem 保存图像ToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem 自动处理ToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem 打开自动处理窗口ToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem 退出ToolStripMenuItem;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.ToolStripMenuItem 帮助ToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem 关于ToolStripMenuItem;

private System.Windows.Forms.GroupBox groupBox1;

private System.Windows.Forms.ComboBox comboBox1;

private System.Windows.Forms.Button button1;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.TrackBar trackBar1;

private System.Windows.Forms.Label labelthreshold;

private System.Windows.Forms.Button button4;

private System.Windows.Forms.Button button3;

private System.Windows.Forms.ComboBox comboBox2;

private System.Windows.Forms.Button button5;

private System.Windows.Forms.ToolStripMenuItem 连接相机ToolStripMenuItem;

private System.Windows.Forms.Button button6;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TrackBar trackBar2;

private System.Windows.Forms.GroupBox groupBox2;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label8;

private System.Windows.Forms.TextBox textBox6;

private System.Windows.Forms.TextBox textBox7;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.TextBox textBox4;

private System.Windows.Forms.TextBox textBox3;

private System.Windows.Forms.Button button9;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.Button button8;

private System.Windows.Forms.Button button7;

private System.Windows.Forms.TextBox textBox8;

private System.Windows.Forms.Label label9;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.TextBox textBox5;

private System.Windows.Forms.Button button10;

private System.Windows.Forms.Button button11;

private System.Windows.Forms.ComboBox comboBox4;

private System.Windows.Forms.ComboBox comboBox3;

private System.Windows.Forms.Label label11;

private System.Windows.Forms.Label label10;

private System.Windows.Forms.TrackBar trackBar4;

private System.Windows.Forms.TrackBar trackBar3;

private System.Windows.Forms.Label label13;

private System.Windows.Forms.Label label12;

private System.Windows.Forms.SaveFileDialog saveImage;

private System.Windows.Forms.Label label15;

private System.Windows.Forms.Label label14;

private System.Windows.Forms.Button button12;

}

}

# HALCON\_SORT.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using HalconDotNet;

using System.IO;

namespace Halcon\_region\_sort

{

public partial class HALCON\_SORT : Form

{

public static HALCON\_SORT form0 = null;

public HALCON\_SORT()

{

InitializeComponent();

form0 = this;

}

//private HalconWindow wch = new HalconWindow();

public HTuple hv\_ExpDefaultWinHandle;

// Local iconic variables 定义图像变量

public HObject ho\_Image, ho\_GrayImage, ho\_Regions, ho\_meanImage, ho\_ImageEdge;

public HObject ho\_Connection, ho\_RegionResult, ho\_SelectedRegions, ho\_ImageInvert;

public HObject ho\_RegionFillUp, ho\_RegionDilation, ho\_RegionUnion;

public HObject ho\_ImageReduced, ho\_Regions1, ho\_RegionErosion;

public HObject ho\_ConnectedRegions, ho\_SelectedRegions1, ho\_RegionDilation1;

public HObject ho\_SortedRegions, ho\_ObjectSelected\_1 = null;

// Local control variables 定义控制变量

public HTuple hv\_Width = null, hv\_Height = null, hv\_WindowHandle = new HTuple();

public HTuple hv\_UsedThreshold = null, hv\_UsedThreshold1 = null;

public HTuple hv\_Number = null, hv\_Area = null, hv\_Row = null;

public HTuple hv\_Column = null, hv\_i = null, hv\_Area1 = null;

public HTuple hv\_Row1 = null, hv\_Column1 = null, hv\_Area2 = null;

public HTuple hv\_area = null, hv\_bizhong = null,hv\_ContLength = null;

public HTuple hv\_Circularity = null, hv\_Ra = null, hv\_Rb = null;

public HTuple hv\_Phi = null, hv\_Quot = null, hv\_all = null, hv\_ro = null;//颗粒方向、长宽比、用于满足长度要求颗粒和圆形颗粒的计数

public HTuple hv\_ti = null, hv\_pi = null; //用于条形颗粒和片状颗粒的计数

public int min, max; //定义用于筛选颗粒的最大最小长度值

public string Pathimg = "";

public String strYMDH;

// 主窗体加载事件

private void HALCON\_SORT\_Load(object sender, EventArgs e)

{

HOperatorSet.GenEmptyObj(out ho\_Image);

HOperatorSet.GenEmptyObj(out ho\_GrayImage);

HOperatorSet.GenEmptyObj(out ho\_meanImage);

HOperatorSet.GenEmptyObj(out ho\_ImageEdge);

HOperatorSet.GenEmptyObj(out ho\_ImageInvert);

HOperatorSet.GenEmptyObj(out ho\_Regions);

HOperatorSet.GenEmptyObj(out ho\_Connection);

HOperatorSet.GenEmptyObj(out ho\_RegionResult);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions);

HOperatorSet.GenEmptyObj(out ho\_RegionFillUp);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation);

HOperatorSet.GenEmptyObj(out ho\_RegionUnion);

HOperatorSet.GenEmptyObj(out ho\_ImageReduced);

HOperatorSet.GenEmptyObj(out ho\_Regions1);

HOperatorSet.GenEmptyObj(out ho\_RegionErosion);

HOperatorSet.GenEmptyObj(out ho\_ConnectedRegions);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions1);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation1);

HOperatorSet.GenEmptyObj(out ho\_SortedRegions);

HOperatorSet.GenEmptyObj(out ho\_ObjectSelected\_1);

hv\_ExpDefaultWinHandle = hWindow\_main.HalconWindow; //获取这个form中的halcon的HWindowControl窗口的窗口句柄

}

private void hWindowControl1\_HMouseMove(object sender, HalconDotNet.HMouseEventArgs e)

{

}

#region(读取图像)

private void 读取图像ToolStripMenuItem\_Click(object sender, EventArgs e)

{

hv\_ExpDefaultWinHandle = hWindow\_main.HalconWindow;

//用于打开文件夹

OpenFileDialog openFile = new OpenFileDialog();

openFile.Filter = "JPG文件(\*.jpg)|\*.jpg|所有文件(\*.\*)|\*.\*||";

if (openFile.ShowDialog() == DialogResult.OK)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle); //清空显示

Pathimg = openFile.FileName;

HOperatorSet.ReadImage(out ho\_Image, Pathimg);

HOperatorSet.DispObj(ho\_Image, hv\_ExpDefaultWinHandle);

//wch.DispImageFit(ho\_Image, hv\_ExpDefaultWinHandle);

HTuple hv\_HeightWin, hv\_WidthWin;

HOperatorSet.GetImageSize(ho\_Image, out hv\_HeightWin, out hv\_WidthWin); // 获取输入图像的尺寸

String str\_imgSize = String.Format("Size:{0}x{1}", hv\_HeightWin, hv\_WidthWin);

//wch.disp\_message(hv\_ExpDefaultWinHandle, str\_imgSize, "window", hv\_ExpDefaultWinHandle.Height - 20, 1, "blue", "false");

}

}

#endregion

private void 打开自动处理窗口ToolStripMenuItem\_Click(object sender, EventArgs e)

{

halcon\_form halconform = new halcon\_form();

halconform.Show();

this.Hide();

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void xToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void trackBar1\_ValueChanged(object sender, EventArgs e)

{

labelthreshold.Text = trackBar1.Value.ToString();

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

}

private void labelthreshold\_Click(object sender, EventArgs e)

{

}

private void trackBar2\_ValueChanged(object sender, EventArgs e)

{

label1.Text = trackBar2.Value.ToString();

}

private void 关于ToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("铁谱图像分析\n版本1.0.1\n设计：毕珂 ");

}

// 图像分割部分

# region（图像分割部分）

private void button3\_Click(object sender, EventArgs e)

{

if (ho\_ImageEdge.CountObj() != 0 || ho\_meanImage.CountObj() != 0)

{

if (ho\_ImageEdge.CountObj() != 0)

{

int threshold\_1 = trackBar1.Value;

int mode = comboBox2.SelectedIndex;

switch (mode)

{

case 0:

ho\_Regions.Dispose();

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

HOperatorSet.Threshold(ho\_ImageEdge, out ho\_Regions, 0, threshold\_1);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_Regions, hv\_ExpDefaultWinHandle);

break;

case 1:

ho\_Regions.Dispose();

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

HOperatorSet.BinaryThreshold(ho\_ImageEdge, out ho\_Regions, "max\_separability", "dark", out hv\_UsedThreshold);

HOperatorSet.DispObj(ho\_Regions, hv\_ExpDefaultWinHandle);

break;

case 2:

ho\_Regions.Dispose();

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

HOperatorSet.BinaryThreshold(ho\_ImageEdge, out ho\_Regions, "smooth\_histo", "dark", out hv\_UsedThreshold);

HOperatorSet.DispObj(ho\_Regions, hv\_ExpDefaultWinHandle);

break;

default:

MessageBox.Show("请选择处理模式");

break;

}

}

else

{

MessageBox.Show("请进行图像增强处理");

}

}

else

{

if (ho\_GrayImage.CountObj() != 0)

{

int threshold\_1 = trackBar1.Value;

int mode = comboBox2.SelectedIndex;

switch (mode)

{

case 0:

ho\_Regions.Dispose();

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

HOperatorSet.Threshold(ho\_GrayImage, out ho\_Regions, 0, threshold\_1);

HOperatorSet.DispObj(ho\_GrayImage, hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_Regions, hv\_ExpDefaultWinHandle);

break;

case 1:

ho\_Regions.Dispose();

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_GrayImage, hv\_ExpDefaultWinHandle);

HOperatorSet.BinaryThreshold(ho\_GrayImage, out ho\_Regions, "max\_separability", "dark", out hv\_UsedThreshold);

HOperatorSet.DispObj(ho\_Regions, hv\_ExpDefaultWinHandle);

break;

case 2:

ho\_Regions.Dispose();

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_GrayImage, hv\_ExpDefaultWinHandle);

HOperatorSet.BinaryThreshold(ho\_GrayImage, out ho\_Regions, "smooth\_histo", "dark", out hv\_UsedThreshold);

HOperatorSet.DispObj(ho\_Regions, hv\_ExpDefaultWinHandle);

break;

default:

MessageBox.Show("请选择处理模式");

break;

}

}

else

{

MessageBox.Show("请先进行灰度处理");

}

}

}

# endregion

// 对图像进行灰度变换

# region(灰度变换)

private void button1\_Click(object sender, EventArgs e)

{

if (ho\_Image.CountObj()!=0)

{

ho\_GrayImage.Dispose();

HOperatorSet.Rgb1ToGray(ho\_Image, out ho\_GrayImage);

HOperatorSet.DispObj(ho\_GrayImage, hv\_ExpDefaultWinHandle);

}

else

{

MessageBox.Show("请先读取图像");

}

}

#endregion

#region(图像平滑)

private void button4\_Click(object sender, EventArgs e)

{

if (ho\_GrayImage.CountObj() != 0)

{

int mean\_size = trackBar4.Value;//定义图像增强参数

int mode = comboBox4.SelectedIndex;

switch (mode)

{

case 0:

if (mean\_size % 2 == 1)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_meanImage.Dispose();

HOperatorSet.GaussFilter(ho\_GrayImage, out ho\_meanImage, mean\_size);

HOperatorSet.DispObj(ho\_meanImage, hv\_ExpDefaultWinHandle);

break;

}

else

{

MessageBox.Show("请输入大于1的奇数");

break;

}

case 1:

if (mean\_size >= 1)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_meanImage.Dispose();

HOperatorSet.MeanImage(ho\_GrayImage, out ho\_meanImage, mean\_size, mean\_size);

HOperatorSet.DispObj(ho\_meanImage, hv\_ExpDefaultWinHandle);

break;

}

else

{

MessageBox.Show("请输入系数");

break;

}

case 2:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_meanImage.Dispose();

HOperatorSet.MedianImage(ho\_GrayImage, out ho\_meanImage, "circle", mean\_size, "mirrored");

HOperatorSet.DispObj(ho\_meanImage, hv\_ExpDefaultWinHandle);

break;

case 3:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_meanImage.Dispose();

HOperatorSet.SmoothImage(ho\_GrayImage, out ho\_meanImage, "deriche2", mean\_size);

HOperatorSet.DispObj(ho\_meanImage, hv\_ExpDefaultWinHandle);

break;

case 4:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_meanImage.Dispose();

HOperatorSet.SigmaImage(ho\_GrayImage, out ho\_meanImage, mean\_size, mean\_size, 3);

HOperatorSet.DispObj(ho\_meanImage, hv\_ExpDefaultWinHandle);

break;

default:

MessageBox.Show("请选择处理模式");

break;

}

}

else

{

MessageBox.Show("请先进行灰度变换");

}

}

#endregion

private void trackBar4\_ValueChanged(object sender, EventArgs e)

{

label10.Text = trackBar4.Value.ToString();

}

private void trackBar3\_ValueChanged(object sender, EventArgs e)

{

label11.Text = trackBar3.Value.ToString();

}

#region(图像增强)

private void button5\_Click(object sender, EventArgs e)

{

if (ho\_meanImage.CountObj()!=0)

#region(处理平滑图像)

{

int em\_size = trackBar3.Value;//定义图像增强参数

int mode = comboBox3.SelectedIndex;

switch (mode)

{

case 0:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.FreiAmp(ho\_meanImage, out ho\_ImageEdge);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

case 1:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.KirschAmp(ho\_meanImage, out ho\_ImageEdge);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

case 2:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.PrewittAmp(ho\_meanImage, out ho\_ImageEdge);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

case 3:

if (em\_size>0)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.GrayErosionShape(ho\_meanImage, out ho\_ImageEdge, em\_size, em\_size, "rhombus");

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

}

else

{

MessageBox.Show("请输入系数");

break;

}

case 4:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.Emphasize(ho\_meanImage, out ho\_ImageEdge, 7, 7, em\_size);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

case 5:

if (em\_size > 0 & em\_size < 13)

{

//HObject a;

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.Illuminate(ho\_meanImage, out ho\_ImageEdge, 41, 41, em\_size);

ho\_ImageInvert.Dispose();

HOperatorSet.InvertImage(ho\_ImageEdge, out ho\_ImageInvert);

ho\_ImageEdge = ho\_ImageInvert;

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

}

else

{

MessageBox.Show("请输入1-12之内的系数");

break;

}

default:

MessageBox.Show("请选择处理模式");

break;

}

}

#endregion

else

{

#region(处理灰度图)

if (ho\_GrayImage.CountObj() != 0)

{

int em\_size = trackBar3.Value;//定义图像增强参数

int mode = comboBox3.SelectedIndex;

switch (mode)

{

case 0:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.FreiAmp(ho\_GrayImage, out ho\_ImageEdge);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

case 1:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.KirschAmp(ho\_GrayImage, out ho\_ImageEdge);

if (HDevWindowStack.IsOpen())

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

case 2:

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.PrewittAmp(ho\_GrayImage, out ho\_ImageEdge);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

case 3:

if (em\_size > 0)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.GrayErosionShape(ho\_GrayImage, out ho\_ImageEdge, em\_size, em\_size, "rhombus");

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

}

else

{

MessageBox.Show("请输入系数");

break;

}

case 4:

if (em\_size > 0 & em\_size<13)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.Emphasize(ho\_GrayImage, out ho\_ImageEdge, 7, 7, em\_size);

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

}

else

{

MessageBox.Show("请输入1-12之内的系数");

break;

}

case 5:

if (em\_size > 0 & em\_size < 13)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);

ho\_ImageEdge.Dispose();

HOperatorSet.Illuminate(ho\_GrayImage, out ho\_ImageEdge, 41, 41, em\_size);

ho\_ImageInvert.Dispose();

HOperatorSet.InvertImage(ho\_ImageEdge, out ho\_ImageInvert);

ho\_ImageEdge = ho\_ImageInvert;

HOperatorSet.DispObj(ho\_ImageEdge, hv\_ExpDefaultWinHandle);

break;

}

else

{

MessageBox.Show("请输入1-12之内的系数");

break;

}

default:

MessageBox.Show("请选择处理模式");

break;

}

}

#endregion

else

{

MessageBox.Show("请先进行灰度处理");

}

}

}

#endregion

private void button7\_Click(object sender, EventArgs e)

{

if (ho\_Regions.CountObj() != 0)

{

HOperatorSet.GetImageSize(ho\_Image, out hv\_Width, out hv\_Height);

HOperatorSet.AreaCenter(ho\_Regions, out hv\_Area1, out hv\_Row1, out hv\_Column1);

hv\_Area2 = hv\_Area1.TupleReal();

hv\_area = ((hv\_Width \* hv\_Height)).TupleReal();

hv\_bizhong = 100\*hv\_Area2 / hv\_area;

///MessageBox.Show("磨粒覆盖面积比" + hv\_bizhong + "%");

textBox1.Text = hv\_bizhong.ToString();

}

else

{

MessageBox.Show("请先进行图像预处理");

}

}

private void button6\_Click(object sender, EventArgs e)

{

int size = trackBar2.Value;

if (ho\_Regions.CountObj()!=0)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle);//清除图像

//图像腐蚀

ho\_RegionErosion.Dispose();

HOperatorSet.ErosionCircle(ho\_Regions, out ho\_RegionErosion, size);

//区域联通计算

ho\_Connection.Dispose();

HOperatorSet.Connection(ho\_RegionErosion, out ho\_Connection);

//选择合适的区域

ho\_SelectedRegions.Dispose();

HOperatorSet.SelectShape(ho\_Connection, out ho\_SelectedRegions, "area", "and", 1000,9999900000);

//填充区域孔洞

ho\_RegionFillUp.Dispose();

HOperatorSet.FillUp(ho\_SelectedRegions, out ho\_RegionFillUp);

//对颗粒进行膨胀处理

ho\_RegionDilation1.Dispose();

HOperatorSet.DilationCircle(ho\_RegionFillUp, out ho\_RegionDilation1, size);

//显示

HOperatorSet.DispObj(ho\_GrayImage, hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_RegionDilation1, hv\_ExpDefaultWinHandle);

}

else

{

MessageBox.Show("请先进行图像分割");

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button8\_Click(object sender, EventArgs e)

{

if (ho\_RegionDilation1.CountObj()!=0)

{

// 数颗粒的个数

HOperatorSet.CountObj(ho\_RegionDilation1, out hv\_Number);

// 对颗粒按照第一个点的行坐标进行排序

ho\_SortedRegions.Dispose();

HOperatorSet.SortRegion(ho\_RegionDilation1, out ho\_SortedRegions, "first\_point", "false", "row");

// 求颗粒的面积和中心坐标

HOperatorSet.AreaCenter(ho\_SortedRegions, out hv\_Area, out hv\_Row, out hv\_Column);

// 显示图像和选中的颗粒

HOperatorSet.DispObj(ho\_Image, hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_SortedRegions, hv\_ExpDefaultWinHandle);

//显示图像的代码

HTuple end\_val47 = hv\_Number;

HTuple step\_val47 = 1;

for (hv\_i = 1; hv\_i.Continue(end\_val47, step\_val47); hv\_i = hv\_i.TupleAdd(step\_val47))

{

ho\_ObjectSelected\_1.Dispose();

HOperatorSet.SelectObj(ho\_SortedRegions, out ho\_ObjectSelected\_1, hv\_i);

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, "red");

HOperatorSet.DispObj(ho\_ObjectSelected\_1, hv\_ExpDefaultWinHandle);

HOperatorSet.SetTposition(hv\_ExpDefaultWinHandle, hv\_Row.TupleSelect(hv\_i - 1),

hv\_Column.TupleSelect(hv\_i - 1));

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, "white");

HOperatorSet.WriteString(hv\_ExpDefaultWinHandle, hv\_i);

//write\_string (WindowHandle, Area[i-1])

}

textBox2.Text = hv\_Number.ToString();

}

else

{

MessageBox.Show("请先进行形态处理");

}

}

private void 帮助ToolStripMenuItem\_Click(object sender, EventArgs e)

{

HALCON\_BIAO help = new HALCON\_BIAO();

help.Show();

}

private void 保存图像ToolStripMenuItem\_Click(object sender, EventArgs e)

{

saveImage.FileName = "TPT\_01";

saveImage.Title = "另存为";

if (saveImage.ShowDialog() == DialogResult.OK)

{

// 保存文件

HOperatorSet.DumpWindow(hv\_ExpDefaultWinHandle, "jpeg", saveImage.FileName);

//ho\_Image.Dispose();

}

}

private void 退出ToolStripMenuItem\_Click(object sender, EventArgs e)

{

Close();

}

private void button9\_Click(object sender, EventArgs e)

{

if (ho\_SortedRegions.CountObj() != 0)

{

if (textBox3.Text != "" && textBox4.Text != "")

{

min = int.Parse(textBox3.Text);

max = int.Parse(textBox4.Text);

if (min >= 0)

{

//计算颗粒的长度

HOperatorSet.Contlength(ho\_SortedRegions, out hv\_ContLength);

//计算颗粒的圆度

HOperatorSet.Circularity(ho\_SortedRegions, out hv\_Circularity);

//计算颗粒等效椭圆的长短轴

HOperatorSet.EllipticAxis(ho\_SortedRegions, out hv\_Ra, out hv\_Rb, out hv\_Phi);

//求长短轴之比

HOperatorSet.TupleDiv(hv\_Ra, hv\_Rb, out hv\_Quot);

//定义总个数

hv\_all = 0;

//定义圆形颗粒个数

hv\_ro = 0;

//定义条形颗粒个数

hv\_ti = 0;

//定义片状颗粒个数

hv\_pi = 0;

HTuple end\_val78 = hv\_Number - 1;

HTuple step\_val78 = 1;

for (hv\_i = 0; hv\_i.Continue(end\_val78, step\_val78); hv\_i = hv\_i.TupleAdd(step\_val78))

{

if ((int)((new HTuple(((hv\_ContLength.TupleSelect(hv\_i))).TupleGreater(min))).TupleAnd(new HTuple(((hv\_ContLength.TupleSelect(hv\_i))).TupleLess(max)))) != 0)

{

hv\_all = hv\_all + 1;

if ((int)(new HTuple(((hv\_Circularity.TupleSelect(hv\_i))).TupleGreater(0.65))) != 0)

{

hv\_ro = hv\_ro + 1;

}

if ((int)(new HTuple(((hv\_Quot.TupleSelect(hv\_i))).TupleGreater(3))) != 0)

{

hv\_ti = hv\_ti + 1;

}

else

{

hv\_pi = hv\_pi + 1;

}

}

}

textBox7.Text = hv\_ro.ToString();

textBox6.Text = hv\_ti.ToString();

textBox8.Text = hv\_pi.ToString();

textBox5.Text = hv\_all.ToString();

}

}

else

{

MessageBox.Show("请输入长度范围");

}

}

else

{

MessageBox.Show("请先进行磨粒个数计算");

}

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

//IsNumber 的作用是判断输入按键是否为数字

//（char）8是退格键的值，可允许用户敲退格键对输入进行更改

//针对其他按键输入则提示错误，不允许输入

//if ((Char.IsNumber(e.KeyChar)) || e.KeyChar == (char)8)

//{

// e.Handled = true;

//}

//else

//{

// MessageBox.Show("请输入数字！");

//}

if (e.KeyChar == 0x20) e.KeyChar = (char)0; //禁止空格键

if ((e.KeyChar == 0x2D) && (((TextBox)sender).Text.Length == 0)) return; //处理负数

if (e.KeyChar > 0x20)

{

try

{

double.Parse(((TextBox)sender).Text + e.KeyChar.ToString());

}

catch

{

e.KeyChar = (char)0;

MessageBox.Show("请输入数字！");//处理非法字符

}

}

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == 0x20) e.KeyChar = (char)0; //禁止空格键

if ((e.KeyChar == 0x2D) && (((TextBox)sender).Text.Length == 0)) return; //处理负数

if (e.KeyChar > 0x20)

{

try

{

double.Parse(((TextBox)sender).Text + e.KeyChar.ToString());

}

catch

{

e.KeyChar = (char)0; //处理非法字符

}

}

}

private void button10\_Click(object sender, EventArgs e)

{

if (hv\_all != null)

{

System.DateTime currentTime = new System.DateTime();//定义时间类

currentTime = DateTime.Now; //获取当前系统的时间

int strD = currentTime.Day;

int strM = currentTime.Month;

int strY = currentTime.Year;

int strH = currentTime.Hour;

int strMIN = currentTime.Minute;

strYMDH = "" + strY + strM + strD + "\_" + strH + strMIN;//定义数据存储文件名

bool txtexists = Directory.Exists("..\\text");//判断文件存储文件夹是否存在

if (!txtexists)

{

Directory.CreateDirectory("..\\text");//如果文件存储文件夹不存在，则创建

}

HOperatorSet.DumpWindow(hv\_ExpDefaultWinHandle, "jpeg", "..\\text\\" + strYMDH + ".txt");

FileStream fs = new FileStream("..\\text\\" + strYMDH + ".txt", FileMode.Create);

StreamWriter sw = new StreamWriter(fs);

//开始写入

String dataImage = "颗粒屏占比为：" + hv\_bizhong +"%"+ "\r\n" + "颗粒总个数" + hv\_Number + "\r\n" + "长度在" + min + "和" + max + "之间的颗粒总个数" + hv\_all + "\r\n" + "圆形颗粒个数为" + hv\_ro + "\r\n" + "条形颗粒个数为" + hv\_ti + "\r\n" + "片形颗粒个数为" + hv\_pi;

sw.Write(dataImage);

//清空缓冲区

sw.Flush();

//关闭流

sw.Close();

fs.Close();

MessageBox.Show("数据保存成功"+"("+"..\\text\\" + strYMDH+")");

}

else

{

MessageBox.Show("请先完成图像处理");

}

}

private void button11\_Click(object sender, EventArgs e)

{

System.DateTime currentTime = new System.DateTime();//定义时间类

currentTime = DateTime.Now; //获取当前系统的时间

int strD = currentTime.Day;

int strM = currentTime.Month;

int strY = currentTime.Year;

int strH = currentTime.Hour;

int strMIN = currentTime.Minute;

String strYMDH = "" + strY + strM + strD +"\_"+ strH + strMIN ;//合并转化成字符串

MessageBox.Show("北京时间"+strYMDH+"距离完成还有一会");

}

private void button12\_Click(object sender, EventArgs e)

{

bool txtexists = Directory.Exists("..\\text");//判断文件存储文件夹是否存在

if (!txtexists)

{

MessageBox.Show("请先保存数据");

}

else

{

EXCEL\_out excelout = new EXCEL\_out("..\\text\\");

excelout.Show();

}

}

// 关闭提醒功能

private void HALCON\_SORT\_FormClosing(object sender, FormClosingEventArgs e)

{

if (MessageBox.Show("确定关闭？", "确定关闭",MessageBoxButtons.OKCancel)==DialogResult.Cancel)

{

e.Cancel = true;

}

}

}

}

# HalconWindow.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using HalconDotNet; //引用halcon

namespace Halcon\_Programs

{

class HalconWindow

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//缩放显示图像使用的类私有全局变量

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private int zoom\_beginRow, zoom\_beginCol, zoom\_endRow, zoom\_endCol; // 设定图像的窗口显示部分

private int current\_beginRow, current\_beginCol, current\_endRow, current\_endCol; // 获取图像的当前显示部分

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//设置hanlcon的显示窗体

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void SetWindow(HWindowControl hWindowControl)

{

HTuple hWindowHandle1 = hWindowControl.HalconWindow;

//系统参数初始化

HOperatorSet.SetColor(hWindowHandle1, "red");

HOperatorSet.SetDraw(hWindowHandle1, "margin");

//HOperatorSet.SetColored(hWindowHandle, 12);

HOperatorSet.SetSystem("int\_zooming", "true");

HOperatorSet.SetSystem("border\_shape\_models", "false");

HOperatorSet.SetSystem("tsp\_width", 6000);

HOperatorSet.SetSystem("tsp\_height", 6000);

}

//按照指定的中心缩放当前图像

public void DispImageZoom(HObject t\_image, HWindowControl hw\_Ctrl, HTuple mode, double Mouse\_row, double Mouse\_col)

{

if (t\_image != null)

{

HTuple width, height;

HOperatorSet.GetImageSize(t\_image, out width, out height); //获取图像的尺寸

int hv\_imageWidth, hv\_imageHeight;

hv\_imageWidth = width;

hv\_imageHeight = height;

try

{

hw\_Ctrl.HalconWindow.GetPart(out current\_beginRow, out current\_beginCol, out current\_endRow, out current\_endCol);

}

catch (Exception ex)

{

return;

}

if (mode > 0) // 放大图像

{

zoom\_beginRow = (int)(current\_beginRow + (Mouse\_row - current\_beginRow) \* 0.300d);

zoom\_beginCol = (int)(current\_beginCol + (Mouse\_col - current\_beginCol) \* 0.300d);

zoom\_endRow = (int)(current\_endRow - (current\_endRow - Mouse\_row) \* 0.300d);

zoom\_endCol = (int)(current\_endCol - (current\_endCol - Mouse\_col) \* 0.300d);

}

else // 缩小图像

{

zoom\_beginRow = (int)(Mouse\_row - (Mouse\_row - current\_beginRow) / 0.700d);

zoom\_beginCol = (int)(Mouse\_col - (Mouse\_col - current\_beginCol) / 0.700d);

zoom\_endRow = (int)(Mouse\_row + (current\_endRow - Mouse\_row) / 0.700d);

zoom\_endCol = (int)(Mouse\_col + (current\_endCol - Mouse\_col) / 0.700d);

}

try

{

int hw\_width, hw\_height;

hw\_width = hw\_Ctrl.WindowSize.Width;

hw\_height = hw\_Ctrl.WindowSize.Height;

bool \_isOutOfArea = true;

bool \_isOutOfSize = true;

bool \_isOutOfPixel = true; //避免像素过大

\_isOutOfArea = zoom\_beginRow >= hv\_imageHeight || zoom\_endRow <= 0 || zoom\_beginCol >= hv\_imageWidth || zoom\_endCol < 0;

\_isOutOfSize = (zoom\_endRow - zoom\_beginRow) > hv\_imageHeight \* 20 || (zoom\_endCol - zoom\_beginCol) > hv\_imageWidth \* 20;

\_isOutOfPixel = hw\_height / (zoom\_endRow - zoom\_beginRow) > 500 || hw\_width / (zoom\_endCol - zoom\_beginCol) > 500;

if (\_isOutOfArea || \_isOutOfSize)

{

DispImageFit(t\_image, hw\_Ctrl);

}

else if (!\_isOutOfPixel)

{

hw\_Ctrl.HalconWindow.ClearWindow();

hw\_Ctrl.HalconWindow.SetPaint(new HTuple("default"));

// 保持图像显示比例

//hw\_Ctrl.HalconWindow.SetPart(zoom\_beginRow, zoom\_beginCol, zoom\_endRow, zoom\_endCol);

hw\_Ctrl.HalconWindow.SetPart(zoom\_beginRow, zoom\_beginCol, zoom\_endRow, zoom\_beginCol + (zoom\_endRow - zoom\_beginRow) \* hw\_width / hw\_height);

hw\_Ctrl.HalconWindow.DispObj(t\_image);

}

}

catch (Exception ex) //ex.Message;

{

DispImageFit(t\_image, hw\_Ctrl);

}

}

}

//最大化图像，适应窗体尺寸显示图像

public void DispImageFit(HObject t\_image, HWindowControl hw\_Ctrl)

{

if (t\_image != null)

{

hw\_Ctrl.HalconWindow.ClearWindow();

HTuple hWindowHandle = hw\_Ctrl.HalconWindow; //图像显示句柄

int hw\_width = hw\_Ctrl.WindowSize.Width; //图像显示尺寸

int hw\_height = hw\_Ctrl.WindowSize.Height;

HTuple width, height;

HOperatorSet.GetImageSize(t\_image, out width, out height);

if (1.0 \* width[0].I / hw\_width > 1.0 \* height[0].I / hw\_height)

{

double real = 1.0 \* width[0].I / hw\_width;

HOperatorSet.SetPart(hWindowHandle, 0, 0, real \* hw\_height, real \* hw\_width);

HOperatorSet.DispObj(t\_image, hWindowHandle);

}

else

{

double real = 1.0 \* height[0].I / hw\_height;

HOperatorSet.SetPart(hWindowHandle, 0, 0, real \* hw\_height, real \* hw\_width);

HOperatorSet.DispObj(t\_image, hWindowHandle);

}

}

}

// public HTuple hv\_ExpDefaultWinHandle;

// Chapter: Graphics / Text //hv\_WindowHandle

// Short Description: This procedure writes a text message.

public void disp\_message(HTuple hv\_ExpDefaultWinHandle, HTuple hv\_String, HTuple hv\_CoordSystem,

HTuple hv\_Row, HTuple hv\_Column, HTuple hv\_Color, HTuple hv\_Box)

{

// Local iconic variables

// Local control variables

HTuple hv\_M = null, hv\_N = null, hv\_Red = null;

HTuple hv\_Green = null, hv\_Blue = null, hv\_RowI1Part = null;

HTuple hv\_ColumnI1Part = null, hv\_RowI2Part = null, hv\_ColumnI2Part = null;

HTuple hv\_RowIWin = null, hv\_ColumnIWin = null, hv\_WidthWin = new HTuple();

HTuple hv\_HeightWin = null, hv\_I = null, hv\_RowI = new HTuple();

HTuple hv\_ColumnI = new HTuple(), hv\_StringI = new HTuple();

HTuple hv\_MaxAscent = new HTuple(), hv\_MaxDescent = new HTuple();

HTuple hv\_MaxWidth = new HTuple(), hv\_MaxHeight = new HTuple();

HTuple hv\_R1 = new HTuple(), hv\_C1 = new HTuple(), hv\_FactorRowI = new HTuple();

HTuple hv\_FactorColumnI = new HTuple(), hv\_UseShadow = new HTuple();

HTuple hv\_ShadowColor = new HTuple(), hv\_Exception = new HTuple();

HTuple hv\_Width = new HTuple(), hv\_Index = new HTuple();

HTuple hv\_Ascent = new HTuple(), hv\_Descent = new HTuple();

HTuple hv\_W = new HTuple(), hv\_H = new HTuple(), hv\_FrameHeight = new HTuple();

HTuple hv\_FrameWidth = new HTuple(), hv\_R2 = new HTuple();

HTuple hv\_C2 = new HTuple(), hv\_DrawMode = new HTuple();

HTuple hv\_CurrentColor = new HTuple();

HTuple hv\_Box\_COPY\_INP\_TMP = hv\_Box.Clone();

HTuple hv\_Color\_COPY\_INP\_TMP = hv\_Color.Clone();

HTuple hv\_Column\_COPY\_INP\_TMP = hv\_Column.Clone();

HTuple hv\_Row\_COPY\_INP\_TMP = hv\_Row.Clone();

HTuple hv\_String\_COPY\_INP\_TMP = hv\_String.Clone();

// Initialize local and output iconic variables

//This procedure displays text in a graphics window.

//

//Input parameters:

//WindowHandle: The WindowHandle of the graphics window, where

// the message should be displayed

//String: A tuple of strings containing the text message to be displayed

//CoordSystem: If set to 'window', the text position is given

// with respect to the window coordinate system.

// If set to 'image', image coordinates are used.

// (This may be useful in zoomed images.)

//Row: The row coordinate of the desired text position

// If set to -1, a default value of 12 is used.

// A tuple of values is allowed to display text at different

// positions.

//Column: The column coordinate of the desired text position

// If set to -1, a default value of 12 is used.

// A tuple of values is allowed to display text at different

// positions.

//Color: defines the color of the text as string.

// If set to [], '' or 'auto' the currently set color is used.

// If a tuple of strings is passed, the colors are used cyclically...

// - if |Row| == |Column| == 1: for each new textline

// = else for each text position.

//Box: If Box[0] is set to 'true', the text is written within an orange box.

// If set to' false', no box is displayed.

// If set to a color string (e.g. 'white', '#FF00CC', etc.),

// the text is written in a box of that color.

// An optional second value for Box (Box[1]) controls if a shadow is displayed:

// 'true' -> display a shadow in a default color

// 'false' -> display no shadow

// otherwise -> use given string as color string for the shadow color

//

//It is possible to display multiple text strings in a single call.

//In this case, some restrictions apply:

//- Multiple text positions can be defined by specifying a tuple

// with multiple Row and/or Column coordinates, i.e.:

// - |Row| == n, |Column| == n

// - |Row| == n, |Column| == 1

// - |Row| == 1, |Column| == n

//- If |Row| == |Column| == 1,

// each element of String is display in a new textline.

//- If multiple positions or specified, the number of Strings

// must match the number of positions, i.e.:

// - Either |String| == n (each string is displayed at the

// corresponding position),

// - or |String| == 1 (The string is displayed n times).

//

if ((int)(new HTuple(hv\_Color\_COPY\_INP\_TMP.TupleEqual(new HTuple()))) != 0)

{

hv\_Color\_COPY\_INP\_TMP = "";

}

if ((int)(new HTuple(hv\_Box\_COPY\_INP\_TMP.TupleEqual(new HTuple()))) != 0)

{

hv\_Box\_COPY\_INP\_TMP = "false";

}

//

//

//Check conditions

//

hv\_M = (new HTuple(hv\_Row\_COPY\_INP\_TMP.TupleLength())) \* (new HTuple(hv\_Column\_COPY\_INP\_TMP.TupleLength()

));

hv\_N = new HTuple(hv\_Row\_COPY\_INP\_TMP.TupleLength());

if ((int)((new HTuple(hv\_M.TupleEqual(0))).TupleOr(new HTuple(hv\_String\_COPY\_INP\_TMP.TupleEqual(

new HTuple())))) != 0)

{

return;

}

if ((int)(new HTuple(hv\_M.TupleNotEqual(1))) != 0)

{

//Multiple positions

//

//Expand single parameters

if ((int)(new HTuple((new HTuple(hv\_Row\_COPY\_INP\_TMP.TupleLength())).TupleEqual(

1))) != 0)

{

hv\_N = new HTuple(hv\_Column\_COPY\_INP\_TMP.TupleLength());

HOperatorSet.TupleGenConst(hv\_N, hv\_Row\_COPY\_INP\_TMP, out hv\_Row\_COPY\_INP\_TMP);

}

else if ((int)(new HTuple((new HTuple(hv\_Column\_COPY\_INP\_TMP.TupleLength()

)).TupleEqual(1))) != 0)

{

HOperatorSet.TupleGenConst(hv\_N, hv\_Column\_COPY\_INP\_TMP, out hv\_Column\_COPY\_INP\_TMP);

}

else if ((int)(new HTuple((new HTuple(hv\_Column\_COPY\_INP\_TMP.TupleLength()

)).TupleNotEqual(new HTuple(hv\_Row\_COPY\_INP\_TMP.TupleLength())))) != 0)

{

throw new HalconException("Number of elements in Row and Column does not match.");

}

if ((int)(new HTuple((new HTuple(hv\_String\_COPY\_INP\_TMP.TupleLength())).TupleEqual(

1))) != 0)

{

HOperatorSet.TupleGenConst(hv\_N, hv\_String\_COPY\_INP\_TMP, out hv\_String\_COPY\_INP\_TMP);

}

else if ((int)(new HTuple((new HTuple(hv\_String\_COPY\_INP\_TMP.TupleLength()

)).TupleNotEqual(hv\_N))) != 0)

{

throw new HalconException("Number of elements in Strings does not match number of positions.");

}

//

}

//

//Prepare window

HOperatorSet.GetRgb(hv\_ExpDefaultWinHandle, out hv\_Red, out hv\_Green, out hv\_Blue);

HOperatorSet.GetPart(hv\_ExpDefaultWinHandle, out hv\_RowI1Part, out hv\_ColumnI1Part,

out hv\_RowI2Part, out hv\_ColumnI2Part);

HOperatorSet.GetWindowExtents(hv\_ExpDefaultWinHandle, out hv\_RowIWin, out hv\_ColumnIWin,

out hv\_WidthWin, out hv\_HeightWin);

HOperatorSet.SetPart(hv\_ExpDefaultWinHandle, 0, 0, hv\_HeightWin - 1, hv\_WidthWin - 1);

//

//Loop over all positions

HTuple end\_val89 = hv\_N - 1;

HTuple step\_val89 = 1;

for (hv\_I = 0; hv\_I.Continue(end\_val89, step\_val89); hv\_I = hv\_I.TupleAdd(step\_val89))

{

hv\_RowI = hv\_Row\_COPY\_INP\_TMP.TupleSelect(hv\_I);

hv\_ColumnI = hv\_Column\_COPY\_INP\_TMP.TupleSelect(hv\_I);

//Allow multiple strings for a single position.

if ((int)(new HTuple(hv\_N.TupleEqual(1))) != 0)

{

hv\_StringI = hv\_String\_COPY\_INP\_TMP.Clone();

}

else

{

//In case of multiple positions, only single strings

//are allowed per position.

//For line breaks, use \n in this case.

hv\_StringI = hv\_String\_COPY\_INP\_TMP.TupleSelect(hv\_I);

}

//Default settings

//-1 is mapped to 12.

if ((int)(new HTuple(hv\_RowI.TupleEqual(-1))) != 0)

{

hv\_RowI = 12;

}

if ((int)(new HTuple(hv\_ColumnI.TupleEqual(-1))) != 0)

{

hv\_ColumnI = 12;

}

//

//Split string into one string per line.

hv\_StringI = ((("" + hv\_StringI) + "")).TupleSplit("\n");

//

//Estimate extentions of text depending on font size.

HOperatorSet.GetFontExtents(hv\_ExpDefaultWinHandle, out hv\_MaxAscent, out hv\_MaxDescent,

out hv\_MaxWidth, out hv\_MaxHeight);

if ((int)(new HTuple(hv\_CoordSystem.TupleEqual("window"))) != 0)

{

hv\_R1 = hv\_RowI.Clone();

hv\_C1 = hv\_ColumnI.Clone();

}

else

{

//Transform image to window coordinates.

hv\_FactorRowI = (1.0 \* hv\_HeightWin) / ((hv\_RowI2Part - hv\_RowI1Part) + 1);

hv\_FactorColumnI = (1.0 \* hv\_WidthWin) / ((hv\_ColumnI2Part - hv\_ColumnI1Part) + 1);

hv\_R1 = (((hv\_RowI - hv\_RowI1Part) + 0.5) \* hv\_FactorRowI) - 0.5;

hv\_C1 = (((hv\_ColumnI - hv\_ColumnI1Part) + 0.5) \* hv\_FactorColumnI) - 0.5;

}

//

//Display text box depending on text size.

hv\_UseShadow = 1;

hv\_ShadowColor = "gray";

if ((int)(new HTuple(((hv\_Box\_COPY\_INP\_TMP.TupleSelect(0))).TupleEqual("true"))) != 0)

{

if (hv\_Box\_COPY\_INP\_TMP == null)

hv\_Box\_COPY\_INP\_TMP = new HTuple();

hv\_Box\_COPY\_INP\_TMP[0] = "#fce9d4";

hv\_ShadowColor = "#f28d26";

}

if ((int)(new HTuple((new HTuple(hv\_Box\_COPY\_INP\_TMP.TupleLength())).TupleGreater(

1))) != 0)

{

if ((int)(new HTuple(((hv\_Box\_COPY\_INP\_TMP.TupleSelect(1))).TupleEqual("true"))) != 0)

{

//Use default ShadowColor set above

}

else if ((int)(new HTuple(((hv\_Box\_COPY\_INP\_TMP.TupleSelect(1))).TupleEqual(

"false"))) != 0)

{

hv\_UseShadow = 0;

}

else

{

hv\_ShadowColor = hv\_Box\_COPY\_INP\_TMP.TupleSelect(1);

//Valid color?

try

{

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, hv\_Box\_COPY\_INP\_TMP.TupleSelect(

1));

}

// catch (Exception)

catch (HalconException HDevExpDefaultException1)

{

HDevExpDefaultException1.ToHTuple(out hv\_Exception);

hv\_Exception = new HTuple("Wrong value of control parameter Box[1] (must be a 'true', 'false', or a valid color string)");

throw new HalconException(hv\_Exception);

}

}

}

if ((int)(new HTuple(((hv\_Box\_COPY\_INP\_TMP.TupleSelect(0))).TupleNotEqual("false"))) != 0)

{

//Valid color?

try

{

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, hv\_Box\_COPY\_INP\_TMP.TupleSelect(

0));

}

// catch (Exception)

catch (HalconException HDevExpDefaultException1)

{

HDevExpDefaultException1.ToHTuple(out hv\_Exception);

hv\_Exception = new HTuple("Wrong value of control parameter Box[0] (must be a 'true', 'false', or a valid color string)");

throw new HalconException(hv\_Exception);

}

//Calculate box extents

hv\_StringI = (" " + hv\_StringI) + " ";

hv\_Width = new HTuple();

for (hv\_Index = 0; (int)hv\_Index <= (int)((new HTuple(hv\_StringI.TupleLength()

)) - 1); hv\_Index = (int)hv\_Index + 1)

{

HOperatorSet.GetStringExtents(hv\_ExpDefaultWinHandle, hv\_StringI.TupleSelect(

hv\_Index), out hv\_Ascent, out hv\_Descent, out hv\_W, out hv\_H);

hv\_Width = hv\_Width.TupleConcat(hv\_W);

}

hv\_FrameHeight = hv\_MaxHeight \* (new HTuple(hv\_StringI.TupleLength()));

hv\_FrameWidth = (((new HTuple(0)).TupleConcat(hv\_Width))).TupleMax();

hv\_R2 = hv\_R1 + hv\_FrameHeight;

hv\_C2 = hv\_C1 + hv\_FrameWidth;

//Display rectangles

HOperatorSet.GetDraw(hv\_ExpDefaultWinHandle, out hv\_DrawMode);

HOperatorSet.SetDraw(hv\_ExpDefaultWinHandle, "fill");

//Set shadow color

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, hv\_ShadowColor);

if ((int)(hv\_UseShadow) != 0)

{

HOperatorSet.DispRectangle1(hv\_ExpDefaultWinHandle, hv\_R1 + 1, hv\_C1 + 1, hv\_R2 + 1,

hv\_C2 + 1);

}

//Set box color

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, hv\_Box\_COPY\_INP\_TMP.TupleSelect(

0));

HOperatorSet.DispRectangle1(hv\_ExpDefaultWinHandle, hv\_R1, hv\_C1, hv\_R2,

hv\_C2);

HOperatorSet.SetDraw(hv\_ExpDefaultWinHandle, hv\_DrawMode);

}

//Write text.

for (hv\_Index = 0; (int)hv\_Index <= (int)((new HTuple(hv\_StringI.TupleLength())) - 1); hv\_Index = (int)hv\_Index + 1)

{

//Set color

if ((int)(new HTuple(hv\_N.TupleEqual(1))) != 0)

{

//Wiht a single text position, each text line

//may get a different color.

hv\_CurrentColor = hv\_Color\_COPY\_INP\_TMP.TupleSelect(hv\_Index % (new HTuple(hv\_Color\_COPY\_INP\_TMP.TupleLength()

)));

}

else

{

//With multiple text positions, each position

//gets a single color for all text lines.

hv\_CurrentColor = hv\_Color\_COPY\_INP\_TMP.TupleSelect(hv\_I % (new HTuple(hv\_Color\_COPY\_INP\_TMP.TupleLength()

)));

}

if ((int)((new HTuple(hv\_CurrentColor.TupleNotEqual(""))).TupleAnd(new HTuple(hv\_CurrentColor.TupleNotEqual(

"auto")))) != 0)

{

//Valid color?

try

{

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, hv\_CurrentColor);

}

// catch (Exception)

catch (HalconException HDevExpDefaultException1)

{

HDevExpDefaultException1.ToHTuple(out hv\_Exception);

hv\_Exception = ((("Wrong value of control parameter Color[" + (hv\_Index % (new HTuple(hv\_Color\_COPY\_INP\_TMP.TupleLength()

)))) + "] == '") + hv\_CurrentColor) + "' (must be a valid color string)";

throw new HalconException(hv\_Exception);

}

}

else

{

HOperatorSet.SetRgb(hv\_ExpDefaultWinHandle, hv\_Red, hv\_Green, hv\_Blue);

}

//Finally display text

hv\_RowI = hv\_R1 + (hv\_MaxHeight \* hv\_Index);

HOperatorSet.SetTposition(hv\_ExpDefaultWinHandle, hv\_RowI, hv\_C1);

HOperatorSet.WriteString(hv\_ExpDefaultWinHandle, hv\_StringI.TupleSelect(hv\_Index));

}

}

//Reset changed window settings

HOperatorSet.SetRgb(hv\_ExpDefaultWinHandle, hv\_Red, hv\_Green, hv\_Blue);

HOperatorSet.SetPart(hv\_ExpDefaultWinHandle, hv\_RowI1Part, hv\_ColumnI1Part, hv\_RowI2Part,

hv\_ColumnI2Part);

return;

}

// Chapter: Graphics / Text

// Short Description: Set font independent of OS

public void set\_display\_font(HTuple hv\_ExpDefaultWinHandle, HTuple hv\_Size, HTuple hv\_Font,

HTuple hv\_Bold, HTuple hv\_Slant)

{

// Local iconic variables

// Local control variables

HTuple hv\_OS = null, hv\_BufferWindowHandle = new HTuple();

HTuple hv\_Ascent = new HTuple(), hv\_Descent = new HTuple();

HTuple hv\_Width = new HTuple(), hv\_Height = new HTuple();

HTuple hv\_Scale = new HTuple(), hv\_Exception = new HTuple();

HTuple hv\_SubFamily = new HTuple(), hv\_Fonts = new HTuple();

HTuple hv\_SystemFonts = new HTuple(), hv\_Guess = new HTuple();

HTuple hv\_I = new HTuple(), hv\_Index = new HTuple(), hv\_AllowedFontSizes = new HTuple();

HTuple hv\_Distances = new HTuple(), hv\_Indices = new HTuple();

HTuple hv\_FontSelRegexp = new HTuple(), hv\_FontsCourier = new HTuple();

HTuple hv\_Bold\_COPY\_INP\_TMP = hv\_Bold.Clone();

HTuple hv\_Font\_COPY\_INP\_TMP = hv\_Font.Clone();

HTuple hv\_Size\_COPY\_INP\_TMP = hv\_Size.Clone();

HTuple hv\_Slant\_COPY\_INP\_TMP = hv\_Slant.Clone();

// Initialize local and output iconic variables

//This procedure sets the text font of the current window with

//the specified attributes.

//It is assumed that following fonts are installed on the system:

//Windows: Courier New, Arial Times New Roman

//Mac OS X: CourierNewPS, Arial, TimesNewRomanPS

//Linux: courier, helvetica, times

//Because fonts are displayed smaller on Linux than on Windows,

//a scaling factor of 1.25 is used the get comparable results.

//For Linux, only a limited number of font sizes is supported,

//to get comparable results, it is recommended to use one of the

//following sizes: 9, 11, 14, 16, 20, 27

//(which will be mapped internally on Linux systems to 11, 14, 17, 20, 25, 34)

//

//Input parameters:

//WindowHandle: The graphics window for which the font will be set

//Size: The font size. If Size=-1, the default of 16 is used.

//Bold: If set to 'true', a bold font is used

//Slant: If set to 'true', a slanted font is used

//

HOperatorSet.GetSystem("operating\_system", out hv\_OS);

// dev\_get\_preferences(...); only in hdevelop

// dev\_set\_preferences(...); only in hdevelop

if ((int)((new HTuple(hv\_Size\_COPY\_INP\_TMP.TupleEqual(new HTuple()))).TupleOr(

new HTuple(hv\_Size\_COPY\_INP\_TMP.TupleEqual(-1)))) != 0)

{

hv\_Size\_COPY\_INP\_TMP = 16;

}

if ((int)(new HTuple(((hv\_OS.TupleSubstr(0, 2))).TupleEqual("Win"))) != 0)

{

//Set font on Windows systems

try

{

//Check, if font scaling is switched on

//open\_window(...);

HOperatorSet.SetFont(hv\_ExpDefaultWinHandle, "-Consolas-16-\*-0-\*-\*-1-");

HOperatorSet.GetStringExtents(hv\_ExpDefaultWinHandle, "test\_string", out hv\_Ascent,

out hv\_Descent, out hv\_Width, out hv\_Height);

//Expected width is 110

hv\_Scale = 110.0 / hv\_Width;

hv\_Size\_COPY\_INP\_TMP = ((hv\_Size\_COPY\_INP\_TMP \* hv\_Scale)).TupleInt();

//close\_window(...);

}

// catch (Exception)

catch (HalconException HDevExpDefaultException1)

{

HDevExpDefaultException1.ToHTuple(out hv\_Exception);

//throw (Exception)

}

if ((int)((new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("Courier"))).TupleOr(

new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("courier")))) != 0)

{

hv\_Font\_COPY\_INP\_TMP = "Courier New";

}

else if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("mono"))) != 0)

{

hv\_Font\_COPY\_INP\_TMP = "Consolas";

}

else if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("sans"))) != 0)

{

hv\_Font\_COPY\_INP\_TMP = "Arial";

}

else if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("serif"))) != 0)

{

hv\_Font\_COPY\_INP\_TMP = "Times New Roman";

}

if ((int)(new HTuple(hv\_Bold\_COPY\_INP\_TMP.TupleEqual("true"))) != 0)

{

hv\_Bold\_COPY\_INP\_TMP = 1;

}

else if ((int)(new HTuple(hv\_Bold\_COPY\_INP\_TMP.TupleEqual("false"))) != 0)

{

hv\_Bold\_COPY\_INP\_TMP = 0;

}

else

{

hv\_Exception = "Wrong value of control parameter Bold";

throw new HalconException(hv\_Exception);

}

if ((int)(new HTuple(hv\_Slant\_COPY\_INP\_TMP.TupleEqual("true"))) != 0)

{

hv\_Slant\_COPY\_INP\_TMP = 1;

}

else if ((int)(new HTuple(hv\_Slant\_COPY\_INP\_TMP.TupleEqual("false"))) != 0)

{

hv\_Slant\_COPY\_INP\_TMP = 0;

}

else

{

hv\_Exception = "Wrong value of control parameter Slant";

throw new HalconException(hv\_Exception);

}

try

{

HOperatorSet.SetFont(hv\_ExpDefaultWinHandle, ((((((("-" + hv\_Font\_COPY\_INP\_TMP) + "-") + hv\_Size\_COPY\_INP\_TMP) + "-\*-") + hv\_Slant\_COPY\_INP\_TMP) + "-\*-\*-") + hv\_Bold\_COPY\_INP\_TMP) + "-");

}

// catch (Exception)

catch (HalconException HDevExpDefaultException1)

{

HDevExpDefaultException1.ToHTuple(out hv\_Exception);

//throw (Exception)

}

}

else if ((int)(new HTuple(((hv\_OS.TupleSubstr(0, 2))).TupleEqual("Dar"))) != 0)

{

//Set font on Mac OS X systems. Since OS X does not have a strict naming

//scheme for font attributes, we use tables to determine the correct font

//name.

hv\_SubFamily = 0;

if ((int)(new HTuple(hv\_Slant\_COPY\_INP\_TMP.TupleEqual("true"))) != 0)

{

hv\_SubFamily = hv\_SubFamily.TupleBor(1);

}

else if ((int)(new HTuple(hv\_Slant\_COPY\_INP\_TMP.TupleNotEqual("false"))) != 0)

{

hv\_Exception = "Wrong value of control parameter Slant";

throw new HalconException(hv\_Exception);

}

if ((int)(new HTuple(hv\_Bold\_COPY\_INP\_TMP.TupleEqual("true"))) != 0)

{

hv\_SubFamily = hv\_SubFamily.TupleBor(2);

}

else if ((int)(new HTuple(hv\_Bold\_COPY\_INP\_TMP.TupleNotEqual("false"))) != 0)

{

hv\_Exception = "Wrong value of control parameter Bold";

throw new HalconException(hv\_Exception);

}

if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("mono"))) != 0)

{

hv\_Fonts = new HTuple();

hv\_Fonts[0] = "Menlo-Regular";

hv\_Fonts[1] = "Menlo-Italic";

hv\_Fonts[2] = "Menlo-Bold";

hv\_Fonts[3] = "Menlo-BoldItalic";

}

else if ((int)((new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("Courier"))).TupleOr(

new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("courier")))) != 0)

{

hv\_Fonts = new HTuple();

hv\_Fonts[0] = "CourierNewPSMT";

hv\_Fonts[1] = "CourierNewPS-ItalicMT";

hv\_Fonts[2] = "CourierNewPS-BoldMT";

hv\_Fonts[3] = "CourierNewPS-BoldItalicMT";

}

else if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("sans"))) != 0)

{

hv\_Fonts = new HTuple();

hv\_Fonts[0] = "ArialMT";

hv\_Fonts[1] = "Arial-ItalicMT";

hv\_Fonts[2] = "Arial-BoldMT";

hv\_Fonts[3] = "Arial-BoldItalicMT";

}

else if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("serif"))) != 0)

{

hv\_Fonts = new HTuple();

hv\_Fonts[0] = "TimesNewRomanPSMT";

hv\_Fonts[1] = "TimesNewRomanPS-ItalicMT";

hv\_Fonts[2] = "TimesNewRomanPS-BoldMT";

hv\_Fonts[3] = "TimesNewRomanPS-BoldItalicMT";

}

else

{

//Attempt to figure out which of the fonts installed on the system

//the user could have meant.

HOperatorSet.QueryFont(hv\_ExpDefaultWinHandle, out hv\_SystemFonts);

hv\_Fonts = new HTuple();

hv\_Fonts = hv\_Fonts.TupleConcat(hv\_Font\_COPY\_INP\_TMP);

hv\_Fonts = hv\_Fonts.TupleConcat(hv\_Font\_COPY\_INP\_TMP);

hv\_Fonts = hv\_Fonts.TupleConcat(hv\_Font\_COPY\_INP\_TMP);

hv\_Fonts = hv\_Fonts.TupleConcat(hv\_Font\_COPY\_INP\_TMP);

hv\_Guess = new HTuple();

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP);

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-Regular");

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "MT");

for (hv\_I = 0; (int)hv\_I <= (int)((new HTuple(hv\_Guess.TupleLength())) - 1); hv\_I = (int)hv\_I + 1)

{

HOperatorSet.TupleFind(hv\_SystemFonts, hv\_Guess.TupleSelect(hv\_I), out hv\_Index);

if ((int)(new HTuple(hv\_Index.TupleNotEqual(-1))) != 0)

{

if (hv\_Fonts == null)

hv\_Fonts = new HTuple();

hv\_Fonts[0] = hv\_Guess.TupleSelect(hv\_I);

break;

}

}

//Guess name of slanted font

hv\_Guess = new HTuple();

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-Italic");

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-ItalicMT");

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-Oblique");

for (hv\_I = 0; (int)hv\_I <= (int)((new HTuple(hv\_Guess.TupleLength())) - 1); hv\_I = (int)hv\_I + 1)

{

HOperatorSet.TupleFind(hv\_SystemFonts, hv\_Guess.TupleSelect(hv\_I), out hv\_Index);

if ((int)(new HTuple(hv\_Index.TupleNotEqual(-1))) != 0)

{

if (hv\_Fonts == null)

hv\_Fonts = new HTuple();

hv\_Fonts[1] = hv\_Guess.TupleSelect(hv\_I);

break;

}

}

//Guess name of bold font

hv\_Guess = new HTuple();

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-Bold");

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-BoldMT");

for (hv\_I = 0; (int)hv\_I <= (int)((new HTuple(hv\_Guess.TupleLength())) - 1); hv\_I = (int)hv\_I + 1)

{

HOperatorSet.TupleFind(hv\_SystemFonts, hv\_Guess.TupleSelect(hv\_I), out hv\_Index);

if ((int)(new HTuple(hv\_Index.TupleNotEqual(-1))) != 0)

{

if (hv\_Fonts == null)

hv\_Fonts = new HTuple();

hv\_Fonts[2] = hv\_Guess.TupleSelect(hv\_I);

break;

}

}

//Guess name of bold slanted font

hv\_Guess = new HTuple();

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-BoldItalic");

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-BoldItalicMT");

hv\_Guess = hv\_Guess.TupleConcat(hv\_Font\_COPY\_INP\_TMP + "-BoldOblique");

for (hv\_I = 0; (int)hv\_I <= (int)((new HTuple(hv\_Guess.TupleLength())) - 1); hv\_I = (int)hv\_I + 1)

{

HOperatorSet.TupleFind(hv\_SystemFonts, hv\_Guess.TupleSelect(hv\_I), out hv\_Index);

if ((int)(new HTuple(hv\_Index.TupleNotEqual(-1))) != 0)

{

if (hv\_Fonts == null)

hv\_Fonts = new HTuple();

hv\_Fonts[3] = hv\_Guess.TupleSelect(hv\_I);

break;

}

}

}

hv\_Font\_COPY\_INP\_TMP = hv\_Fonts.TupleSelect(hv\_SubFamily);

try

{

HOperatorSet.SetFont(hv\_ExpDefaultWinHandle, (hv\_Font\_COPY\_INP\_TMP + "-") + hv\_Size\_COPY\_INP\_TMP);

}

// catch (Exception)

catch (HalconException HDevExpDefaultException1)

{

HDevExpDefaultException1.ToHTuple(out hv\_Exception);

//throw (Exception)

}

}

else

{

//Set font for UNIX systems

hv\_Size\_COPY\_INP\_TMP = hv\_Size\_COPY\_INP\_TMP \* 1.25;

hv\_AllowedFontSizes = new HTuple();

hv\_AllowedFontSizes[0] = 11;

hv\_AllowedFontSizes[1] = 14;

hv\_AllowedFontSizes[2] = 17;

hv\_AllowedFontSizes[3] = 20;

hv\_AllowedFontSizes[4] = 25;

hv\_AllowedFontSizes[5] = 34;

if ((int)(new HTuple(((hv\_AllowedFontSizes.TupleFind(hv\_Size\_COPY\_INP\_TMP))).TupleEqual(

-1))) != 0)

{

hv\_Distances = ((hv\_AllowedFontSizes - hv\_Size\_COPY\_INP\_TMP)).TupleAbs();

HOperatorSet.TupleSortIndex(hv\_Distances, out hv\_Indices);

hv\_Size\_COPY\_INP\_TMP = hv\_AllowedFontSizes.TupleSelect(hv\_Indices.TupleSelect(

0));

}

if ((int)((new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("mono"))).TupleOr(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual(

"Courier")))) != 0)

{

hv\_Font\_COPY\_INP\_TMP = "courier";

}

else if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("sans"))) != 0)

{

hv\_Font\_COPY\_INP\_TMP = "helvetica";

}

else if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("serif"))) != 0)

{

hv\_Font\_COPY\_INP\_TMP = "times";

}

if ((int)(new HTuple(hv\_Bold\_COPY\_INP\_TMP.TupleEqual("true"))) != 0)

{

hv\_Bold\_COPY\_INP\_TMP = "bold";

}

else if ((int)(new HTuple(hv\_Bold\_COPY\_INP\_TMP.TupleEqual("false"))) != 0)

{

hv\_Bold\_COPY\_INP\_TMP = "medium";

}

else

{

hv\_Exception = "Wrong value of control parameter Bold";

throw new HalconException(hv\_Exception);

}

if ((int)(new HTuple(hv\_Slant\_COPY\_INP\_TMP.TupleEqual("true"))) != 0)

{

if ((int)(new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("times"))) != 0)

{

hv\_Slant\_COPY\_INP\_TMP = "i";

}

else

{

hv\_Slant\_COPY\_INP\_TMP = "o";

}

}

else if ((int)(new HTuple(hv\_Slant\_COPY\_INP\_TMP.TupleEqual("false"))) != 0)

{

hv\_Slant\_COPY\_INP\_TMP = "r";

}

else

{

hv\_Exception = "Wrong value of control parameter Slant";

throw new HalconException(hv\_Exception);

}

try

{

HOperatorSet.SetFont(hv\_ExpDefaultWinHandle, ((((((("-adobe-" + hv\_Font\_COPY\_INP\_TMP) + "-") + hv\_Bold\_COPY\_INP\_TMP) + "-") + hv\_Slant\_COPY\_INP\_TMP) + "-normal-\*-") + hv\_Size\_COPY\_INP\_TMP) + "-\*-\*-\*-\*-\*-\*-\*");

}

// catch (Exception)

catch (HalconException HDevExpDefaultException1)

{

HDevExpDefaultException1.ToHTuple(out hv\_Exception);

if ((int)((new HTuple(((hv\_OS.TupleSubstr(0, 4))).TupleEqual("Linux"))).TupleAnd(

new HTuple(hv\_Font\_COPY\_INP\_TMP.TupleEqual("courier")))) != 0)

{

HOperatorSet.QueryFont(hv\_ExpDefaultWinHandle, out hv\_Fonts);

hv\_FontSelRegexp = (("^-[^-]\*-[^-]\*[Cc]ourier[^-]\*-" + hv\_Bold\_COPY\_INP\_TMP) + "-") + hv\_Slant\_COPY\_INP\_TMP;

hv\_FontsCourier = ((hv\_Fonts.TupleRegexpSelect(hv\_FontSelRegexp))).TupleRegexpMatch(

hv\_FontSelRegexp);

if ((int)(new HTuple((new HTuple(hv\_FontsCourier.TupleLength())).TupleEqual(

0))) != 0)

{

hv\_Exception = "Wrong font name";

//throw (Exception)

}

else

{

try

{

HOperatorSet.SetFont(hv\_ExpDefaultWinHandle, (((hv\_FontsCourier.TupleSelect(

0)) + "-normal-\*-") + hv\_Size\_COPY\_INP\_TMP) + "-\*-\*-\*-\*-\*-\*-\*");

}

// catch (Exception)

catch (HalconException HDevExpDefaultException2)

{

HDevExpDefaultException2.ToHTuple(out hv\_Exception);

//throw (Exception)

}

}

}

//throw (Exception)

}

}

// dev\_set\_preferences(...); only in hdevelop

return;

}

}

}

# HALCON\_AUTO.Designer.cs

namespace Halcon\_region\_sort

{

partial class halcon\_form

{

HDevelopExport hd = new HDevelopExport();

//string imagePayh;

/// <summary>

/// 必需的设计器变量。

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// 清理所有正在使用的资源。

/// </summary>

/// <param name="disposing">如果应释放托管资源，为 true；否则为 false。</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows 窗体设计器生成的代码

/// <summary>

/// 设计器支持所需的方法 - 不要

/// 使用代码编辑器修改此方法的内容。

/// </summary>

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(halcon\_form));

this.hWindowControl1 = new HalconDotNet.HWindowControl();

this.read\_button = new System.Windows.Forms.Button();

this.button\_prcoessing = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// hWindowControl1

//

this.hWindowControl1.AllowDrop = true;

this.hWindowControl1.BackColor = System.Drawing.Color.Black;

this.hWindowControl1.BorderColor = System.Drawing.Color.Black;

this.hWindowControl1.ImagePart = new System.Drawing.Rectangle(-15, -20, 2590, 1960);

this.hWindowControl1.Location = new System.Drawing.Point(158, 12);

this.hWindowControl1.Name = "hWindowControl1";

this.hWindowControl1.Size = new System.Drawing.Size(1095, 739);

this.hWindowControl1.TabIndex = 2;

this.hWindowControl1.WindowSize = new System.Drawing.Size(1095, 739);

this.hWindowControl1.HMouseMove += new HalconDotNet.HMouseEventHandler(this.hWindowControl1\_HMouseMove);

//

// read\_button

//

this.read\_button.BackColor = System.Drawing.SystemColors.AppWorkspace;

this.read\_button.Font = new System.Drawing.Font("隶书", 42F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.read\_button.Location = new System.Drawing.Point(12, 385);

this.read\_button.Name = "read\_button";

this.read\_button.Size = new System.Drawing.Size(130, 367);

this.read\_button.TabIndex = 0;

this.read\_button.Text = "图像处理";

this.read\_button.UseVisualStyleBackColor = false;

this.read\_button.Click += new System.EventHandler(this.read\_button\_Click);

//

// button\_prcoessing

//

this.button\_prcoessing.BackColor = System.Drawing.SystemColors.ButtonShadow;

this.button\_prcoessing.Font = new System.Drawing.Font("隶书", 42F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.button\_prcoessing.Location = new System.Drawing.Point(12, 12);

this.button\_prcoessing.Name = "button\_prcoessing";

this.button\_prcoessing.Size = new System.Drawing.Size(130, 367);

this.button\_prcoessing.TabIndex = 1;

this.button\_prcoessing.Text = "读取图片";

this.button\_prcoessing.UseVisualStyleBackColor = false;

this.button\_prcoessing.Click += new System.EventHandler(this.button\_prcoessing\_Click);

//

// halcon\_form

//

this.AllowDrop = true;

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.Black;

this.ClientSize = new System.Drawing.Size(1278, 751);

this.Controls.Add(this.hWindowControl1);

this.Controls.Add(this.button\_prcoessing);

this.Controls.Add(this.read\_button);

this.Cursor = System.Windows.Forms.Cursors.Default;

this.DoubleBuffered = true;

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;

this.HelpButton = true;

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.MaximizeBox = false;

this.Name = "halcon\_form";

this.Text = "自动处理";

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.halcon\_form\_FormClosing);

this.Load += new System.EventHandler(this.halcon\_form\_Load);

this.Click += new System.EventHandler(this.halcon\_form\_Click);

this.ResumeLayout(false);

}

#endregion

private HalconDotNet.HWindowControl hWindowControl1;

private System.Windows.Forms.Button read\_button;

private System.Windows.Forms.Button button\_prcoessing;

}

}

# HALCON\_AUTO.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using HalconDotNet; //引用halcon

using System.IO;

namespace Halcon\_region\_sort

{

public partial class halcon\_form : Form

{

public halcon\_form()

{

InitializeComponent();

}

//private HalconWindow wch = new HalconWindow();

public HTuple hv\_ExpDefaultWinHandle;

// Local iconic variables 定义图像变量

public HObject ho\_Image, ho\_GrayImage, ho\_Regions;

public HObject ho\_Connection, ho\_RegionResult, ho\_SelectedRegions;

public HObject ho\_RegionFillUp, ho\_RegionDilation, ho\_RegionUnion;

public HObject ho\_ImageReduced, ho\_Regions1, ho\_RegionErosion;

public HObject ho\_ConnectedRegions, ho\_SelectedRegions1, ho\_RegionDilation1;

public HObject ho\_SortedRegions, ho\_ObjectSelected\_1 = null;

// Local control variables 定义控制变量

public HTuple hv\_Width = null, hv\_Height = null, hv\_WindowHandle = new HTuple();

public HTuple hv\_UsedThreshold = null, hv\_UsedThreshold1 = null;

public HTuple hv\_Number = null, hv\_Area = null, hv\_Row = null;

public HTuple hv\_Column = null, hv\_i = null, hv\_Area1 = null;

public HTuple hv\_Row1 = null, hv\_Column1 = null, hv\_Area2 = null;

public HTuple hv\_area = null, hv\_bizhong = null;

// 主窗体加载事件

private void halcon\_form\_Load(object sender, EventArgs e)

{

HOperatorSet.GenEmptyObj(out ho\_Image);

HOperatorSet.GenEmptyObj(out ho\_GrayImage);

HOperatorSet.GenEmptyObj(out ho\_Regions);

HOperatorSet.GenEmptyObj(out ho\_Connection);

HOperatorSet.GenEmptyObj(out ho\_RegionResult);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions);

HOperatorSet.GenEmptyObj(out ho\_RegionFillUp);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation);

HOperatorSet.GenEmptyObj(out ho\_RegionUnion);

HOperatorSet.GenEmptyObj(out ho\_ImageReduced);

HOperatorSet.GenEmptyObj(out ho\_Regions1);

HOperatorSet.GenEmptyObj(out ho\_RegionErosion);

HOperatorSet.GenEmptyObj(out ho\_ConnectedRegions);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions1);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation1);

HOperatorSet.GenEmptyObj(out ho\_SortedRegions);

HOperatorSet.GenEmptyObj(out ho\_ObjectSelected\_1);

hv\_ExpDefaultWinHandle = hWindowControl1.HalconWindow;

}

//

private void read\_button\_Click(object sender, EventArgs e)

{

//程序开始

HOperatorSet.SetDraw(hv\_ExpDefaultWinHandle, "fill");

HOperatorSet.SetColored(hv\_ExpDefaultWinHandle, 12);

//读取图片

//ho\_Image.Dispose();

//HOperatorSet.ReadImage(out ho\_Image, "E:/WORKS/halcon/zhao\_jobs/zhao\_jobs/1-500-2.JPG");

//dev\_close\_window(...);

HOperatorSet.GetImageSize(ho\_Image, out hv\_Width, out hv\_Height);

//获取图像窗口

//dev\_open\_window(...);

//彩色图转成灰度图

ho\_GrayImage.Dispose();

HOperatorSet.Rgb1ToGray(ho\_Image, out ho\_GrayImage);

//阈值处理

ho\_Regions.Dispose();

HOperatorSet.BinaryThreshold(ho\_GrayImage, out ho\_Regions, "max\_separability",

"dark", out hv\_UsedThreshold);

//求图像联通域

ho\_Connection.Dispose();

HOperatorSet.Connection(ho\_Regions, out ho\_Connection);

//将区域闭合

ho\_RegionResult.Dispose();

HOperatorSet.CloseEdges(ho\_Connection, ho\_GrayImage, out ho\_RegionResult, 16);

//选择合适的区域

ho\_SelectedRegions.Dispose();

HOperatorSet.SelectShape(ho\_RegionResult, out ho\_SelectedRegions, "area", "and", 2040.82, 200000);

//填充区域孔洞

ho\_RegionFillUp.Dispose();

HOperatorSet.FillUp(ho\_SelectedRegions, out ho\_RegionFillUp);

//区域膨胀处理

ho\_RegionDilation.Dispose();

HOperatorSet.DilationCircle(ho\_RegionFillUp, out ho\_RegionDilation, 15.5);

//将区域群合成一个区域

ho\_RegionUnion.Dispose();

HOperatorSet.Union2(ho\_RegionDilation, ho\_RegionDilation, out ho\_RegionUnion);

// 获取刚兴趣区域图像

ho\_ImageReduced.Dispose();

HOperatorSet.ReduceDomain(ho\_GrayImage, ho\_RegionUnion, out ho\_ImageReduced);

//对感兴趣区域图像进行阈值处理

ho\_Regions1.Dispose();

HOperatorSet.BinaryThreshold(ho\_ImageReduced, out ho\_Regions1, "max\_separability", "dark", out hv\_UsedThreshold1);

//区域腐蚀处理，用于去除区域之间的粘结

ho\_RegionErosion.Dispose();

HOperatorSet.ErosionCircle(ho\_Regions1, out ho\_RegionErosion, 1.5);

//求图像连通域

ho\_ConnectedRegions.Dispose();

HOperatorSet.Connection(ho\_RegionErosion, out ho\_ConnectedRegions);

//选择合适的颗粒，去除无用的杂物

ho\_SelectedRegions1.Dispose();

HOperatorSet.SelectShape(ho\_ConnectedRegions, out ho\_SelectedRegions1, "area", "and", 2000, 9999999);

//对颗粒进行膨胀处理

ho\_RegionDilation1.Dispose();

HOperatorSet.DilationCircle(ho\_SelectedRegions1, out ho\_RegionDilation1, 10.5);

//数颗粒的个数

HOperatorSet.CountObj(ho\_RegionDilation1, out hv\_Number);

//对颗粒按照第一个点的行坐标进行排序

ho\_SortedRegions.Dispose();

HOperatorSet.SortRegion(ho\_RegionDilation1, out ho\_SortedRegions, "first\_point", "false", "row");

//求颗粒的面积和中心坐标

HOperatorSet.AreaCenter(ho\_SortedRegions, out hv\_Area, out hv\_Row, out hv\_Column);

//显示图像和选中的颗粒

HOperatorSet.DispObj(ho\_Image, hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_SortedRegions, hv\_ExpDefaultWinHandle);

//显示图像的代码

HTuple end\_val47 = hv\_Number;

HTuple step\_val47 = 1;

for (hv\_i = 1; hv\_i.Continue(end\_val47, step\_val47); hv\_i = hv\_i.TupleAdd(step\_val47))

{

ho\_ObjectSelected\_1.Dispose();

HOperatorSet.SelectObj(ho\_SortedRegions, out ho\_ObjectSelected\_1, hv\_i);

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, "red");

HOperatorSet.DispObj(ho\_ObjectSelected\_1, hv\_ExpDefaultWinHandle);

HOperatorSet.SetTposition(hv\_ExpDefaultWinHandle, hv\_Row.TupleSelect(hv\_i - 1),

hv\_Column.TupleSelect(hv\_i - 1));

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, "white");

HOperatorSet.WriteString(hv\_ExpDefaultWinHandle, hv\_i);

//write\_string (WindowHandle, Area[i-1])

}

//求面积占比

HOperatorSet.AreaCenter(ho\_Regions, out hv\_Area1, out hv\_Row1, out hv\_Column1);

hv\_Area2 = hv\_Area1.TupleReal();

hv\_area = ((hv\_Width \* hv\_Height)).TupleReal();

hv\_bizhong = hv\_Area2 / hv\_area;

//decompose3 (Image, Image1, Image2, Image3)

ho\_Image.Dispose();

ho\_GrayImage.Dispose();

ho\_Regions.Dispose();

ho\_Connection.Dispose();

ho\_RegionResult.Dispose();

ho\_SelectedRegions.Dispose();

ho\_RegionFillUp.Dispose();

ho\_RegionDilation.Dispose();

ho\_RegionUnion.Dispose();

ho\_ImageReduced.Dispose();

ho\_Regions1.Dispose();

ho\_RegionErosion.Dispose();

ho\_ConnectedRegions.Dispose();

ho\_SelectedRegions1.Dispose();

ho\_RegionDilation1.Dispose();

ho\_SortedRegions.Dispose();

ho\_ObjectSelected\_1.Dispose();

MessageBox.Show("占图比重为：" + hv\_bizhong + "\n颗粒个数：" + hv\_Number + "\n圆形颗粒个数：" + hv\_ro + "\n条形颗粒个数" + hv\_ti + "\n片形颗粒个数" + hv\_pi);

}

private void button\_prcoessing\_Click(object sender, EventArgs e)

{

hv\_ExpDefaultWinHandle = hWindowControl1.HalconWindow;

OpenFileDialog openFile = new OpenFileDialog();

openFile.Filter = "JPG文件(\*.jpg)|\*.jpg|所有文件(\*.\*)|\*.\*||";

if (openFile.ShowDialog() == DialogResult.OK)

{

HOperatorSet.ClearWindow(hv\_ExpDefaultWinHandle); //清空显示

string Path = openFile.FileName;

HOperatorSet.ReadImage(out ho\_Image, Path);

HOperatorSet.DispObj(ho\_Image, hv\_ExpDefaultWinHandle);

//wch.DispImageFit(ho\_Image, hv\_ExpDefaultWinHandle);

HTuple hv\_HeightWin, hv\_WidthWin;

HOperatorSet.GetImageSize(ho\_Image, out hv\_HeightWin, out hv\_WidthWin); // 获取输入图像的尺寸

String str\_imgSize = String.Format("Size:{0}x{1}", hv\_HeightWin, hv\_WidthWin);

//wch.disp\_message(hv\_ExpDefaultWinHandle, str\_imgSize, "window", hv\_ExpDefaultWinHandle.Height - 20, 1, "blue", "false");

}

}

private void hWindowControl1\_HMouseMove(object sender, HMouseEventArgs e)

{

}

private void halcon\_form\_Click(object sender, EventArgs e)

{

MessageBox.Show("程序属性\n铁谱处理2");

}

private void halcon\_form\_FormClosing(object sender, FormClosingEventArgs e)

{

HALCON\_SORT.form0.Show();

}

}

public partial class HDevelopExport

{

public HTuple hv\_ExpDefaultWinHandle;

public double bizhong;

// Local iconic variables 定义图像变量

public HObject ho\_Image, ho\_GrayImage, ho\_Regions;

HObject ho\_Connection, ho\_RegionResult, ho\_SelectedRegions;

HObject ho\_RegionFillUp, ho\_RegionDilation, ho\_RegionUnion;

HObject ho\_ImageReduced, ho\_Regions1, ho\_RegionErosion;

HObject ho\_ConnectedRegions, ho\_SelectedRegions1, ho\_RegionDilation1;

HObject ho\_SortedRegions, ho\_ObjectSelected\_1 = null;

// Local control variables 定义控制变量

public HTuple hv\_Width = null, hv\_Height = null, hv\_WindowHandle = new HTuple();

HTuple hv\_UsedThreshold = null, hv\_UsedThreshold1 = null;

HTuple hv\_Number = null, hv\_Area = null, hv\_Row = null;

HTuple hv\_Column = null, hv\_i = null, hv\_Area1 = null;

HTuple hv\_Row1 = null, hv\_Column1 = null, hv\_Area2 = null;

public HTuple hv\_area = null, hv\_bizhong = null;

// Main procedure

private void action(HObject x)

{

ho\_Image = x;

//ho\_Image.Dispose();

//Local iconic variables 定义图像变量

//HObject ho\_Image, ho\_GrayImage, ho\_Regions;

//HObject ho\_Connection, ho\_RegionResult, ho\_SelectedRegions;

//HObject ho\_RegionFillUp, ho\_RegionDilation, ho\_RegionUnion;

//HObject ho\_ImageReduced, ho\_Regions1, ho\_RegionErosion;

//HObject ho\_ConnectedRegions, ho\_SelectedRegions1, ho\_RegionDilation1;

//HObject ho\_SortedRegions, ho\_ObjectSelected\_1 = null;

//// Local control variables 定义控制变量

//HTuple hv\_Width = null, hv\_Height = null, hv\_WindowHandle = new HTuple();

//HTuple hv\_UsedThreshold = null, hv\_UsedThreshold1 = null;

//HTuple hv\_Number = null, hv\_Area = null, hv\_Row = null;

//HTuple hv\_Column = null, hv\_i = null, hv\_Area1 = null;

//HTuple hv\_Row1 = null, hv\_Column1 = null, hv\_Area2 = null;

//HTuple hv\_area = null, hv\_bizhong = null;

// Initialize local and output iconic variables 初始化本地图像变量和输出图像变量

HOperatorSet.GenEmptyObj(out ho\_Image);

HOperatorSet.GenEmptyObj(out ho\_GrayImage);

HOperatorSet.GenEmptyObj(out ho\_Regions);

HOperatorSet.GenEmptyObj(out ho\_Connection);

HOperatorSet.GenEmptyObj(out ho\_RegionResult);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions);

HOperatorSet.GenEmptyObj(out ho\_RegionFillUp);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation);

HOperatorSet.GenEmptyObj(out ho\_RegionUnion);

HOperatorSet.GenEmptyObj(out ho\_ImageReduced);

HOperatorSet.GenEmptyObj(out ho\_Regions1);

HOperatorSet.GenEmptyObj(out ho\_RegionErosion);

HOperatorSet.GenEmptyObj(out ho\_ConnectedRegions);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions1);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation1);

HOperatorSet.GenEmptyObj(out ho\_SortedRegions);

HOperatorSet.GenEmptyObj(out ho\_ObjectSelected\_1);

//程序开始

HOperatorSet.SetDraw(hv\_ExpDefaultWinHandle, "fill");

HOperatorSet.SetColored(hv\_ExpDefaultWinHandle, 12);

//读取图片

ho\_Image.Dispose();

HOperatorSet.ReadImage(out ho\_Image, "E:/WORKS/halcon/zhao\_jobs/zhao\_jobs/1-500-2.JPG");

//dev\_close\_window(...);

HOperatorSet.GetImageSize(ho\_Image, out hv\_Width, out hv\_Height);

//获取图像窗口

//dev\_open\_window(...);

//彩色图转成灰度图

ho\_GrayImage.Dispose();

HOperatorSet.Rgb1ToGray(ho\_Image, out ho\_GrayImage);

//阈值处理

ho\_Regions.Dispose();

HOperatorSet.BinaryThreshold(ho\_GrayImage, out ho\_Regions, "max\_separability",

"dark", out hv\_UsedThreshold);

//求图像联通域

ho\_Connection.Dispose();

HOperatorSet.Connection(ho\_Regions, out ho\_Connection);

//将区域闭合

ho\_RegionResult.Dispose();

HOperatorSet.CloseEdges(ho\_Connection, ho\_GrayImage, out ho\_RegionResult, 16);

//选择合适的区域

ho\_SelectedRegions.Dispose();

HOperatorSet.SelectShape(ho\_RegionResult, out ho\_SelectedRegions, "area", "and", 2040.82, 200000);

//填充区域孔洞

ho\_RegionFillUp.Dispose();

HOperatorSet.FillUp(ho\_SelectedRegions, out ho\_RegionFillUp);

//区域膨胀处理

ho\_RegionDilation.Dispose();

HOperatorSet.DilationCircle(ho\_RegionFillUp, out ho\_RegionDilation, 15.5);

//将区域群合成一个区域

ho\_RegionUnion.Dispose();

HOperatorSet.Union2(ho\_RegionDilation, ho\_RegionDilation, out ho\_RegionUnion);

// 获取刚兴趣区域图像

ho\_ImageReduced.Dispose();

HOperatorSet.ReduceDomain(ho\_GrayImage, ho\_RegionUnion, out ho\_ImageReduced);

//对感兴趣区域图像进行阈值处理

ho\_Regions1.Dispose();

HOperatorSet.BinaryThreshold(ho\_ImageReduced, out ho\_Regions1, "max\_separability", "dark", out hv\_UsedThreshold1);

//区域腐蚀处理，用于去除区域之间的粘结

ho\_RegionErosion.Dispose();

HOperatorSet.ErosionCircle(ho\_Regions1, out ho\_RegionErosion, 1.5);

//求图像连通域

ho\_ConnectedRegions.Dispose();

HOperatorSet.Connection(ho\_RegionErosion, out ho\_ConnectedRegions);

//选择合适的颗粒，去除无用的杂物

ho\_SelectedRegions1.Dispose();

HOperatorSet.SelectShape(ho\_ConnectedRegions, out ho\_SelectedRegions1, "area", "and", 2000, 9999999);

//对颗粒进行膨胀处理

ho\_RegionDilation1.Dispose();

HOperatorSet.DilationCircle(ho\_SelectedRegions1, out ho\_RegionDilation1, 10.5);

//数颗粒的个数

HOperatorSet.CountObj(ho\_RegionDilation1, out hv\_Number);

//对颗粒按照第一个点的行坐标进行排序

ho\_SortedRegions.Dispose();

HOperatorSet.SortRegion(ho\_RegionDilation1, out ho\_SortedRegions, "first\_point", "false", "row");

//求颗粒的面积和中心坐标

HOperatorSet.AreaCenter(ho\_SortedRegions, out hv\_Area, out hv\_Row, out hv\_Column);

//显示图像和选中的颗粒

HOperatorSet.DispObj(ho\_Image, hv\_ExpDefaultWinHandle);

HOperatorSet.DispObj(ho\_SortedRegions, hv\_ExpDefaultWinHandle);

//显示图像的代码

HTuple end\_val47 = hv\_Number;

HTuple step\_val47 = 1;

for (hv\_i = 1; hv\_i.Continue(end\_val47, step\_val47); hv\_i = hv\_i.TupleAdd(step\_val47))

{

ho\_ObjectSelected\_1.Dispose();

HOperatorSet.SelectObj(ho\_SortedRegions, out ho\_ObjectSelected\_1, hv\_i);

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, "red");

HOperatorSet.DispObj(ho\_ObjectSelected\_1, hv\_ExpDefaultWinHandle);

HOperatorSet.SetTposition(hv\_ExpDefaultWinHandle, hv\_Row.TupleSelect(hv\_i - 1),

hv\_Column.TupleSelect(hv\_i - 1));

HOperatorSet.SetColor(hv\_ExpDefaultWinHandle, "white");

HOperatorSet.WriteString(hv\_ExpDefaultWinHandle, hv\_i);

//write\_string (WindowHandle, Area[i-1])

}

//求面积占比

HOperatorSet.AreaCenter(ho\_Regions, out hv\_Area1, out hv\_Row1, out hv\_Column1);

hv\_Area2 = hv\_Area1.TupleReal();

hv\_area = ((hv\_Width \* hv\_Height)).TupleReal();

hv\_bizhong = hv\_Area2 / hv\_area;

bizhong = hv\_bizhong;

//decompose3 (Image, Image1, Image2, Image3)

ho\_Image.Dispose();

ho\_GrayImage.Dispose();

ho\_Regions.Dispose();

ho\_Connection.Dispose();

ho\_RegionResult.Dispose();

ho\_SelectedRegions.Dispose();

ho\_RegionFillUp.Dispose();

ho\_RegionDilation.Dispose();

ho\_RegionUnion.Dispose();

ho\_ImageReduced.Dispose();

ho\_Regions1.Dispose();

ho\_RegionErosion.Dispose();

ho\_ConnectedRegions.Dispose();

ho\_SelectedRegions1.Dispose();

ho\_RegionDilation1.Dispose();

ho\_SortedRegions.Dispose();

ho\_ObjectSelected\_1.Dispose();

}

public HObject showimg(HTuple Window)

{

hv\_ExpDefaultWinHandle = Window;

//HObject ho\_Image, ho\_GrayImage, ho\_Regions;

//HObject ho\_Connection, ho\_RegionResult, ho\_SelectedRegions;

//HObject ho\_RegionFillUp, ho\_RegionDilation, ho\_RegionUnion;

//HObject ho\_ImageReduced, ho\_Regions1, ho\_RegionErosion;

//HObject ho\_ConnectedRegions, ho\_SelectedRegions1, ho\_RegionDilation1;

//HObject ho\_SortedRegions, ho\_ObjectSelected\_1 = null;

// Local control variables

//HTuple hv\_Width = null, hv\_Height = null, hv\_WindowHandle = new HTuple();

// Initialize local and output iconic variables

HOperatorSet.GenEmptyObj(out ho\_Image);

HOperatorSet.GenEmptyObj(out ho\_GrayImage);

HOperatorSet.GenEmptyObj(out ho\_Regions);

HOperatorSet.GenEmptyObj(out ho\_Connection);

HOperatorSet.GenEmptyObj(out ho\_RegionResult);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions);

HOperatorSet.GenEmptyObj(out ho\_RegionFillUp);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation);

HOperatorSet.GenEmptyObj(out ho\_RegionUnion);

HOperatorSet.GenEmptyObj(out ho\_ImageReduced);

HOperatorSet.GenEmptyObj(out ho\_Regions1);

HOperatorSet.GenEmptyObj(out ho\_RegionErosion);

HOperatorSet.GenEmptyObj(out ho\_ConnectedRegions);

HOperatorSet.GenEmptyObj(out ho\_SelectedRegions1);

HOperatorSet.GenEmptyObj(out ho\_RegionDilation1);

HOperatorSet.GenEmptyObj(out ho\_SortedRegions);

HOperatorSet.GenEmptyObj(out ho\_ObjectSelected\_1);

//程序开始

HOperatorSet.SetDraw(hv\_ExpDefaultWinHandle, "fill");

HOperatorSet.SetColored(hv\_ExpDefaultWinHandle, 12);

ho\_Image.Dispose();

HOperatorSet.ReadImage(out ho\_Image, "E:/WORKS/halcon/zhao\_jobs/zhao\_jobs/1-500-2.JPG");

//dev\_close\_window(...);

HOperatorSet.GetImageSize(ho\_Image, out hv\_Width, out hv\_Height);

HOperatorSet.DispObj(ho\_Image, hv\_ExpDefaultWinHandle);

return (ho\_Image);

}

public void InitHalcon()

{

// Default settings used in HDevelop

HOperatorSet.SetSystem("width", 512);

HOperatorSet.SetSystem("height", 512);

}

public void RunHalcon(HTuple Window)

{

hv\_ExpDefaultWinHandle = Window;

action(ho\_Image);

}

}

}

# HALCON\_BIAO.Designer.cs

namespace Halcon\_region\_sort

{

partial class HALCON\_BIAO

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(HALCON\_BIAO));

this.label1 = new System.Windows.Forms.Label();

this.textBox1 = new System.Windows.Forms.TextBox();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

this.SuspendLayout();

//

// label1

//

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("隶书", 15.75F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.label1.Location = new System.Drawing.Point(26, 9);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(171, 21);

this.label1.TabIndex = 0;

this.label1.Text = "软件的操作流程";

//

// textBox1

//

this.textBox1.AccessibleRole = System.Windows.Forms.AccessibleRole.Grip;

this.textBox1.Font = new System.Drawing.Font("宋体", 15F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));

this.textBox1.Location = new System.Drawing.Point(787, 33);

this.textBox1.Multiline = true;

this.textBox1.Name = "textBox1";

this.textBox1.ReadOnly = true;

this.textBox1.Size = new System.Drawing.Size(442, 468);

this.textBox1.TabIndex = 1;

this.textBox1.Text = resources.GetString("textBox1.Text");

//

// pictureBox1

//

this.pictureBox1.Image = global::Halcon\_region\_sort.Properties.Resources.铁铺仪软件图1;

this.pictureBox1.Location = new System.Drawing.Point(2, 33);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(779, 438);

this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.pictureBox1.TabIndex = 2;

this.pictureBox1.TabStop = false;

//

// HALCON\_BIAO

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1241, 543);

this.Controls.Add(this.pictureBox1);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.label1);

this.MaximizeBox = false;

this.Name = "HALCON\_BIAO";

this.Text = "帮助文档";

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.PictureBox pictureBox1;

}

}

# HALCON\_BIAO.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Halcon\_region\_sort

{

public partial class HALCON\_BIAO : Form

{

public HALCON\_BIAO()

{

InitializeComponent();

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void HALCON\_BIAO\_Load(object sender, EventArgs e)

{

}

}

}

# EXCEL\_out.Designer.cs

namespace Halcon\_region\_sort

{

partial class EXCEL\_out

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.hWindowControl1 = new HalconDotNet.HWindowControl();

this.textBox1 = new System.Windows.Forms.TextBox();

this.textBox2 = new System.Windows.Forms.TextBox();

this.textBox3 = new System.Windows.Forms.TextBox();

this.textBox4 = new System.Windows.Forms.TextBox();

this.textBox5 = new System.Windows.Forms.TextBox();

this.textBox6 = new System.Windows.Forms.TextBox();

this.treeView1 = new System.Windows.Forms.TreeView();

this.button1 = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// hWindowControl1

//

this.hWindowControl1.BackColor = System.Drawing.Color.Black;

this.hWindowControl1.BorderColor = System.Drawing.Color.Black;

this.hWindowControl1.ImagePart = new System.Drawing.Rectangle(0, 0, 640, 480);

this.hWindowControl1.Location = new System.Drawing.Point(175, 2);

this.hWindowControl1.Name = "hWindowControl1";

this.hWindowControl1.Size = new System.Drawing.Size(476, 304);

this.hWindowControl1.TabIndex = 0;

this.hWindowControl1.WindowSize = new System.Drawing.Size(476, 304);

this.hWindowControl1.HMouseMove += new HalconDotNet.HMouseEventHandler(this.hWindowControl1\_HMouseMove);

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(175, 393);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(151, 21);

this.textBox1.TabIndex = 1;

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(440, 393);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(151, 21);

this.textBox2.TabIndex = 2;

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(175, 442);

this.textBox3.Name = "textBox3";

this.textBox3.Size = new System.Drawing.Size(240, 21);

this.textBox3.TabIndex = 3;

//

// textBox4

//

this.textBox4.Location = new System.Drawing.Point(440, 442);

this.textBox4.Name = "textBox4";

this.textBox4.Size = new System.Drawing.Size(151, 21);

this.textBox4.TabIndex = 4;

//

// textBox5

//

this.textBox5.Location = new System.Drawing.Point(175, 493);

this.textBox5.Name = "textBox5";

this.textBox5.Size = new System.Drawing.Size(151, 21);

this.textBox5.TabIndex = 5;

//

// textBox6

//

this.textBox6.Location = new System.Drawing.Point(440, 493);

this.textBox6.Name = "textBox6";

this.textBox6.Size = new System.Drawing.Size(151, 21);

this.textBox6.TabIndex = 6;

//

// treeView1

//

this.treeView1.Location = new System.Drawing.Point(1, 2);

this.treeView1.Name = "treeView1";

this.treeView1.Size = new System.Drawing.Size(168, 606);

this.treeView1.TabIndex = 7;

this.treeView1.AfterSelect += new System.Windows.Forms.TreeViewEventHandler(this.treeView1\_AfterSelect);

this.treeView1.NodeMouseDoubleClick += new System.Windows.Forms.TreeNodeMouseClickEventHandler(this.treeView1\_NodeMouseDoubleClick);

//

// button1

//

this.button1.Location = new System.Drawing.Point(335, 567);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(121, 31);

this.button1.TabIndex = 8;

this.button1.Text = "显示当前系统时间";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// EXCEL\_out

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(654, 610);

this.Controls.Add(this.button1);

this.Controls.Add(this.treeView1);

this.Controls.Add(this.textBox6);

this.Controls.Add(this.textBox5);

this.Controls.Add(this.textBox4);

this.Controls.Add(this.textBox3);

this.Controls.Add(this.textBox2);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.hWindowControl1);

this.Name = "EXCEL\_out";

this.Text = "浏览数据";

this.Load += new System.EventHandler(this.EXCEL\_out\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private HalconDotNet.HWindowControl hWindowControl1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.TextBox textBox3;

private System.Windows.Forms.TextBox textBox4;

private System.Windows.Forms.TextBox textBox5;

private System.Windows.Forms.TextBox textBox6;

private System.Windows.Forms.TreeView treeView1;

private System.Windows.Forms.Button button1;

}

}

# EXCEL\_out.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Microsoft.Office.Interop.Excel;

using HalconDotNet;

using System.IO;

//using Aspose.Cells;

namespace Halcon\_region\_sort

{

public partial class EXCEL\_out : Form

{

public HObject ho\_Image;

public string text;//定义传递的文件名

public string img;//定义传递的图像路径

public HTuple hv\_WinHandle;

public EXCEL\_out(String textName)

{

text = textName;

img = textName;

InitializeComponent();

}

private void EXCEL\_out\_Load(object sender, EventArgs e)

{

//生成treeview

PaintTreeView(treeView1, text);

//hv\_WinHandle = hWindowControl1.HalconWindow;

//HOperatorSet.GenEmptyObj(out ho\_Image);

//HOperatorSet.ReadImage(out ho\_Image, img);

//HOperatorSet.DispObj(ho\_Image, hv\_WinHandle);

//HOperatorSet.DumpWindow(hv\_WinHandle, "jpeg", "..\\text\\" + text);

//StreamReader sr = new StreamReader(text+".txt");

//char[] separator ={'\r','\n'};

//string [] spltxt=new string[100];

//string str = sr.ReadToEnd();

//spltxt = str.Split();

//textBox1.Text = spltxt[0];

//textBox2.Text = spltxt[1];

//textBox3.Text = spltxt[2];

//textBox4.Text = spltxt[3];

//textBox5.Text = spltxt[4];

//textBox6.Text = spltxt[5];

}

#region 生成程序所在根目录的TreeView

private void PaintTreeView(TreeView treeView, string fullPath)

{

try

{

treeView.Nodes.Clear(); //清空TreeView

DirectoryInfo dirs = new DirectoryInfo(fullPath); //获得程序所在路径的目录对象

DirectoryInfo[] dir = dirs.GetDirectories();//获得目录下文件夹对象

FileInfo[] file = dirs.GetFiles("\*.txt");//获得目录下txt文件对象

int dircount = dir.Count();//获得文件夹对象数量

int filecount = file.Count();//获得文件对象数量

//循环文件夹

for (int i = 0; i < dircount; i++)

{

treeView.Nodes.Add(dir[i].Name);

string pathNode = fullPath + "\\" + dir[i].Name;

//GetMultiNode(treeView.Nodes[i], pathNode);

}

//循环文件

for (int j = 0; j < filecount; j++)

{

treeView.Nodes.Add(file[j].Name);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\r\n出错的位置为：Form1.PaintTreeView()");

}

}

#endregion

private void hWindowControl1\_HMouseMove(object sender, HMouseEventArgs e)

{

}

private void treeView1\_AfterSelect(object sender, TreeViewEventArgs e)

{

}

#region 生成双击treeview node事件

private void treeView1\_NodeMouseDoubleClick(object sender, TreeNodeMouseClickEventArgs e)

{

TreeView treeView = (TreeView)sender;

string selectedNode = treeView.SelectedNode.FullPath; //获取双击的文件的文件名

char[] sep= { '.' };

string[] splNode = new string[10];

splNode = selectedNode.Split(sep);

//string datapath = selectedNode.ToString();

//读取并显示事例图像

hv\_WinHandle = hWindowControl1.HalconWindow;

HOperatorSet.GenEmptyObj(out ho\_Image);

HOperatorSet.ReadImage(out ho\_Image, img + selectedNode + ".jpg");

HOperatorSet.DispObj(ho\_Image, hv\_WinHandle);

// 读取数据文件

StreamReader sr = new StreamReader(text +splNode[0] +".txt");

//定义分割符

char[] separator = { '\r' };

string[] spltxt = new string[100];

string str = sr.ReadToEnd();

spltxt = str.Split(separator); //分割字符串

//MessageBox.Show(spltxt[1]);

//输出显示

textBox1.Text = spltxt[0];

textBox2.Text = spltxt[1];

textBox3.Text = spltxt[2];

textBox4.Text = spltxt[3];

textBox5.Text = spltxt[4];

textBox6.Text = spltxt[5];

}

#endregion

private void button1\_Click(object sender, EventArgs e)

{

System.DateTime currentTime = new System.DateTime();//定义时间类

currentTime = DateTime.Now; //获取当前系统的时间

int strD = currentTime.Day;

int strM = currentTime.Month;

int strY = currentTime.Year;

int strH = currentTime.Hour;

int strMIN = currentTime.Minute;

String strYMDH = "" + strY + strM + strD + "\_" + strH + strMIN;//合并转化成字符串

MessageBox.Show("北京时间：" + strYMDH );

}

}

}