

**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HCM**  
**KHOA ĐIỆN-ĐIỆN TỬ**  
**BỘ MÔN VIỄN THÔNG**

---



**TIỂU LUẬN XỬ LÝ ẢNH**  
**ĐỀ TÀI: ỨNG DỤNG NHẬN DIỆN KÍ TỰ TỪ ẢNH**  
**TRÊN C SHARP**

**Sinh viên: Trần Nguyên Tiến Đạt - 1410846**  
**Trần Minh Quang – 1413112**  
**Nguyễn Duy Tân – 1413450**

## I. TỔNG QUAN:

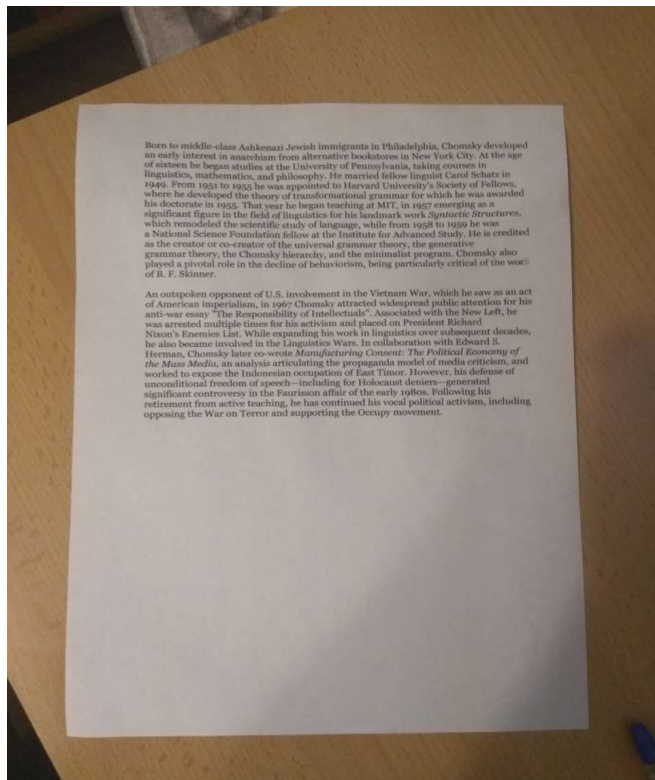
### 1. Đặt vấn đề:

Giả sử chúng ta cần ghi chép một thông tin mà chúng ta không có nhiều thời gian để ghi chép. Chẳng hạn như ghi lại bài tập trên bảng trước khi nó được xóa nhanh.



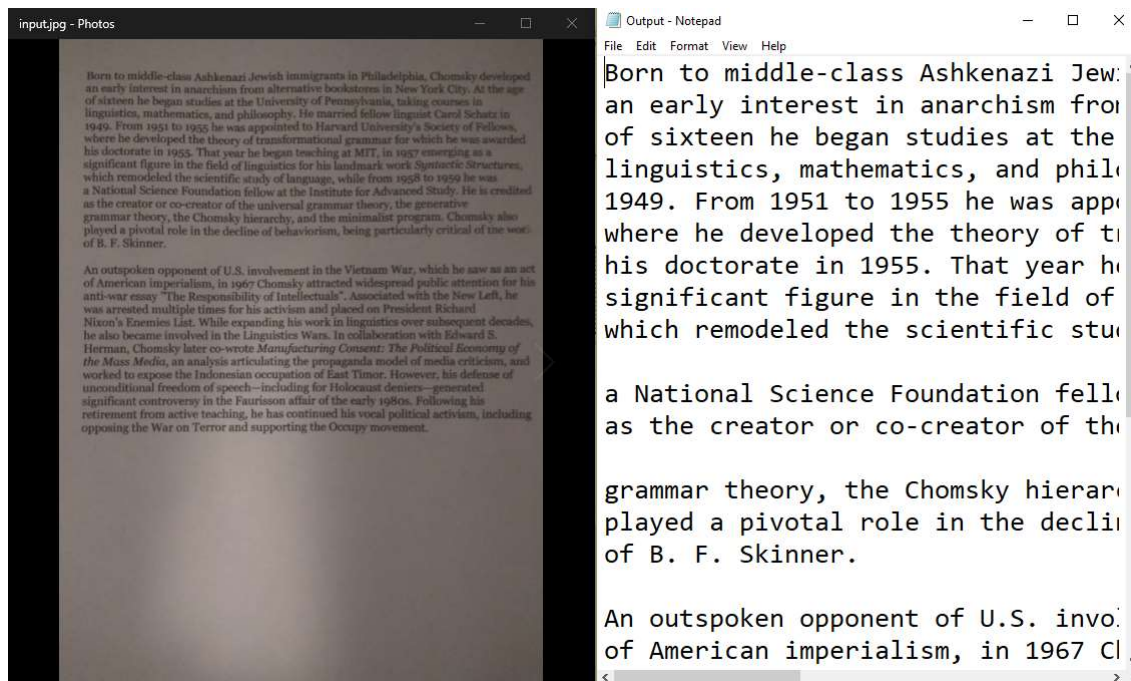
Hình 1.1 Ứng dụng của ảnh số.

Hoặc chúng ta muốn lấy thông tin từ văn bản mà ta chỉ có ảnh của nó mà không có file text.



Hình 1.2 Văn bản không có file kí tự hỗ trợ

Giải pháp: Vậy cách giải quyết đơn giản nhất là dùng bức hình chụp nhanh đó để xử lý cho ra đoạn text như chúng ta mong muốn.



Hình 1.3 Giải pháp nhận diện ký tự từ ảnh

## 2. Phần mềm và thư viện:






- Phần mềm: Dùng ngôn ngữ C# Visual Studio.
- Thư viện: Emgu.C, Tesseract.
- Cách thêm thư viện Emgu.CV vào C#:

## 3. Tích hợp thư viện EmguCV vào ứng dụng trên C#:

*Bước 1:* Tải thư viện EmguCV (từ đường dẫn: <https://sourceforge.net/projects/emgu/>) và cài vào máy tính.

*Bước 2:* Tạo một ứng dụng Application Form trên phần mềm Visual Studio.

*Bước 3:* Ở phần Reference trên cây thư mục project, nhấn phải chuột và chọn Add Reference... Sau đó, chọn Browse và trở thư mục đến đường dẫn EmguCV vừa cài đặt, chọn các tập tin .dll sau:

 Emgu.CV.DebuggerVisualizers.VS2017.dll	30/10/2017 1:08 PM	Application extens...	8 KB
 Emgu.CV.UI.dll	30/10/2017 1:08 PM	Application extens...	115 KB
 Emgu.CV.UI.GL.dll	30/10/2017 1:08 PM	Application extens...	27 KB
 Emgu.CV.World.dll	30/10/2017 1:08 PM	Application extens...	608 KB
 ZedGraph.dll	2/9/2017 3:57 PM	Application extens...	300 KB

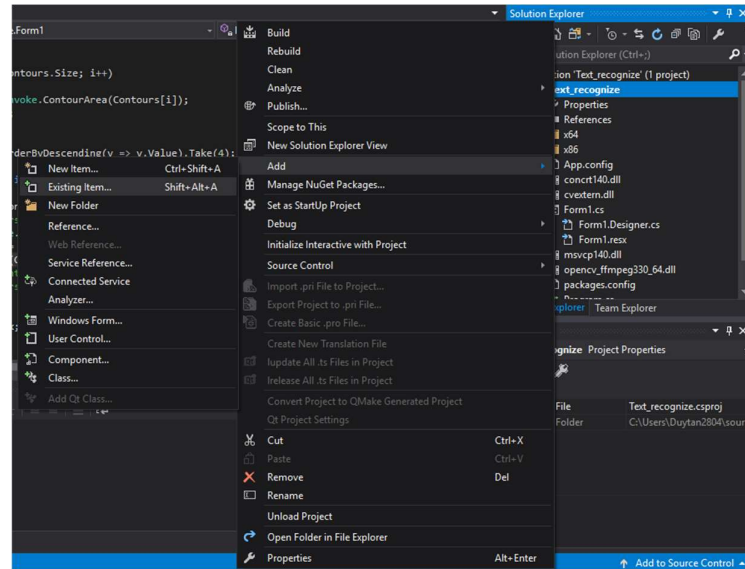
Hình 1.4 Các tập tin Reference cần thiết

Tiếp tục chọn Browse và trở đến đường dẫn:

C:/Windows/Microsoft.NET/Framework/v4.0.30319

để bổ sung tập tin System.ServiceModel.dll

*Bước 4:* Ở cây thư mục, nhấp phải chuột và chọn Add... Để thêm tất cả các tập tin trong thư mục C:/EmguCV/bin.



Hình 1.5 Thêm tập tin .dll vào project.

*Bước 5:* Nhấp phải chọn tất cả các tập tin .dll vừa mới bổ sung và nhấp phải chọn Properties, tiếp theo đó vào thuộc tính Copy to Output Directory, chọn Copy always.

*Bước 6:* Nhấp phải chuột vào project và chọn Properties > Build. Ở mục Platform target chọn x64.

#### 4. Tích hợp Tesseract vào Visual Studio:

- Giới thiệu tổng quan:
  - Tesseract là một công cụ OCR ( Optimal Character Recognition), với sự hỗ trợ của Unicode. Có khả năng nhận dạng hơn 100 ngôn ngữ, dùng để chuyển các hình ảnh chữ viết tay hoặc đánh máy sang các văn bản tài liệu. Nó có thể được huấn luyện để nhận dạng tốt hơn.
  - Tesseract được sử dụng để nhận dạng chữ viết từ hình ảnh, hỗ trợ trên mobile devices, video, Gmail, hỗ trợ trên rất nhiều nền tảng như Window, IOS, và nhiều ngôn ngữ như C#, Python...

- Các bước thực hiện:

*Bước 1:* Truy cập vào đường dẫn sau và chọn version mong muốn:

<https://www.nuget.org/packages/Tesseract/3.0.2>

Mở cửa sổ Package Manager Console và gõ câu lệnh với version tương ứng từ link trên. Ở đây nhóm chọn version 3.0.2 nên sẽ gõ dòng lệnh sau:

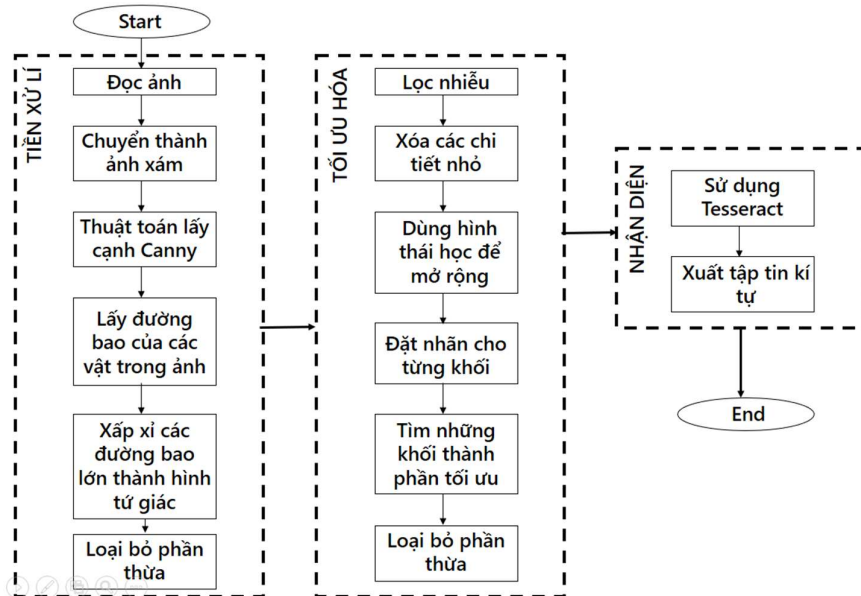
Install-Package Tesseract -Version 3.0.2

Bước 2: Vào đường link sau chọn và tải gói data training tương ứng với version và ngôn ngữ mong muốn, tải về và copy vào folder Debug của project:

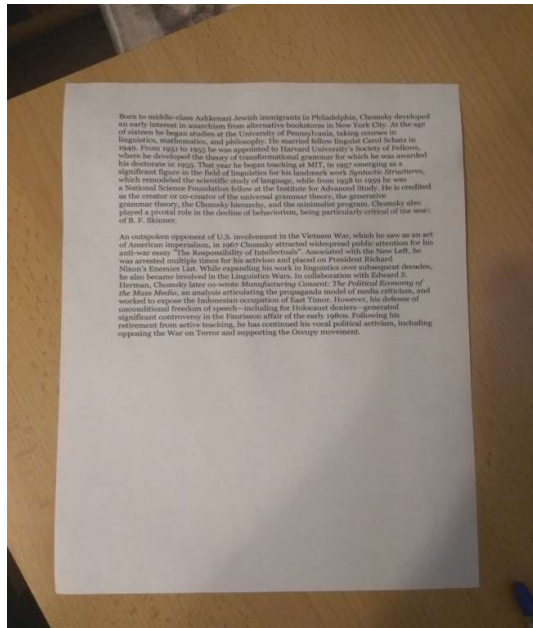
<https://github.com/tesseract-ocr/tesseract/wiki/Data-Files>

## II. NỘI DUNG THUẬT TOÁN:

### 1. Lưu đồ giải thuật:

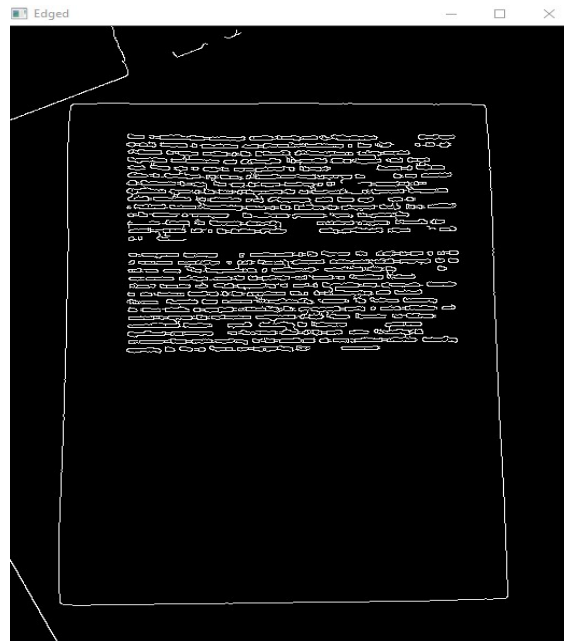


### 2. Các bước thực hiện:



Hình II.1 Ảnh gốc

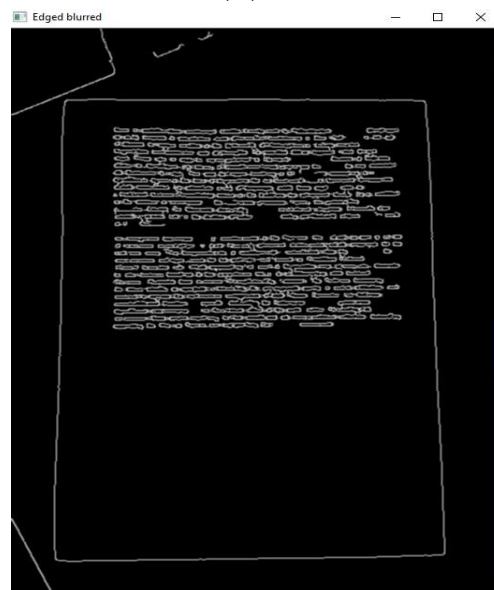
- Để tách được chữ, ta dùng bộ lọc Canny để lấy cạnh:



Hình II.2 Ảnh sau khi sử dụng bộ lọc Canny

- Để nhận biết các loại ảnh có cạnh không vuông như hình chữ nhật, ta dùng thêm bộ lọc Gaussian

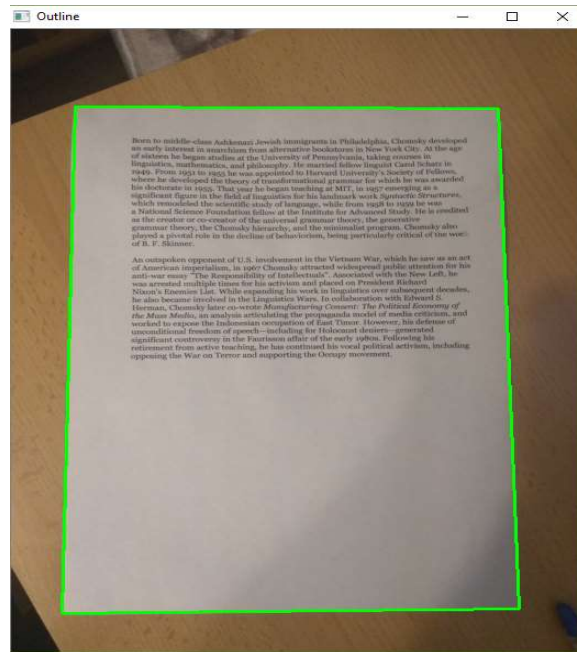
`edge_copy = edge_copy.SmoothGaussian(3); // Kích thước bộ lọc 3x3`



Hình II.3 Ảnh sau khi áp dụng bộ lọc Gaussian

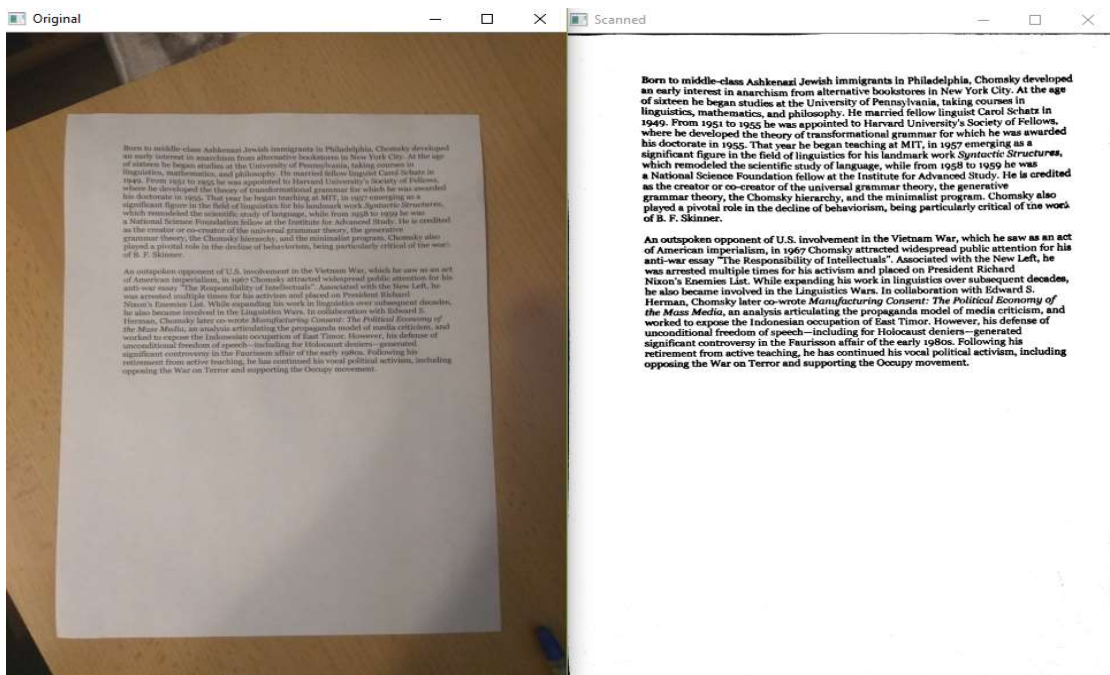
- Để đơn giản việc nhận dạng, ta bỏ các phần thừa, chỉ lấy phần chữ viết. Dùng hàm Contour để thực hiện:
  - Hàm contour trả về các cạnh với số điểm tương ứng.





Hình II.4 Ảnh khi được phát hiện khung hình

- Sau khi có được contour, thực hiện cắt đi phần không cần thiết:



Hình II.5 Cắt bỏ phần ảnh ngoài khung hình

- Tiếp theo, để lọc nhiễu, ta thực hiện phép dilation:

Born to middle-class Ashkenazi Jewish immigrants in Philadelphia, Chomsky developed an early interest in anarchism from alternative bookstores in New York City. At the age of sixteen he began studies at the University of Pennsylvania, taking courses in linguistics, mathematics, and philosophy. He married fellow linguist Carol Schatz in 1949. From 1951 to 1955 he was appointed to Harvard University's Society of Fellows, where he developed the theory of transformational grammar for which he was awarded his doctorate in 1955. That year he began teaching at MIT, in 1957 emerging as a significant figure in the field of linguistics for his landmark work *Syntactic Structures*, which remodeled the scientific study of language, while from 1958 to 1959 he was a National Science Foundation fellow at the Institute for Advanced Study. He is credited as the creator or co-creator of the universal grammar theory, the generative grammar theory, the Chomsky hierarchy, and the minimalist program. Chomsky also played a pivotal role in the decline of behaviorism, being particularly critical of the work of B. F. Skinner.

An outspoken opponent of U.S. involvement in the Vietnam War, which he saw as an act of American imperialism, in 1967 Chomsky attracted widespread public attention for his anti-war essay "The Responsibility of Intellectuals". Associated with the New Left, he was arrested multiple times for his activism and placed on President Richard Nixon's Enemies List. While expanding his work in linguistics over subsequent decades, he also became involved in the Linguistics Wars. In collaboration with Edward S. Herman, Chomsky later co-wrote *Manufacturing Consent: The Political Economy of the Mass Media*, an analysis articulating the propaganda model of media criticism, and worked to expose the Indonesian occupation of East Timor. However, his defense of unconditional freedom of speech—including for Holocaust deniers—generated significant controversy in the Faurisson affair of the early 1980s. Following his retirement from active teaching, he has continued his vocal political activism, including opposing the War on Terror and supporting the Occupy movement.

Born to middle-class Ashkenazi Jewish immigrants in Philadelphia, Chomsky developed an early interest in anarchism from alternative bookstores in New York City. At the age of sixteen he began studies at the University of Pennsylvania, taking courses in linguistics, mathematics, and philosophy. He married fellow linguist Carol Schatz in 1949. From 1951 to 1955 he was appointed to Harvard University's Society of Fellows, where he developed the theory of transformational grammar for which he was awarded his doctorate in 1955. That year he began teaching at MIT, in 1957 emerging as a significant figure in the field of linguistics for his landmark work *Syntactic Structures*, which remodeled the scientific study of language, while from 1958 to 1959 he was a National Science Foundation fellow at the Institute for Advanced Study. He is credited as the creator or co-creator of the universal grammar theory, the generative grammar theory, the Chomsky hierarchy, and the minimalist program. Chomsky also played a pivotal role in the decline of behaviorism, being particularly critical of the work of B. F. Skinner.

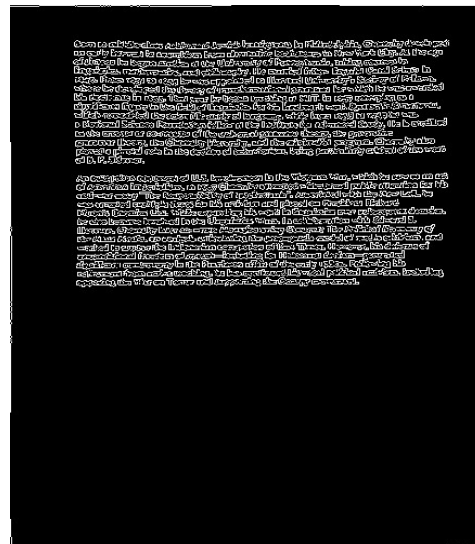
An outspoken opponent of U.S. involvement in the Vietnam War, which he saw as an act of American imperialism, in 1967 Chomsky attracted widespread public attention for his anti-war essay "The Responsibility of Intellectuals". Associated with the New Left, he was arrested multiple times for his activism and placed on President Richard Nixon's Enemies List. While expanding his work in linguistics over subsequent decades, he also became involved in the Linguistics Wars. In collaboration with Edward S. Herman, Chomsky later co-wrote *Manufacturing Consent: The Political Economy of the Mass Media*, an analysis articulating the propaganda model of media criticism, and worked to expose the Indonesian occupation of East Timor. However, his defense of unconditional freedom of speech—including for Holocaust deniers—generated significant controversy in the Faurisson affair of the early 1980s. Following his retirement from active teaching, he has continued his vocal political activism, including opposing the War on Terror and supporting the Occupy movement.

(a) Trước khi áp dụng

(b) Sau khi áp dụng

Hình II.6 Áp dụng phép biến đổi dilation

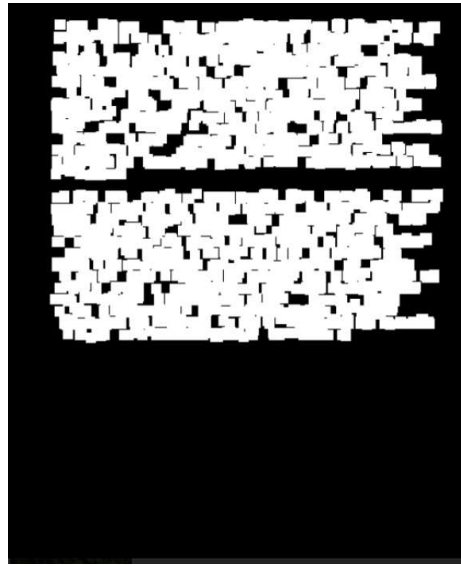
- Ở đây vẫn chưa thể recognize text hiệu quả được, ta cần phải crop bỏ phần không có chữ để thuật toán nhận dạng chữ được tập trung vào vùng có chữ, điều này cũng giúp nhận diện nhanh hơn , gồm các bước :
  - Dùng bộ median blur để xóa khung còn dư và xóa các chi tiết nhỏ :



Hình II.7 Sau khi áp dụng bộ lọc median để xóa bỏ các chi tiết nhỏ

- Để nhận dạng được các khối có chữ, ta dùng thuật toán binary dilation, dùng để liên kết các pixel trắng lại với nhau:





Hình II.8 Nhận diện khối chữ

- Ta sẽ đặt một mức ngưỡng, nếu các pixel này liên kết tạo thành số khối nhỏ hơn mức ngưỡng sẽ ngưng thực hiện.
- Sau đó, để thực hiện việc “label” cho các khối, ta dùng giải thuật connected components (dạng row-by-row) như sau:
  - Label = 0.
  - Quét từng pixel từng hàng, nếu là pixel 1 thì xét tiếp:
    - Có liên kết với những pixel phía trên và phía trước, đặt label cho pixel.
    - Không liên kết, Label++, đặt label.
  - Nếu pixel 0 thì đặt nhãn 0 và tiếp tục quét.
- Sau khi có components, ta tiến hành thực hiện tìm các components tối ưu (find optimal components), việc này được dựa trên 2 tiêu chí :
  - Tối đa số pixel trắng trong tổng số pixel. (Precision)
  - Càng nhỏ càng tốt. (Recall)
  - Nói cách khác, tối ưu hóa F1-Score, tức có sự cân bằng giữa Precision và Recall:

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Thuật toán:
  - Ta sắp số các components có diện tích pixel trắng từ lớn đến nhỏ. (N components)
  - Tính Recall và Preciso cho components k-1 và k (k max = N) cộng dồn cho đến khi F1 nhỏ hơn giá trị trước đó hoặc chu kỳ lặp thì kết thúc.

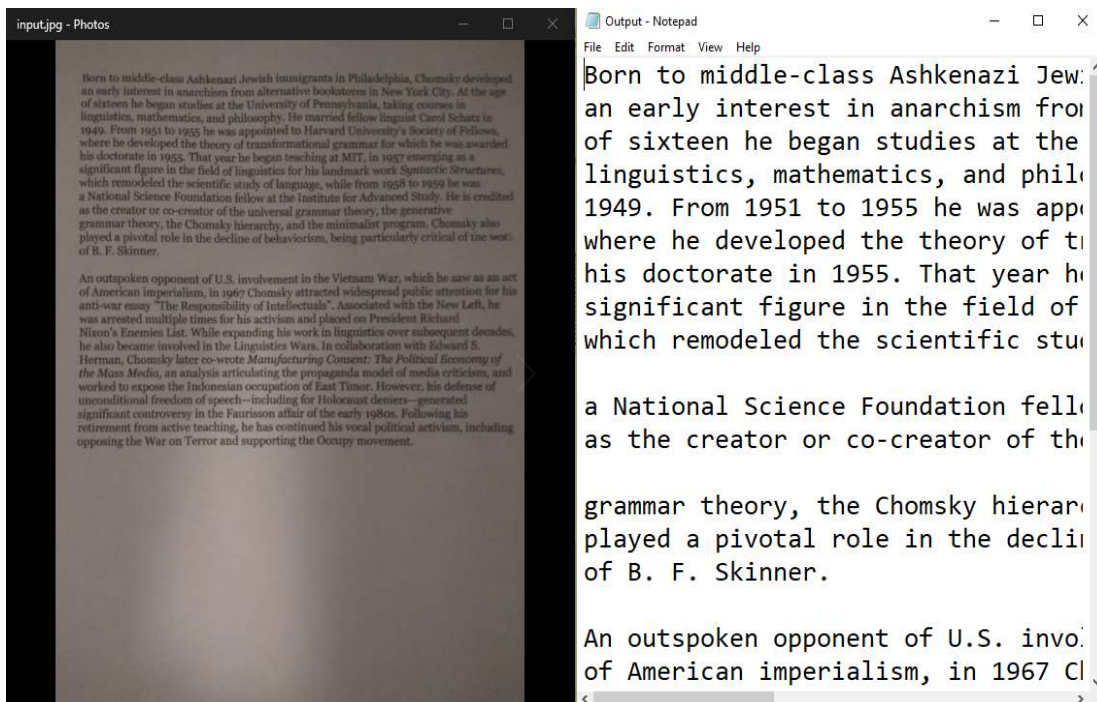
- Kết quả lấy đoạn chỉ có chữ :

Born to middle-class Ashkenazi Jewish immigrants in Philadelphia, Chomsky developed an early interest in anarchism from alternative bookstores in New York City. At the age of sixteen he began studies at the University of Pennsylvania, taking courses in linguistics, mathematics, and philosophy. He married fellow linguist Carol Schatz in 1949. From 1951 to 1955 he was appointed to Harvard University's Society of Fellows, where he developed the theory of transformational grammar for which he was awarded his doctorate in 1955. That year he began teaching at MIT, in 1957 emerging as a significant figure in the field of linguistics for his landmark work *Syntactic Structures*, which remodeled the scientific study of language, while from 1958 to 1959 he was a National Science Foundation fellow at the Institute for Advanced Study. He is credited as the creator or co-creator of the universal grammar theory, the generative grammar theory, the Chomsky hierarchy, and the minimalist program. Chomsky also played a pivotal role in the decline of behaviorism, being particularly critical of the work of B. F. Skinner.

An outspoken opponent of U.S. involvement in the Vietnam War, which he saw as an act of American imperialism, in 1967 Chomsky attracted widespread public attention for his anti-war essay "The Responsibility of Intellectuals". Associated with the New Left, he was arrested multiple times for his activism and placed on President Richard Nixon's Enemies List. While expanding his work in linguistics over subsequent decades, he also became involved in the Linguistics Wars. In collaboration with Edward S. Herman, Chomsky later co-wrote *Manufacturing Consent: The Political Economy of the Mass Media*, an analysis articulating the propaganda model of media criticism, and worked to expose the Indonesian occupation of East Timor. However, his defense of unconditional freedom of speech—including for Holocaust deniers—generated significant controversy in the Faurisson affair of the early 1980s. Following his retirement from active teaching, he has continued his vocal political activism, including opposing the War on Terror and supporting the Occupy movement.

Hình II.9 Ảnh đã được cắt với những phần có chữ cần thiết

- Sau khi cắt ảnh, ta bỏ qua bộ nhận diện chữ Tesseract để nhận diện:



Hình II.10 Chữ đã được nhận diện

### 3. Nội dung code C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```

using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

using Emgu;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using Emgu.CV.CvEnum;
using Emgu.CV.UI;
using Tesseract;
using System.Drawing.Imaging;

namespace Text_recognize
{
    public partial class Form1 : Form
    {
        VideoCapture capture;
        static uint ButtonID = 0;
        Image<Bgr, byte> My_Image;
        Image<Gray, byte> gray_image;
        Image<Bgr, byte> My_image_resize;
        Image<Bgr, byte> My_image_display;
        Image<Bgr, byte> orig;
        Image<Gray, byte> gray;
        Image<Gray, byte> edge;
        Image<Gray, byte> edge_copy;
        Image<Bgr, byte> warp_copy;
    }
}

```

```
VectorOfVectorOfPoint Contours = new VectorOfVectorOfPoint();  
VectorOfPoint screenCnt = new VectorOfPoint();  
double ratio;
```

```
public Form1()  
{  
    InitializeComponent();  
}
```

```
private void Form1_Load(object sender, EventArgs e)  
{  
  
}
```

```
private void Load_btn_Click(object sender, EventArgs e)  
{  
    ButtonID = 0;  
    OpenFileDialog Openfile = new OpenFileDialog();  
    if (Openfile.ShowDialog() == DialogResult.OK)  
    {  
  
        //Display the Image  
        My_Image = new Image<Bgr, byte>(Openfile.FileName);  
        Display_ptbx.SizeMode = PictureBoxSizeMode.AutoSize;  
        Display_ptbx.Image = My_Image.ToBitmap();  
        orig = My_Image.Copy();  
    }  
}
```

```
private void Cam_btn_Click(object sender, EventArgs e)  
{  
    if (capture == null)  
    {
```

```

        capture = new VideoCapture(1);
    }
    capture.ImageGrabbed += Capture_ImageGrabbed;
    capture.Start();
}
private void Capture_ImageGrabbed(object sender, EventArgs e)
{
    try
    {
        Mat m = new Mat();
        capture.Retrieve(m);
        Display_ptbx.SizeMode = PictureBoxSizeMode.Normal;
        Display_ptbx.Image = m.ToImage<Bgr, byte>().Bitmap;
        My_Image = m.ToImage<Bgr, byte>();
    }
    catch (Exception)
    {

    }
}

private void Stop_btn_Click(object sender, EventArgs e)
{
    if (capture != null)
    {
        capture = null;
    }
}

private void Process_btn_Click(object sender, EventArgs e)
{
    if (My_Image != null)

```

```

{
    int Resized_Height, Resized_Width;

    switch (ButtonID)
    {
        case 0:
            ratio = My_Image.Height / 700.0;
            orig = My_Image.Copy();
            Resized_Height = 700;
            Resized_Width = Convert.ToInt16(700.0 * My_Image.Width /
My_Image.Height);
            My_image_resize = My_Image.Resize(Resized_Width, Resized_Height,
Inter.Linear);

            Display_ptbx.Image = My_image_resize.ToBitmap();
            ButtonID++;
            Process_btn.Text = "Blur";
            break;

        case 1:
            gray = My_image_resize.Convert<Gray, Byte>();
            gray = gray.SmoothMedian(5);
            Display_ptbx.Image = gray.ToBitmap();
            ButtonID++;
            Process_btn.Text = "Edge";
            break;

        case 2:
            edge = gray.Canny(40.0, 150.0);
            edge_copy = edge.Copy();
            Display_ptbx.Image = edge.ToBitmap();
            ButtonID++;
            Process_btn.Text = "Contour";
            break;

        case 3:
            edge_copy = edge_copy.SmoothGaussian(3);
            Mat hier = new Mat();

```



```

        My_image_display = My_image_resize.Copy();
        CvInvoke.FindContours(edge_copy, Contours, hier, RetrType.List,
ChainApproxMethod.ChainApproxSimple);
        CvInvoke.DrawContours(My_image_display, Contours, -1, new
MCvScalar(0, 0, 255));
        Display_ptbx.Image = My_image_display.Bitmap;
        My_image_display = My_image_resize.Copy();
        ButtonID++;
        Process_btn.Text = "Approximate";
        break;
case 4:
    Dictionary<int, double> dict = new Dictionary<int, double>();
    if (Contours.Size > 0)
    {
        for (int i = 0; i < Contours.Size; i++)
        {
            double area = CvInvoke.ContourArea(Contours[i]);
            dict.Add(i, area);
        }
    }
    var ContourOrder = dict.OrderByDescending(v => v.Value).Take(4);

    foreach (var contourorder in ContourOrder)
    {
        int key = int.Parse(contourorder.Key.ToString());
        //CvInvoke.DrawContours(My_image_copy, Contours, key, new
MCvScalar(0, 0, 255));
        double peri = CvInvoke.ArcLength(Contours[key], true);
        VectorOfPoint approx = new VectorOfPoint();
        CvInvoke.ApproxPolyDP(Contours[key], approx, 0.015 * peri, true);
        //VectorOfVectorOfPoint approx_sub = new
VectorOfVectorOfPoint(approx);
        //CvInvoke.DrawContours(My_image_copy, approx_sub, -1, new
MCvScalar(255, 0, 0));

```

```

        if (approx.Size == 4)
        {
            screenCnt = approx;
        }
    }
    if (screenCnt.Size != 0)
    {
        VectorOfVectorOfPoint approx_sub = new
VectorOfVectorOfPoint(screenCnt);
        CvInvoke.DrawContours(My_image_display, approx_sub, -1, new
MCvScalar(255, 0, 0));

    }
    Display_ptbx.Image = My_image_display.Bitmap;
    My_image_display = My_image_resize.Copy();
    ButtonID++;
    Process_btn.Text = "Wrap";
    break;
case 5:
    if (screenCnt.Size != 0)
    {
        List<Point> RecoverCnt = new List<Point>();
        for (int i = 0; i < screenCnt.Size; i++)
        {
            Point point_temp = new Point();
            point_temp.X = Convert.ToInt16(Convert.ToDouble(screenCnt[i].X) *
ratio);
            point_temp.Y = Convert.ToInt16(Convert.ToDouble(screenCnt[i].Y) *
ratio);
            RecoverCnt.Add(point_temp);
        }
        // order would be: top-left, top-right, bottom-right, bottom-left

```

// the top-left point will have the smallest sum, the bottom-right point will have the largest sum

```
var s = new int[4];
for (int k = 0; k < 4; k++)
{
    s[k] = RecoverCnt[k].X + RecoverCnt[k].Y;
}
PointF[] rect = new PointF[4];
int maxvalue = s.Max();
rect[2] = RecoverCnt[s.ToList().IndexOf(maxvalue)];
int minvalue = s.Min();
rect[0] = RecoverCnt[s.ToList().IndexOf(minvalue)];
```

//the top-right point will have the smallest difference, the bottom - left will have the largest difference

```
var diff = new int[4];
for (int i = 0; i < 4; i++)
{
    diff[i] = RecoverCnt[i].Y - RecoverCnt[i].X;
}
minvalue = diff.Min();
rect[1] = RecoverCnt[diff.ToList().IndexOf(minvalue)];
maxvalue = diff.Max();
rect[3] = RecoverCnt[diff.ToList().IndexOf(maxvalue)];
PointF top_left = rect[0];
PointF top_rite = rect[1];
PointF bot_left = rect[3];
PointF bot_rite = rect[2];
double[] width = new double[2];
width[0] = Math.Sqrt((top_rite.X - top_left.X) * (top_rite.X - top_left.X)
+ (top_rite.Y - top_left.Y) * (top_rite.Y - top_left.Y));
width[1] = Math.Sqrt((bot_rite.X - bot_left.X) * (bot_rite.X - bot_left.X)
+ (bot_rite.Y - bot_left.Y) * (bot_rite.Y - bot_left.Y));
double[] height = new double[2];
```

```

        height[0] = Math.Sqrt((top_left.X - bot_left.X) * (top_left.X - bot_left.X)
+ (top_left.Y - bot_left.Y) * (top_left.Y - bot_left.Y));
        height[1] = Math.Sqrt((top_rite.X - bot_rite.X) * (top_rite.X - bot_rite.X)
+ (top_rite.Y - bot_rite.Y) * (top_rite.Y - bot_rite.Y));
        PointF[] dst = new PointF[4] {new PointF() {X = 0, Y = 0},
                                     new PointF() {X = Convert.ToInt16(width.Max()) - 1,
Y = 0 },
                                     new PointF() {X = Convert.ToInt16(width.Max()) - 1,
Y = Convert.ToInt16(height.Max()) - 1},
                                     new    PointF()    {X    =    0,    Y    =
Convert.ToInt16(height.Max()) - 1}};
        Mat Mm = CvInvoke.GetPerspectiveTransform(rect, dst);
        Image<Bgr, Byte> warp = new Image<Bgr,
byte>(Convert.ToInt16(width.Max()), Convert.ToInt16(height.Max()));
        CvInvoke.WarpPerspective(orig, warp, Mm, new
Size(Convert.ToInt16(width.Max()), Convert.ToInt16(height.Max())), Inter.Linear,
Warp.Default);

        warp_copy = warp.Copy();
    }
    else
        warp_copy = orig.Copy();
    //warp_copy = warp_copy.Flip(Emgu.CV.CvEnum.FlipType.Horizontal);
    Display_ptbx.Image = warp_copy.Bitmap;
    ButtonID++;
    Process_btn.Text = "Threshold";
    break;
case 6:
    Image<Gray, byte> warp_gray = warp_copy.Convert<Gray, byte>();
    warp_gray = warp_gray.ThresholdAdaptive(new Gray(255),
AdaptiveThresholdType.GaussianC, ThresholdType.Binary, 251, new Gray(10));
    Display_ptbx.Image = warp_gray.Bitmap;
    CvInvoke.Imwrite("warpped.jpg", warp_gray);
    ButtonID++;
    Process_btn.Text = "Crop";
    break;

```

```

        case 7:
            Crop("warpped.jpg", "Cropped_image.jpg", false);
            ButtonID++;
            Process_btn.Text = "Recognize";
            break;
        case 8:
            Recognize("Cropped_image.jpg", "Text.doc");
            ButtonID++;
            Process_btn.Text = "Next";
            break;
        default:
            break;
    }

}

}

private void Recognize(string path, string out_file)
{
    Image<Rgb, Byte> img = new Image<Rgb, byte>(path);
    var ocr = new TesseractEngine("./TESSDATA", "eng",
EngineMode.TesseractAndCube);
    var page = ocr.Process(img.ToBitmap());
    File.WriteAllText(out_file, page.GetText());
}

private void Crop(string path, string out_path, bool show)
{
    float downscaled_height, scale;
    int[] Crop_Info = new int[4];
    int components = 0;
    Mat orig_im = new Mat();
    Mat nonoise_im = new Mat();

```

```

orig_im = CvInvoke.Imread(path, Emgu.CV.CvEnum.ImreadModes.Grayscale);
Mat Kernel =
CvInvoke.GetStructuringElement(Emgu.CV.CvEnum.ElementShape.Rectangle, new Size(3,
3), new Point(-1, -1));
CvInvoke.Dilate(orig_im, nonoise_im, Kernel, new Point(-1, -1), 1,
Emgu.CV.CvEnum.BorderType.Default, new MCvScalar(1.0));
CvInvoke.Imwrite("No_noise.jpg", nonoise_im);
downscaled_height = 600;
scale = orig_im.Height / downscaled_height;

Mat im = new Mat();
im = Resize_User(nonoise_im, (int)downscaled_height);

Mat edges = new Mat();

CvInvoke.Canny(im, edges, 30, 200, 3);
Image<Gray, Byte> edges_1 = edges.ToImage<Gray, Byte>();
for (int i = 0; i < edges_1.Height; i++)
{
    for (int j = 0; j < edges_1.Width; j++)
    {
        if (edges_1.Data[i, j, 0] > 0)
            edges_1.Data[i, j, 0] = 255;
    }
}

Mat edges_blurred = new Mat();
CvInvoke.Imwrite("Before_MeidanBlurforFindComponets.jpg", edges);
CvInvoke.MedianBlur(edges, edges_blurred, 3);
CvInvoke.Imwrite("MeidanBlurforFindComponets.jpg", edges_blurred);
Mat Stats = new Mat();
components = find_components(edges_blurred, Stats, 10, show);

Image<Gray, Int32> Stats_11 = Stats.ToImage<Gray, Int32>();

```



```

test_print_2(Stats_11);

if (components == 0)
{
    Console.WriteLine("No text! Please change another image");
    return;
}
find_optimal_components_subset(components, edges, Stats, Crop_Info);
for (int i = 0; i < 4; i++)
{
    Crop_Info[i] = (int)((float)Crop_Info[i] * scale);
}
Image<Gray, Byte> Cropped_Image = new Image<Gray, Byte>(Crop_Info[2] -
Crop_Info[0] + 1, Crop_Info[3] - Crop_Info[1] + 1);
Image<Gray, Byte>

    Image = orig_im.ToImage<Gray, Byte>();
for (int i = 0; i < Cropped_Image.Width; i++)
{
    for (int j = 0; j < Cropped_Image.Height; j++)
    {
        Cropped_Image.Data[j, i, 0] = Image.Data[j + Crop_Info[1], i + Crop_Info[0],
0];
    }
}
//CvInvoke.Imshow("Cropped-Image", Cropped_Image);
Display_ptbx.Image = Cropped_Image.Bitmap;
CvInvoke.Imwrite(out_path, Cropped_Image);
}

private int find_components(Mat edges, Mat Stats_Array, int max_components = 10,
bool imshow = false)

```

```

{
    int count, n, Components = 0;
    //IOutputArray dilated_image = null, CCWS_Centroids_image = null,
    CCWS_labels_image = null; `
    Mat dilated_image = new Mat();
    Mat CCWS_labels_image = new Mat();
    Mat CCWS_Centroids_image = new Mat();
    //Mat CCWS_Output_Stats = new Mat();

    Mat                                Kernel                                =
    CvInvoke.GetStructuringElement(Emgu.CV.CvEnum.ElementShape.Rectangle, new Size(3,
    3), new Point(-1, -1));

    // Dilate the image until there are just a few connected components.
    count = max_components + 1;
    n = 1;
    //components = None;
    Image<Gray, Byte> edges_temp = edges.ToImage<Gray, Byte>();
    while (count > max_components)
    {
        n += 1;
        //Dont divide into 255
        CvInvoke.Dilate(edges_temp, dilated_image, Kernel, new Point(-1, -1), n,
        Emgu.CV.CvEnum.BorderType.Default, new MCvScalar(1.0));

        Components = CvInvoke.ConnectedComponentsWithStats(dilated_image,
        CCWS_labels_image, Stats_Array, CCWS_Centroids_image,
        Emgu.CV.CvEnum.LineType.EightConnected);
        count = Components;
    }
    CvInvoke.Imwrite("Edge_temp.jpg", edges_temp);
    CvInvoke.Imwrite("Dilate.jpg", dilated_image);
    Image<Gray, Byte> dilated_image_2 = dilated_image.ToImage<Gray, Byte>();
    Console.WriteLine("{0} {1} ", dilated_image_2.Height, dilated_image_2.Width);

    return Components;
}

```

```

    }

    private void find_optimal_components_subset(int components, Mat edges, Mat Stats,
int[] Crop_Info)
    {

        float total = 0;
        Image<Gray, Byte> edges_image = edges.ToImage<Gray, Byte>();
        Image<Gray, Int32> Stats_image = Stats.ToImage<Gray, Int32>();
        test_print_2(Stats_image);
        for (int i = 0; i < edges_image.Height; i++)
        {
            for (int j = 0; j < edges_image.Width; j++)
            {
                total = total + edges_image.Data[i, j, 0];
            }
        }
        total = total / 255;//sum of whites pixel in entire image

        //Compute sum of whites pixel each block

        for (int each_block = 0; each_block < Stats_image.Height; each_block++)
        {
            int sum_temp = 0;
            for (int i = Stats_image.Data[each_block, 0, 0]; i <=
Stats_image.Data[each_block, 0, 0] + Stats_image.Data[each_block, 2, 0] - 1; i++)
            {
                for (int j = Stats_image.Data[each_block, 1, 0]; j <=
Stats_image.Data[each_block, 1, 0] + Stats_image.Data[each_block, 3, 0] - 1; j++)
                {
                    sum_temp = sum_temp + edges_image.Data[j, i, 0] / 255;
                }
            }
        }
    }

```

```

Stats_image.Data[each_block, 4, 0] = sum_temp;
}

//Sort Array
int temp_sort = 0;
int repeat_sort = 0;
do
{
    repeat_sort = 0;
    for (int each_block = 0; each_block < Stats_image.Height - 1; each_block++)
    {
        if (Stats_image.Data[each_block, 4, 0] < Stats_image.Data[each_block + 1, 4,
0])
        {
            for (int n = 0; n < Stats_image.Width; n++)
            {
                temp_sort = Stats_image.Data[each_block, n, 0];
                Stats_image.Data[each_block, n, 0] = Stats_image.Data[each_block + 1,
n, 0];
                Stats_image.Data[each_block + 1, n, 0] = temp_sort;
            }
            repeat_sort++;
        }
    }

} while (repeat_sort > 0);

test_print_2(Stats_image);

//Union crops
int x1_crop = Stats_image.Data[1, 0, 0];

```

```

int y1_crop = Stats_image.Data[1, 1, 0];
int x2_crop = Stats_image.Data[1, 2, 0] + x1_crop - 1;
int y2_crop = Stats_image.Data[1, 3, 0] + y1_crop - 1;

int covered_sum = Stats_image.Data[1, 4, 0];
int[] Array_Crop = { x1_crop, y1_crop, x2_crop, y2_crop };
for (int each_block = 2; each_block < Stats_image.Height; each_block++)
{
    int[] Array_Crop_Proposal = { Stats_image.Data[each_block, 0, 0],
Stats_image.Data[each_block, 1, 0], Stats_image.Data[each_block, 2, 0] +
Stats_image.Data[each_block, 0, 0] - 1, Stats_image.Data[each_block, 3, 0] +
Stats_image.Data[each_block, 1, 0] - 1 };

    float temp_1 = covered_sum / total;
    if (temp_1 < 0.9)
    {
        Array_Crop[0] = Math.Min(Array_Crop[0], Array_Crop_Proposal[0]);
        Array_Crop[1] = Math.Min(Array_Crop[1], Array_Crop_Proposal[1]);
        Array_Crop[2] = Math.Max(Array_Crop[2], Array_Crop_Proposal[2]);
        Array_Crop[3] = Math.Max(Array_Crop[3], Array_Crop_Proposal[3]);
        covered_sum = covered_sum + Stats_image.Data[each_block, 4, 0];
    }
    else
        break;
}
for (int i = 0; i < 4; i++)
{
    Crop_Info[i] = Array_Crop[i];
}
}

private Mat Resize_User(Mat Image_User, int Height_Resized = 0, int Width_Resized
= 0)
{
    Mat Image_Resized = new Mat();

```

```

int h_user;
int w_user;
float r = 0;
Size dim_Resized = new Size(Width_Resized, Height_Resized);
h_user = Image_User.Height;
w_user = Image_User.Width;
if (Width_Resized == 0 && Height_Resized == 0)
    return Image_User;
if (Width_Resized == 0)
{
    r = (float)Height_Resized / h_user;
    r = r * (float)w_user;
    dim_Resized = new Size((int)(r + 1), Height_Resized);
}
else
{
    r = (float)Width_Resized / w_user;
    r = r * (float)h_user;
    dim_Resized = new Size(Width_Resized, (int)(r + 1));
}

CvInvoke.Resize(Image_User, Image_Resized, dim_Resized, 0, 0,
Emgu.CV.CvEnum.Inter.Linear);
return Image_Resized;
}

private void test_connectedComponentsWithStats()
{
    Mat frame = new Mat();
    Mat Output_Label = new Mat();
    Mat Output_Stats = new Mat();
    Mat Output_Centroid = new Mat();

```



```

        string path =
"C:\\Users\\Asus\\Desktop\\Text_recognize\\Text_recognize\\bin\\Debug\\Edged_dilated.jpg";
        frame = CvInvoke.Imread(path, Emgu.CV.CvEnum.ImreadModes.AnyColor);
        CvInvoke.ConnectedComponentsWithStats(frame, Output_Label, Output_Stats,
Output_Centroid, Emgu.CV.CvEnum.LineType.EightConnected);
        Console.WriteLine(Output_Stats.Height);
        Console.WriteLine(Output_Stats.Width);

        Image<Gray, Byte> Output_Stats_im = Output_Stats.ToImage<Gray, Byte>();

        Console.ReadLine();

    }

    private void test_print(Image<Gray, Byte> frame)
    {
        for (int i = 0; i < frame.Height; i++)
        {
            for (int j = 0; j < frame.Width; j++)
            {
                Console.Write("{0} ", frame.Data[i, j, 0]);
            }
            Console.WriteLine();
        }
    }

    private void test_print_2(Image<Gray, Int32> frame)
    {
        for (int i = 0; i < frame.Height; i++)
        {
            for (int j = 0; j < frame.Width; j++)
            {
                Console.Write("{0} ", frame.Data[i, j, 0]);
            }
        }
    }

```

```
    }  
    Console.WriteLine();  
    }  
    }  
    }  
}
```