ОПТИЧНО РАЗПОЗНАВАНЕ НА СИМВОЛИ И КОНВЕРТИРАНЕ НА ИЗОБРАЖЕНИЕ В ТЕКСТ ПОСРЕДСТВОМ КРОС КОРЕЛАЦИЯ

гл. ас. д-р Димитър Минчев, Бургаски Свободен Университет, Център по информатика и Технически науки, e-mail: mitko@bfu.bg

АБСТРАКТ

В тази публикация е демонстрирано как техниката крос корелация може да се използва за разпознаване на символи и конвертиране на изображенията в текст. Представен е демонстрационен експеримент написан на езика C# разработен посредством Microsoft Visual Studio.

ОСНОВНИ ПРИНЦИПИ НА ОБРАБОТКА ПРИ СИСТЕМИТЕ ЗА ОПТИЧНО РАЗПОЗНАВАНЕ НА СИМВОЛИ

Всички системи за оптично разпознаване на символи и конвертиране на изображение в текст, включват следните базови основни принципи при обработката на изображението, което следва да бъде разпознато в текст:

- Придобиване на изображение;
- Предварителна обработка;
- Сегментиране;
- Разпознаване на символите.

РАЗПОЗНАВАНЕ НА СИМВОЛИ С ПОМОЩТА НА КОРЕЛАЦИЯ

Корелацията е техника за съвпадение на сигнали. Тази техника е изключително важен компонент при цифровата обработка на сигнали. Тя често се използва при обработка на сигнали за анализиране на функции или серии от стойности, като например време честотни сигнали. Корелация е полезна, защото тя може да показва предсказуеми отношения, които могат да бъдат използвани в практиката.

Крос корелацията е стандартен метод за оценка на степента, в която две серии си съответстват. Да вземем за пример две серии x(i) и y(i), където i = 0, 1, 2 ... N-1. Крос корелацията r при закъснение d се определя като израз:

$$r = \frac{\sum_i [\,(x(i) - mx) * (y(i-d) - my)\,]}{\sqrt{\sum_i (x(i) - mx)^2}\sqrt{\sum_i (y(i-d) - my)^2}} \qquad (1)$$

където mx и my са осреднените стойности за съответната серия. Ако по-горният израз се изчисли за всички закъснения d = 0, 1, 2, ... N-1, тогава полученият израз е крос корелация, с дължина два пъти оригиналните серии.

$$r(d) = \frac{\sum_i [\,(x(i) - mx) * (y(i-d) - my)\,]}{\sqrt{\sum_i (x(i) - mx)^2}\sqrt{\sum_i (y(i-d) - my)^2}} \qquad (2)$$

Въпросът какво да се прави, когато индексът в серията е по-малко от 0 или по-голям или равен на броя на точките. (i-d < 0 или i-d >= N). Най-често срещаните подходи в тези случаи са или да игнорираме тези точки или да приемем, че сериите x и y са равни на нула за i < 0 и i >= N. В много приложения за цифрова обработка на сигнали, се

приема, че сериите са периодични, и в този случай индексите извън обхвата са "опаковани" в обхват, а именно: x(-1) = x(N-1), x(N+5) = x(5) и т.н.

Обхватът от закъснения d и по този начин дължината на крос корелационната серия може да бъде по-малка от N, например за да се провери съответствието само на кратки закъснения. Знаменателя в израза по-горе служи за нормализиране на корелационните коефициенти, така че -1 <= r(d) <= 1, граници показващи максималната корелация и нула показваща, че няма корелация. Висока отрицателна корелация показва висока корелация, но в обратната посока за една от сериите.

Максималната корелация се постига, като се имат предвид горните изрази, чрез плъзгане на втората серия данни покрай първата, при всяко изместване се изчислява сумата на новата подредба в сериите. Тази сума ще бъде най-голяма, когато изместването е такова, че структурата на двете серии съвпада. Това по същество е същата процедура като конволюцията, с изключение на нормализирането в знаменателя.

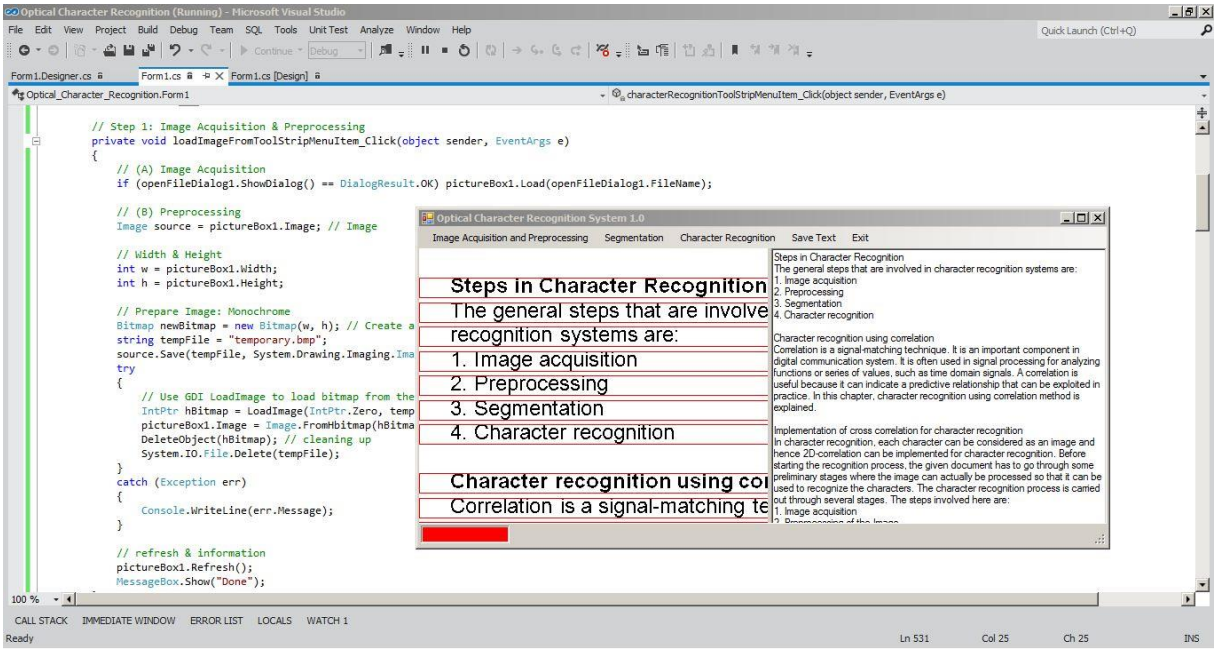
СТЪПКИ НА АЛГОРИТЪМ ЗА РАЗПОЗНАВАНЕ НА СИМВОЛИ БАЗИРАН НА КРОС КОРЕЛАЦИЯ

В системите за разпознаване на символи, всеки символ може да се счита като изображение и следователно може да се прилага дву-дименсионна корелация за разпознаването на символа. Преди да започне същинският процес на разпознаване на символите в даден документ, той трябва да мине през някои предварителни етапи на обработка, където изображението действително трябва да бъде преработено така, че да може да бъде използвано за разпознаване на символи.
Процеса на разпознаване на символи се извършва на няколко етапа, като стъпките на алгоритъма за разпознаване на символи базиран на крос корелация са:

- Придобиване на изображение;
- Предварителна обработка на изображението;
- Сегментиране;
- Извличане на символ;
- Разпознаване.

ПРОГРАМНА РЕАЛИЗАЦИЯ НА ТЕХНИКАТА ЗА РАЗПОЗНАВАНЕ НА СИМВОЛИ БАЗИРАН НА КРОС КОРЕЛАЦИЯ

В предложената реализация на техника за разпознаване на символи базиран на крос корелация се следват предложените по-горе стъпки на алгоритъма за разпознаване на символи. При етапа на предварителна обработка на изображението, с цел изчистване на дефекти (*например: при използване на JPEG изображение и компресия с загуба на качество*), то бива преобразувано в монохромно изображение (*2 бита*), което изчиства евентуалните пикселизирания и дефекти в изображението. При етапа на сегментиране изображението се разделя на редове. При етапа на извличане на символ от редовете се извличат изображенията на всеки символ в текста и се съхраняват отново като изображения.
При етапа на разпознаване на символите за всеки символ на сегментираното изображение се изготвят поредица от серии изображения за всички букви A .. Z и a .. z, които се сравняват със символа. Избира се тази буква от азбуката, която съответства на максималната корелация на изготвените серии. Предположението, че максималната корелация дава в резултат максималната вероятност получената буква да е търсения символ, се оказва вярно и е видно на работния екран на програмния продукт на Фиг.1.



Фиг.1. Разпознаване на символи и конвертиране на изображение в текст

ПРИЛОЖЕНИЕ С ПРОГРАМЕН КОД

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.IO;

// Optical Character Recognition
namespace Optical_Character_Recognition
{
  public partial class Form1 : Form
  {
    // const
    private const int LR_LOADFROMFILE = 0x0010;
    private const int LR_MONOCHROME = 0x0001;
    // var
    private List<Bitmap> content = new List<Bitmap>();

    // constructor
    public Form1()
    {
      InitializeComponent();
    }

    // Bitmap To Monochrome
    private Bitmap ToMonochrome(Bitmap bmp) // O(1)
    {
      // Width & Height
      int w = bmp.Width;
      int h = bmp.Height;

      // Create a blank bitmap the same size as original
      Bitmap bmp_result = new Bitmap(w, h);

      try
      {
        // Save as temporary file
        string tempFile = "temporary.bmp";
        bmp.Save(tempFile,
System.Drawing.Imaging.ImageFormat.Bmp);

        // creates 1 bpp bitmap
        IntPtr hBitmap = LoadImage(IntPtr.Zero,
tempFile, 0, 0, 0, (LR_LOADFROMFILE |
LR_MONOCHROME));
        // Creating Image object from HBITMAP handle
and saving it to destination file
        bmp_result = Bitmap.FromHbitmap(hBitmap);
        DeleteObject(hBitmap);
        System.IO.File.Delete(tempFile);
      }
      catch (Exception err)
      {
        Console.WriteLine(err.Message);
      }

      // result result image
      return bmp_result;
    }

    // user32.dll
    [DllImport("user32.dll")]
    static extern IntPtr LoadImage(IntPtr hinst, string
lpszName, uint uType, int cxDesired, int cyDesired, uint
fuLoad);

    // gdi32.dll
    [DllImport("gdi32.dll")]
    static extern bool DeleteObject(IntPtr hObject);

    // Step 1: Image Acquisition & Preprocessing
    private void
loadImageFromToolStripMenuItem_Click(object sender,
EventArgs e)
    {
      // (A) Image Acquisition
      if (openFileDialog1.ShowDialog() ==
DialogResult.OK)
pictureBox1.Load(openFileDialog1.FileName);

      // (B) Preprocessing
      Image source = pictureBox1.Image; // Image

      // Prepare Image: To Monochrome
      pictureBox1.Image =
(Image)ToMonochrome((Bitmap)pictureBox1.Image);

      // refresh
      pictureBox1.Refresh();

      // next menu
      segmentationToolStripMenuItem.Enabled = true;
    }

    // Step 2-A: Segmentation
    private void
segmentationToolStripMenuItem_Click(object sender,
EventArgs e) // O(N^2)
    {
      // Image
      Image source = pictureBox1.Image;
      Bitmap bmp = new Bitmap(source);
      bmp.Save("region//regions.bmp");

      // Width & Height
      int w = bmp.Width;
      int h = bmp.Height;

      // status
      ProgressBar1.Maximum = h;

      // Regions
      Rectangle[] regions = new Rectangle[h];
      int RegionCounter = 0;

      // prepare regions file
      StreamWriter fs = new
StreamWriter("region//regions.txt");

      // find regions
      int top = 0, bottom=0;
      for (int i = 0; i < h; i++)
      {
        int sum = 0;
        // rows sum
        for (int j = 1; j < w; j++)
          if ((bmp.GetPixel(j, i)).ToString() != "Color
[A=255, R=255, G=255, B=255]")
            sum++;
        // positions: top & bottom
        if ((sum > 0) && (top == 0)) top = i;
        if ((top > 0) && (bottom == 0) && (sum == 0))
        {
          bottom = i;
          // add new region
          regions[RegionCounter] = new Rectangle(0,
top, w, bottom - top);
          RegionCounter++; // increse region counter
          // text
          fs.WriteLine("0
"+System.Convert.ToString(top)+"
"+System.Convert.ToString(w)+"
"+System.Convert.ToString(bottom - top));
          bottom = 0;
          top = 0;
        }
        // progress
        ProgressBar1.Value++;
      }
      fs.Close();

      // status
      ProgressBar1.Maximum = RegionCounter;
      ProgressBar1.Value = 0;

      // Draw regions rectangles
      Graphics g = Graphics.FromImage(bmp); // get a
graphics object from the new image
```

```
        PointF ulCorner = new PointF(0F, 0F); // Create
point for upper-left corner of image.
        g.DrawImage(bmp, ulCorner);
        g.DrawRectangles(new Pen(Brushes.Red), regions);
// draw regions rectangles
        pictureBox1.Image = bmp;
        pictureBox1.Refresh();

        // Get Regions And Save Them
        GetRegionsAndSaveThem("region//regions.bmp");

        // next menu
        characterRecognitionToolStripMenuItem.Enabled =
true;
    }

    // Step 2-B: Get the Regions And Save Them as
Images
    private void GetRegionsAndSaveThem(string
filename)
    {
        if (File.Exists("region//subregions.txt"))
File.Delete("region//subregions.txt");

        // vars
        char[] delimiter = { ' ' };
        string line;
        int region = 1;

        // regions
        StreamReader fs = new
StreamReader("region//regions.txt");

        // read the regions
        while ((line = fs.ReadLine()) != null)
        {
            // preprocess
            string[] vars = new string[4];
            vars = line.Split(delimiter);
            int x = int.Parse(vars[0].ToString());
            int y = int.Parse(vars[1].ToString());
            int w = int.Parse(vars[2].ToString());
            int h = int.Parse(vars[3].ToString());

            // Crop and Save the Image Region
            Rectangle cropArea = new Rectangle(x, y, w, h);
            Bitmap SourceImage = new Bitmap(filename);
            Bitmap ImageRegion =
SourceImage.Clone(cropArea, SourceImage.PixelFormat);

            // Prepare Image: To Monochrome
            ImageRegion = ToMonochrome(ImageRegion);

            // Save Image
            ImageRegion.Save("region//" +
System.Convert.ToString(region) + ".bmp");

            // Get the Image Sub Regions
            GetSubRegionsAndSaveThem(ImageRegion,
region, w, h);

            // Next Region
            region++;

            // progress
            ProgressBar1.Value++;
        }
    }

    // Step 2-C: Get the Sub Regions And Save Them as
Images
    private void GetSubRegionsAndSaveThem(Bitmap
source, int region, int w, int h)
    {
        // sub regions
        StreamWriter fs = new
StreamWriter("region//"+region+".txt");

        int start = 0, end = 0, oldend = 0, subregion = 1;
        for (int i = 1; i < w; i++)
        {
            // find columns sums
            int sum = 0;
            for (int j = 1; j < h; j++)
            if ((source.GetPixel(i,j)).ToString() != "Color
[A=255, R=255, G=255, B=255]") sum++;

            // process
            if ((sum > 0) && (start == 0)) start = i;
            if ((sum == 0) && (start > 0)) end = i;
            if ((sum == 0) && (end > 0))
            {
                // Cropping Areas
                Rectangle cropArea = new Rectangle(start, 0,
end - start, h);
                Bitmap image1 = new Bitmap("region//" +
System.Convert.ToString(region) + ".bmp");
                Bitmap image2 = image1.Clone(cropArea,
image1.PixelFormat);

                // Clear Empty Spaces
                image2 = ClearEmptySpaces(image2);

                // Prepare Image: To Monochrome
                image2 = ToMonochrome(image2);

                // vars
                string re = System.Convert.ToString(region);
                string sr = System.Convert.ToString(subregion);
                string st = System.Convert.ToString(start);
                string ed = System.Convert.ToString(end);

                // Add to list for processing
                content.Add(image2);

                // Save for processing
                image2.Save("region//"+re + "-" + sr +
".bmp");
                fs.WriteLine(re + " " + sr + " " + st + " " + ed);

                // vars
                subregion++;
                start = 0;
                end = 0;
            }
        }
        fs.Close();

        // write subregions to file
        List<string> list = new List<string>();
        if (File.Exists("region//subregions.txt"))
        {
            string[] subregions_file =
File.ReadAllLines("region//subregions.txt");
            list.AddRange(subregions_file);
        }
        else File.Open("region//subregions.txt",
FileMode.Create).Close(); //
File.Create("region//subregions.txt");
        // insert
        list.Insert(list.Count,
System.Convert.ToString(subregion - 1));
        File.WriteAllLines("region//subregions.txt",
list.ToArray());
    }

    // Step 3-A: Create Image From Text
    // Generate Image from text using C# OR Convert
Text in to Image using C# (9 May 2008) by chiragrdarji
    //
http://chiragrdarji.wordpress.com/2008/05/09/generate-
image-from-text-using-c-or-convert-text-in-to-image-
using-c/
    public Bitmap CreateImageFromText(string text)
    {
        // Create font and brush.
        Font font = new Font("Arial", 50);
        SolidBrush brush = new SolidBrush(Color.Black);

        // Create the bmpImage again with the correct size
for the text and font.
        Bitmap bmp = new Bitmap(150, 150);

        // Create a graphics object to measure the text's
width and height.
        Graphics g = Graphics.FromImage(bmp);

        // Set Background color
        g.Clear(Color.White);

        // Draw string to screen.
        g.DrawString(text, font, brush, 0, 0);
        g.Flush();

        // Prepare Image: To Monochrome
        bmp = ToMonochrome(bmp);

        // Clear Empty Spaces
        bmp = ClearEmptySpaces(bmp);

        // Return the bitmap
        return bmp;
    }

    // Step 3-B: Clear Empty Spaces
    public static Bitmap ClearEmptySpaces(Bitmap
source) // 4 x O(N^2)
    {
        // preprocessing
        int h = source.Height;
        int w = source.Width;

        // top
        int top = 0;
        for (int i = 0; i < h; i++)
        {
            int sum = 0;
            // sum
            for (int j = 1; j < w; j++)
              if ((source.GetPixel(j,i)).ToString() != "Color
[A=255, R=255, G=255, B=255]")
                sum++;
            if ((sum > 0) && (top == 0))
            {
                top = i;
                break;
            }
        }
        // bottom
        int bottom = 0;
        for (int i = h - 1; i >= 0; i--)
        {
            int sum = 0;
            for (int j = 1; j < w; j++)
              if ((source.GetPixel(j, i)).ToString() != "Color
[A=255, R=255, G=255, B=255]")
                sum++;
            if ((sum > 0) && (bottom == 0))
            {
                bottom = i;
                break;
            }
        }
        // left
        int left = 0;
        for (int i = 0; i < w; i++)
        {
            int sum = 0;
            for (int j = 1; j < h; j++)
              if ((source.GetPixel(i, j)).ToString() != "Color
[A=255, R=255, G=255, B=255]")
                sum++;
            if ((sum > 0) && (left == 0))
            {
                left = i;
                break;
            }
        }
        // right
        int right = 0;
        for (int i = w - 1; i >= 0; i--)
        {
            int sum = 0;
            for (int j = h - 1; j > 0; j--)
              if ((source.GetPixel(i, j)).ToString() != "Color
[A=255, R=255, G=255, B=255]")
                sum++;
            if ((sum > 0) && (right == 0))
            {
                right = i;
                break;
            }
        }
        // Cropping Areas
        Rectangle cropArea = new Rectangle(left, top, right
- left + 1, bottom - top + 1);
        Bitmap ResultImage = source.Clone(cropArea,
source.PixelFormat);

        // result
        return ResultImage;
    }

    // Step 4: Normalized Cross Correlation
    // HELP:
http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/c
orrelate/
    public static double NCC(byte[] x, byte[] y)
    {
        int n = Math.Max(x.Length, y.Length);
        double CC = (1.0/(n-1)) * (Number(x,y) / (SD(x) *
SD(y)));
        return CC;
    }
    // Number
    public static double Number(byte[] x, byte[] y)
    {
        double avgx = Average(x);
        double avgy = Average(y);
        int n = Math.Min(x.Length, y.Length);
        double sum = 0;
        for (int i = 0; i < n; i++)
            sum = sum + ((x[i] - avgx) * (y[i] - avgy));
        return sum;
    }
```

```
    // Average
    public static double Average(byte[] data) // O(n)
    {
      double length = data.Length;
      if (length == 0) throw new Exception("No picture
data");
      double sum = 0;
      for (int i = 0; i < data.Length; i++) sum += data[i];
      return (sum / length);
    }
    // Variance
    public static double Variance(byte[] data) // O(n)
    {
      double length = data.Length;
      double avg = Average(data); // Average
      double sum = 0;
      for (int i = 0; i < data.Length; i++) sum +=
Math.Pow((data[i] - avg), 2);
      return (sum / length) - 1.0;
    }
    // Standard devaiation
    public static double SD(byte[] data)
    {
      return Math.Sqrt(Variance(data));
    }

    // Resizing an Image in C#
    public Bitmap ResizeImage(Bitmap Image, int w, int
h)
    {
      // Resize
      Bitmap bmp = new Bitmap(w, h);
      using (Graphics g = Graphics.FromImage(bmp))
      {

        g.InterpolationMode =
System.Drawing.Drawing2D.InterpolationMode.NearestN
eighbor; // HighQualityBicubic;
        g.DrawImage(Image, new
System.Drawing.Rectangle(0, 0, w, h));
      }

      // Prepare Image: To Monochrome
      bmp = ToMonochrome(bmp);

      // Return the bitmap
      return bmp;
    }

    // Step 5: Optical Character Recognition Processing
    private void
characterRecognitionToolStripMenuItem_Click(object
sender, EventArgs e)
    {
      // ASCII (48 .. 122)
      int min = 48, max = 122;
      // int min = 33, max = 126;

      // Step 5: Create Image From Text  (Arial font)
      Bitmap[] ARIAL = new Bitmap[512];
      // Load or Save
      for (int i = min; i <= max; i++)
      {
        // File Name
        String file = "ascii//" + i + ".bmp";
        // Load
        if (File.Exists(file)) ARIAL[i] = new Bitmap(file);
        else
        {
          // Save
          ARIAL[i] =
CreateImageFromText(System.Convert.ToString((char)i));
// ASCII
          ARIAL[i].Save( file );
        }
      }

      // read subregions file
      List<string> subregion_list = new List<string>();
      string[] subregions_file =
File.ReadAllLines("region//subregions.txt");
      subregion_list.AddRange(subregions_file);

      // progress
      ProgressBar1.Maximum = content.Count;
      ProgressBar1.Value = 0;

      // process the subregions
      foreach (Bitmap ImageToProcess in content)
      {
        // NEW: Spaces
        if (ImageToProcess == null)
        {
          textBox1.AppendText(" ");
          continue;
        }

        // Correlation
        double[] CORR = new double[512];

        // Max Correlation Index
        int MAX_CORR = 0;


        // Process
        for (int i = min; i <= max; i++)
        {
          // dimmensions
          double Aw = ARIAL[i].Width;
          double Ah = ARIAL[i].Height;
          double Iw = ImageToProcess.Width;
          double Ih = ImageToProcess.Height;
          // tolerance
          double TOLERANCE_W = Aw / 4; // W
          double TOLERANCE_H = Ah / 6; // H
          // tolerance check before resizing
          if (((Aw < Iw * (Ah / Ih) - TOLERANCE_W) ||
(Aw > Iw * (Ah / Ih) + TOLERANCE_W)) && // W
              ((Ah < Ih * (Aw / Iw) - TOLERANCE_H) || (Ah
> Ih * (Aw / Iw) + TOLERANCE_H)))  // H
          {
            CORR[i] = 0;
            continue;
          }

          // Resize image for cross corelation
          Bitmap Im = ImageToProcess;
          Im = ResizeImage(Im,(int)Aw,(int)Ah);


          // Convert images to byte
          MemoryStream stream = new
MemoryStream();
          ARIAL[i].Save(stream,
System.Drawing.Imaging.ImageFormat.Bmp);
          Byte[] image1 = stream.ToArray();
          Im.Save(stream,
System.Drawing.Imaging.ImageFormat.Bmp);
          Byte[] image2 = stream.ToArray();

          // Step 6: Normalized Cross Correlation
          CORR[i] = NCC(image1, image2);

          // print
          if (CORR[MAX_CORR] <= CORR[i]) MAX_CORR
= i;
        }
        // text

textBox1.AppendText(System.Convert.ToString((char)MAX
_CORR));
        // progress
        ProgressBar1.Value++;
      }

      // next menu
      saveTextToolStripMenuItem.Enabled = true;
    }

    // Save Text
    private void saveTextToolStripMenuItem_Click(object
sender, EventArgs e)
    {
      if (saveFileDialog1.ShowDialog() ==
DialogResult.OK)
      {
        string file = saveFileDialog1.FileName.ToString();
        string data = textBox1.Text;
        StreamWriter wf = new StreamWriter(file);
        wf.Write(data);
        wf.Close();
      }
    }

    // Exit
    private void exitToolStripMenuItem_Click(object
sender, EventArgs e)
    {
      Close();
    }
  }
}
```

ЛИТЕРАТУРА

[1] Д. Минчев, Дисертационен труд „ИНТЕРФЕРОМЕТРИЧНИ МЕТОДИ И АЛГОРИТМИ ЗА МОДЕЛИРАНЕ И ОБРАБОТКА НА САТЕЛИТНИ SAR ИЗОБРАЖЕНИЯ“, БАН, ИИКТ, Октомври 2012.

[2] Д. Минчев, Автореферат „ИНТЕРФЕРОМЕТРИЧНИ МЕТОДИ И АЛГОРИТМИ ЗА МОДЕЛИРАНЕ И ОБРАБОТКА НА САТЕЛИТНИ SAR ИЗОБРАЖЕНИЯ“, БАН, ИИКТ, Октомври 2012.

[3] Д. Минчев, Числено моделиране на диферентни интерферограми с ефективна подпикселна крос-корелационна корегистрация., Годишник БСУ, 2011.

[4] http://chiragrdarji.wordpress.com/2008/05/09/generate-image-from-text-using-c-or-convert-text-in-to-image-using-c/, Generate Image from text using C# OR Convert Text in to Image using C# (9 May 2008) by chiragrdarji, Интернет, Септември 2012.

[5] http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/correlate/, Normalized Cross Correlation, Интернет, Август 2012.