E.A.D ASSIGNMENT

Submitted by: M. Rehan Asghar BSSE 7 15126

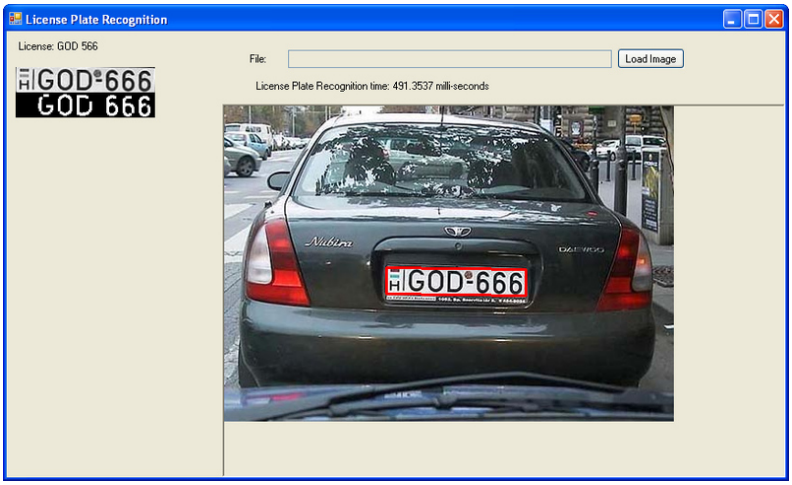
SEPTEMBER 11, 2017

SUBMITTED TO: SIR AWAIS SHAH

CS & IT Dept. Superior University

License Plate Recognition in C#

Program Code:



Program Screen Shot

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Text;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.OCR;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using Emgu.Util;

namespace LicensePlateRecognition
{
  public class LicensePlateDetector : DisposableObject
  {
    private Tesseract _ocr;
    public LicensePlateDetector(String dataPath)
    {
      _ocr = new Tesseract(dataPath, "eng", OcrEngineMode.TesseractCubeCombined);
      _ocr.SetVariable("tessedit_char_whitelist", "ABCDEFGHIJKLMNOPQRSTUVWXYZ-1234567890");
    }
    public List<String> DetectLicensePlate(
      IInputArray img,
```

```
        List<IInputOutputArray> licensePlateImagesList,
        List<IInputOutputArray> filteredLicensePlateImagesList,
        List<RotatedRect> detectedLicensePlateRegionList)
    {
        List<String> licenses = new List<String>();
        using (Mat gray = new Mat())
        using (Mat canny = new Mat())
        using (VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint())
        {
            CvInvoke.CvtColor(img, gray, ColorConversion.Bgr2Gray);
            CvInvoke.Canny(gray, canny, 100, 50, 3, false);
            int[,] hierachy = CvInvoke.FindContourTree(canny, contours,
ChainApproxMethod.ChainApproxSimple);
            FindLicensePlate(contours, hierachy, 0, gray, canny, licensePlateImagesList,
filteredLicensePlateImagesList, detectedLicensePlateRegionList, licenses);
        }
        return licenses;
    }
    private static int GetNumberOfChildren(int[,] hierachy, int idx)
    {
        idx = hierachy[idx,2];
        if (idx < 0)
            return 0;
        int count = 1;
        while (hierachy[idx,0] > 0)
        {
            count++;
            idx = hierachy[idx,0];
        }
        return count;
```

```
    }
    private void FindLicensePlate(
      VectorOfVectorOfPoint contours, int[,] hierachy, int idx, IInputArray gray, IInputArray
canny,
      List<IInputOutputArray> licensePlateImagesList, List<IInputOutputArray>
filteredLicensePlateImagesList, List<RotatedRect> detectedLicensePlateRegionList,
      List<String> licenses)
    {
      for (; idx >= 0;  idx = hierachy[idx,0])
      {
        int numberOfChildren = GetNumberOfChildren(hierachy, idx);
        //if it does not contains any children (charactor), it is not a license plate region
        if (numberOfChildren == 0) continue;

        using (VectorOfPoint contour = contours[idx])
        {
          if (CvInvoke.ContourArea(contour) > 400)
          {
            if (numberOfChildren < 3)
            {
              FindLicensePlate(contours, hierachy, hierachy[idx, 2], gray, canny,
licensePlateImagesList,
                filteredLicensePlateImagesList, detectedLicensePlateRegionList, licenses);
              continue;
            }
            RotatedRect box = CvInvoke.MinAreaRect(contour);
            if (box.Angle < -45.0)
            {
              float tmp = box.Size.Width;
              box.Size.Width = box.Size.Height;
```

```
            box.Size.Height = tmp;
            box.Angle += 90.0f;
          }
          else if (box.Angle > 45.0)
          {
            float tmp = box.Size.Width;
            box.Size.Width = box.Size.Height;
            box.Size.Height = tmp;
            box.Angle -= 90.0f;
          }

          double whRatio = (double) box.Size.Width/box.Size.Height;
          if (!(3.0 < whRatio && whRatio < 10.0))
            //if (!(1.0 < whRatio && whRatio < 2.0))
          {
            if (hierachy[idx, 2] > 0)
              FindLicensePlate(contours, hierachy, hierachy[idx, 2], gray, canny,
licensePlateImagesList,
                filteredLicensePlateImagesList, detectedLicensePlateRegionList, licenses);
            continue
        }
          using (UMat tmp1 = new UMat())
          using (UMat tmp2 = new UMat())
          {
            PointF[] srcCorners = box.GetVertices();
            PointF[] destCorners = new PointF[] {
              new PointF(0, box.Size.Height - 1),
              new PointF(0, 0),
              new PointF(box.Size.Width - 1, 0),
              new PointF(box.Size.Width - 1, box.Size.Height - 1)};
```

```
            using (Mat rot = CameraCalibration.GetAffineTransform(srcCorners,
destCorners))
            {
              CvInvoke.WarpAffine(gray, tmp1, rot, Size.Round(box.Size));
            }
            Size approxSize = new Size(240, 180);
            double scale = Math.Min(approxSize.Width/box.Size.Width,
approxSize.Height/box.Size.Height);
            Size newSize = new Size( (int)Math.Round(box.Size.Width*scale),(int)
Math.Round(box.Size.Height*scale));
            CvInvoke.Resize(tmp1, tmp2, newSize, 0, 0, Inter.Cubic);
            int edgePixelSize = 2;
            Rectangle newRoi = new Rectangle(new Point(edgePixelSize, edgePixelSize),
              tmp2.Size - new Size(2*edgePixelSize, 2*edgePixelSize));
            UMat plate = new UMat(tmp2, newRoi);

            UMat filteredPlate = FilterPlate(plate);

            Tesseract.Character[] words;
            StringBuilder strBuilder = new StringBuilder();
            using (UMat tmp = filteredPlate.Clone())
            {
              _ocr.Recognize(tmp);
              words = _ocr.GetCharacters();
              if (words.Length == 0) continue;
              for (int i = 0; i < words.Length; i++)
              {
                strBuilder.Append(words[i].Text);
              }
```

```
                }
                licenses.Add(strBuilder.ToString());
                licensePlateImagesList.Add(plate);
                filteredLicensePlateImagesList.Add(filteredPlate);
                detectedLicensePlateRegionList.Add(box);
              }
            }
          }
        }
    }
    private static UMat FilterPlate(UMat plate)
    {
      UMat thresh = new UMat();
      CvInvoke.Threshold(plate, thresh, 120, 255, ThresholdType.BinaryInv);
      //Image<Gray, Byte> thresh = plate.ThresholdBinaryInv(new Gray(120), new
Gray(255));

      Size plateSize = plate.Size;
      using (Mat plateMask = new Mat(plateSize.Height, plateSize.Width, DepthType.Cv8U,
1))
      using (Mat plateCanny = new Mat())
      using (VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint())
      {
        plateMask.SetTo(new MCvScalar(255.0));
        CvInvoke.Canny(plate, plateCanny, 100, 50);
        CvInvoke.FindContours(plateCanny, contours, null, RetrType.External,
ChainApproxMethod.ChainApproxSimple);

        int count = contours.Size;
        for (int i = 1; i < count; i++)
```

```
        {
          using (VectorOfPoint contour = contours[i])
          {
            Rectangle rect = CvInvoke.BoundingRectangle(contour);
            if (rect.Height > (plateSize.Height >> 1))
            {
              rect.X -= 1; rect.Y -= 1; rect.Width += 2; rect.Height += 2;
              Rectangle roi = new Rectangle(Point.Empty, plate.Size);
              rect.Intersect(roi);
              CvInvoke.Rectangle(plateMask, rect, new MCvScalar(), -1);
              //plateMask.Draw(rect, new Gray(0.0), -1);
            }
          }

        }
        thresh.SetTo(new MCvScalar(), plateMask);
      }
      CvInvoke.Erode(thresh, thresh, null, new Point(-1, -1), 1, BorderType.Constant,
CvInvoke.MorphologyDefaultBorderValue);
      CvInvoke.Dilate(thresh, thresh, null, new Point(-1, -1), 1, BorderType.Constant,
CvInvoke.MorphologyDefaultBorderValue);
      return thresh;
    }
    protected override void DisposeObject()
    {
      _ocr.Dispose();
    }
  }
}
```