1. Підготовка структури роботи

Спершу установимо EmguCV. Для цього після створення проекту ПКМ на рішення у вікні «Оглядач рішень»→Керування пакетами NuGet. Потім встановлюємо наступну обгортку OpenCV для C#.

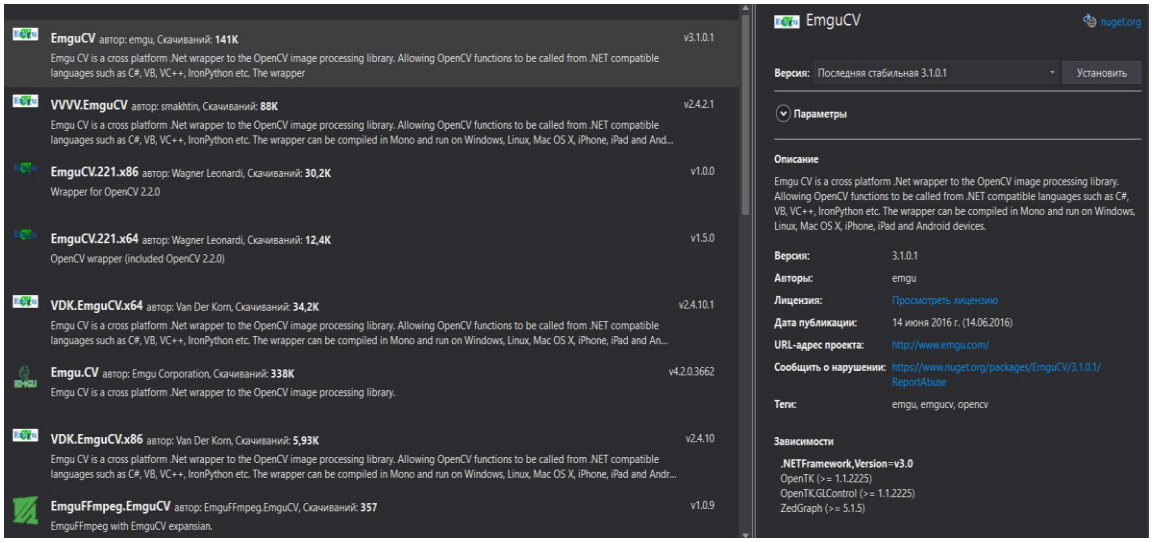

Рис. 1. Установлення EmguCV

Тепер потрібно перемістити папку HaarCascade з файлами .xml в проект для огляду файлів, а також в папку Debug даного проекту для швидкого пошуку файлів проектом. Для змоги оглядати файли безпосередньо в VS2019 можна натиснути ЛКМ на цю папку і тримати її, пересуваючи в «Оглядач Рішень».

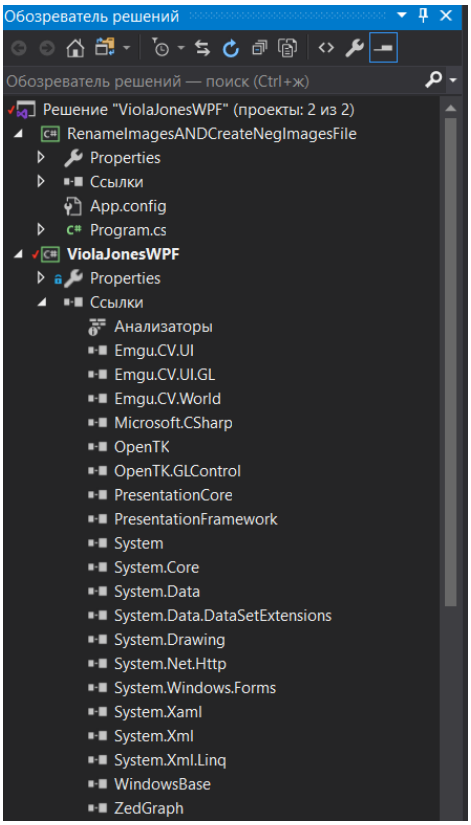

Рис. 2. «Оглядач Рішень»

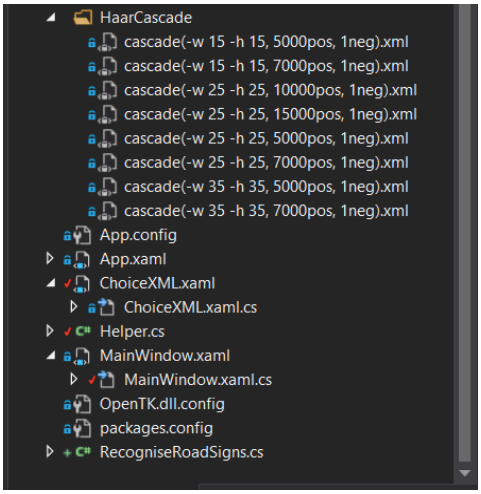

Рис. 3. Продовження

Файл ChoiceXML.xaml – діалогове вікно для вибору параметрів каскаду.

2. Створення інтерфейсу форми

Для нашого додатку нам потрібні наступні елементи:

- 2 Grid-и. Grid – найчастіший елемент у використанні. Містить стовпці й рядки.
  1) Перший зовнішній. Призначений для компоновки усіх елементів;
  2) другий відповідно внутрішній. Існує для компоновки кнопок. Без нього кнопки будуть лишатися на одному тому ж місці при збільшенні розміру вікна відповідно до заданого програмно;
- 1 Menu – відображає команди програми та параметри, згруповані за функціональними можливостями смужка. Містить 4 MenuItem. 1 зовнішній під назвою File та 3 внутрішній (їх видно, якщо натиснути ЛКМ на зовнішній) з різними впливовими властивостями:
  1) `Click="SelectImage_Click" x:Name="SelectImage" Header="Select an image".` Ця кнопка потрібна для вибору зображення, яке буде відображатись у елементі Image;
  2) `Click="SelectTestSet_Click" x:Name="SelectTestSet" Header="Select a test set",` необхідна для вибору тестового набору даних для визначення точності.
  3) `Click="SelectXML_Click" x:Name="SelectXML" Header="Select a file .xml"` для відображення діалогового вікна, де користувач має змогу вибрати .xml-файл;
- 1 Image – елемент, який призначений для показу зображень. Тут створений ефект `DropShadowEffect` призначений для тіні зображення властивість `BlurRadius` задає розмиття, чим більше у нього значення тим більший розмитий колір, який задається у властивості `Color`;
- 1 Border – рамка, в межах якої буде відображатись зображення. У даному випадку вона призначена лише ясності, де буде відображатись зображення.

- 2 Button (кнопка). Одна для розпізнавання дорожніх знаків, друга - для обчислення точності.

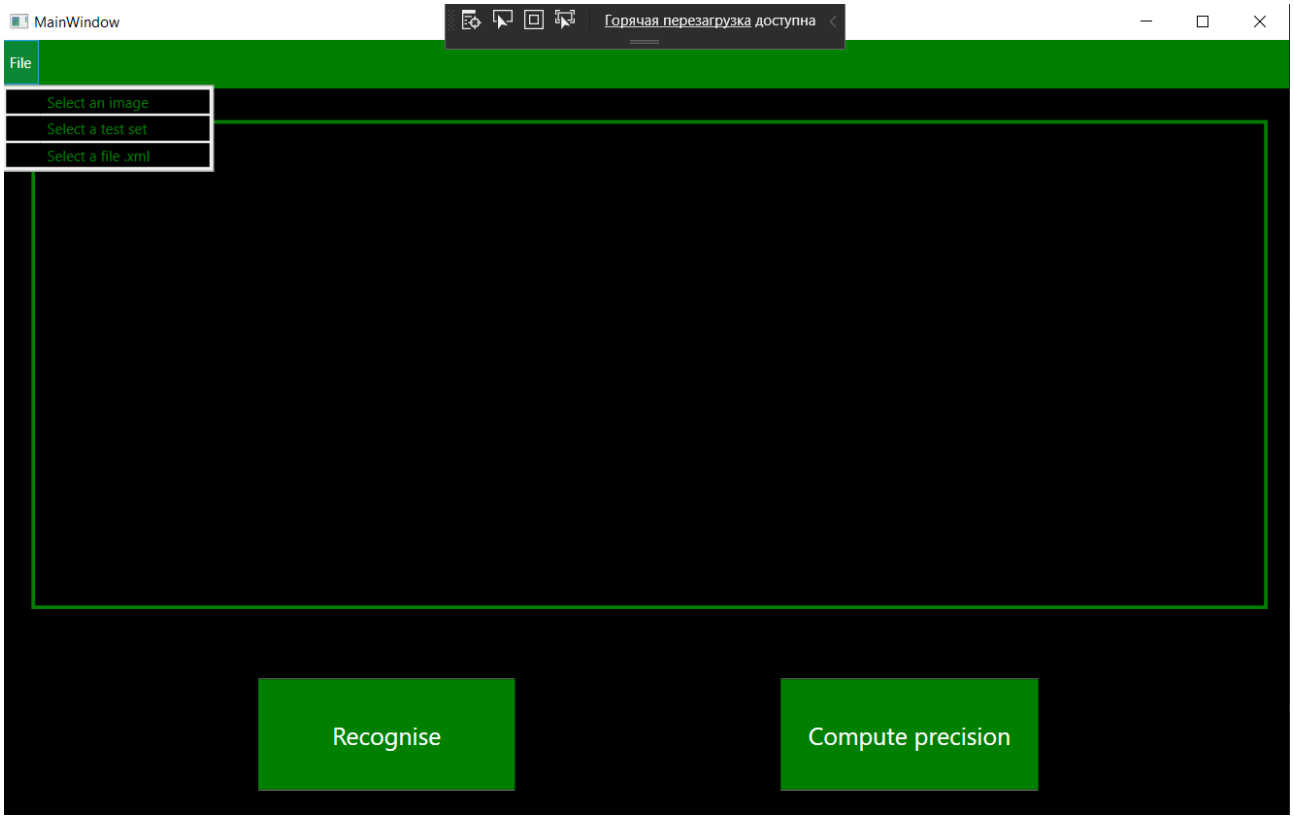

Рис. 4. Інтерфейс головної форми

3. Створення інтерфейсу діалогового вікна

Це діалогове вікно існує для вибору користувачем параметрів каскаду. Воно виглядає наступним чином, властивість Name усіх RadioButton, які використовуються для вибору каскад з «позитивними» зображеннями, задавались у форматі “RB_<висота_зразків>_<кількість_зразків>”, а назва RadioButton для каскаду з «позитивними» та «негативними» зображенням у вигляді“RB_<висота_зразків>_<кількість_«позитивних»_зразків>_<кількість_«негативних»_зразків>”. В тезі Window задані властивості:

- `WindowStartupLocation="CenterScreen"` – вікно з’являється посередині екрану;
- `ResizeMode="NoResize"` – користувач не має змоги змінити розмір вікна

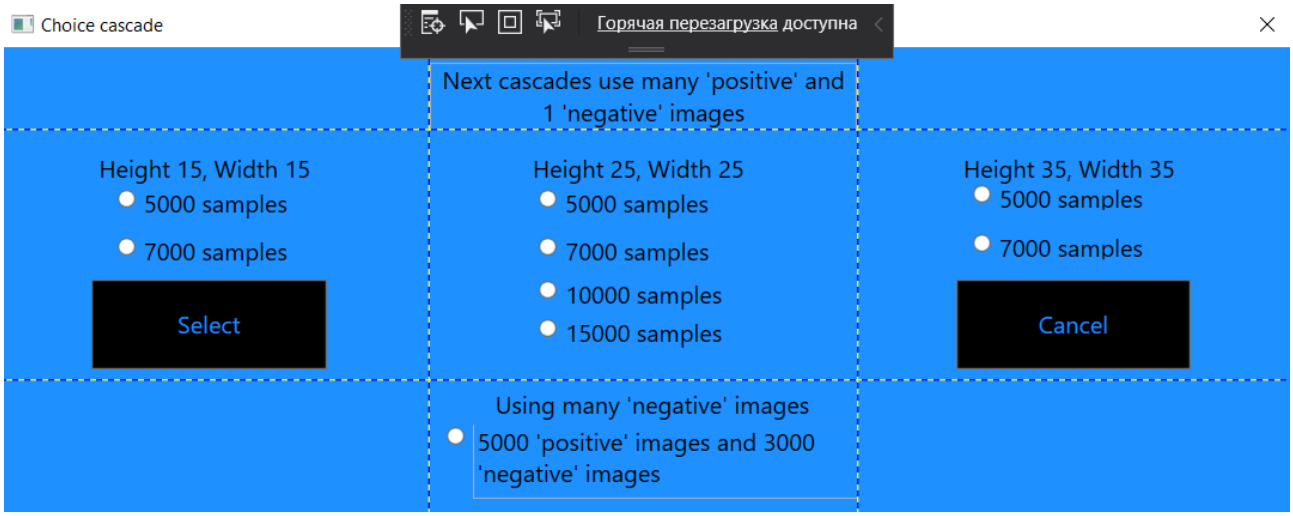

Рис. 5. Інтерфейс діалогового вікна

XAML код діалогового вікна:

```
<Window x:Class="ViolaJonesWPF.ChoiceXML"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ViolaJonesWPF"
        mc:Ignorable="d"
        Title="Choice cascade" Height="349.243" Width="878.428" Background="DodgerBlue"
WindowStartupLocation="CenterScreen" ResizeMode="NoResize" Closing="Window_Closing">
    <Grid Margin="0,0,4,-0.2" ShowGridLines="True">
        <Grid.RowDefinitions>
            <RowDefinition Height="55"/>
            <RowDefinition/>
            <RowDefinition Height="90"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <TextBox FontSize="16" Background="DodgerBlue" Foreground="Black"
TextAlignment="Center" TextWrapping="WrapWithOverflow" Grid.Column="1"
Margin="0.4,10,0.2,0.2">
            Next cascades use many 'positive' and 1 'negative' images
        </TextBox>
        <Label Grid.Row="1" Grid.Column="0" Content="Height 15, Width 15"
HorizontalAlignment="Left" Margin="61,9.8,0,0" VerticalAlignment="Top" FontSize="16"/>
        <Label Grid.Row="1" Grid.Column="1" Content="Height 25, Width 25"
HorizontalAlignment="Left" Margin="64.4,9.8,0,0" VerticalAlignment="Top" Width="152"
FontSize="16"/>
        <Label Grid.Row="1" Grid.Column="2" Content="Height 35, Width 35"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="65.8,9.8,0,0" FontSize="16"/>
        <RadioButton Grid.Row="1" Grid.Column="0" Checked="RB_15_5000_Checked"
x:Name="RB_15_5000" Content="5000 samples" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="77,38.8,0,0" FontSize="16"/>
        <RadioButton Grid.Row="1" Grid.Column="0" Checked="RB_15_7000_Checked"
x:Name="RB_15_7000" Content="7000 samples" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="77,70.8,0,0" FontSize="16"/>
        <RadioButton Grid.Row="1" Grid.Column="1" Checked="RB_25_5000_Checked"
x:Name="RB_25_5000"  Content="5000 samples" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="72,38.8,0,0" FontSize="16"/>
        <RadioButton Grid.Row="1" Grid.Column="1" Checked="RB_25_7000_Checked"
x:Name="RB_25_7000" Content="7000 samples" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="72.4,70.8,0,0" FontSize="16" />
```

```
        <RadioButton Grid.Row="1" Grid.Column="1" Checked="RB_25_10000_Checked"
x:Name="RB_25_10000" Content="10000 samples" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="72.4,99.8,0,0" FontSize="16" />
        <RadioButton Grid.Row="1" Grid.Column="1" Checked="RB_25_15000_Checked"
x:Name="RB_25_15000" Content="15000 samples" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="72.4,125.8,0,0" FontSize="16" />
        <RadioButton Checked="RB_35_5000_Checked" x:Name="RB_35_5000" Content="5000 samples"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="75.8,35.8,0,0" Grid.Column="2"
Height="18" Grid.Row="1" Width="117" FontSize="16"/>
        <RadioButton Checked="RB_35_7000_Checked" x:Name="RB_35_7000" Content="7000 samples"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="75.8,68.8,0,0" Grid.Column="2"
Height="18" Grid.Row="1" Width="117" FontSize="16"/>
        <Button Background="Black" Foreground="DodgerBlue" Click="Select_Click"
Content="Select" HorizontalAlignment="Left" Margin="61,99.8,0,0" VerticalAlignment="Top"
Width="157" Height="60" Grid.Row="1" FontSize="16"/>
        <Label Content="Using many 'negative' images" HorizontalAlignment="Left"
Margin="39.4,0,0,0" VerticalAlignment="Top" Height="42" Grid.Row="2" Width="269"
Grid.Column="1" FontSize="16" Grid.ColumnSpan="2"/>
        <RadioButton Checked="RB_25_5000_3000_Checked" x:Name="RB_25_5000_3000"
HorizontalAlignment="Left" Margin="10.4,30,0,0" VerticalAlignment="Top" Height="50"
Grid.Row="2" Width="279" Grid.Column="1" FontSize="16">
            <TextBox Background="DodgerBlue" Foreground="Black" TextWrapping="Wrap"
Width="258" Height="50" FontSize="16">5000 'positive' images and 3000 'negative' images
            </TextBox>
        </RadioButton>
        <Button Click="Cancel_Click" Background="Black" Foreground="DodgerBlue"
Content="Cancel" HorizontalAlignment="Left" Margin="65.8,99.8,0,0" Grid.Row="1"
VerticalAlignment="Top" Width="157" Height="60" Grid.Column="2" FontSize="16"/>
    </Grid>
</Window>
```

4. Реалізація класу RecogniseRoadSigns

Цей клас містить методи для:

а) зчитування даних з файлу, який містить опис тестового набору зображень та отримання істинних даних про дорожні знаки.

Лістинг 1. Метод для отримання істинних даних про дорожні знаки. Повертаюче значення вказує, чи знайдено в цьому наборі даних з назвою nameFile поточне зображення з іменем nameImage. Змінна amountSign вказує, скільки зображень міститься на зображенні, що потрібне для подальшої роботи, і саме тому має модифікатор out. Використовується List<>, а не масив, оскільки нам невідомо скільки є дорожніх знаків на зображенні.

```
        public Boolean ReadTextFile(String nameFile, String nameImage, out Int32 amountSign,
List<String> data)
                {
                    using (StreamReader sr = new StreamReader(nameFile))
                    {
                        String shortNameFile;
                        String[] arrayData;
                        Boolean rightRead = false;

                        amountSign = 0;
                        while (sr.Peek() >= 0)
                        {
                            String textRow = sr.ReadLine();
                            shortNameFile = "";

                            Int32 i = textRow.IndexOf(';') - 1;
                            for (Int32 j = 0; j <= i; j++)
                            {
```

```
                    shortNameFile += textRow[j];
                }

                if (nameImage == shortNameFile)
                {
                    rightRead = true;
                }
                while (rightRead)
                {
                    arrayData = new String[6];
                    arrayData = textRow.Split(';');

                    data.Add(arrayData[1]);
                    data.Add(arrayData[2]);
                    data.Add(arrayData[3]);
                    data.Add(arrayData[4]);

                    amountSign++;
                    if (sr.Peek() >= 0)
                    {
                        textRow = sr.ReadLine();
                        i = textRow.IndexOf(';') - 1;
                        shortNameFile = "";
                        for (Int32 j = 0; j <= i; j++)
                        {
                            shortNameFile += textRow[j];
                        }
                        if (nameImage != shortNameFile)
                        {
                            return rightRead;
                        }
                    }

                }
            }
            return rightRead;
        }
    }
```

Клас IO.StreamReader служить для спрощення операцій зчитування текстових файлів. При використанні конструкції using не потрібно використовувати метод StreamReader.Close() для закриття об’єкта типу StreamReader і звільнення всіх системних ресурсів асоційованих з даним потоком [1].

Функція Peek() повертає наступний символ у файлі, але не переміщає поточну позицію на нього. Зазвичай використовується для перевірки, чи повністю зчитаний файл.

ReadLine() повертає рядок з файлу, переводить курсив на наступний рядок [1].

b) перевірки на правильність запису цього файлу.

Лістинг 2. Цей метод нічого не повертає. Параметр типу List<String> повинен містити дані про зображення. Ці дані мають братися з файлу про тестовий набір даних. Другий параметр типу Int32 вказує на кількість істинних дорожніх знаків.

```
public void IsRightRecord(List<String> data, Int32 amountSign)
        {
            topRow = new Int32[amountSign];
```

```
            leftCol = new Int32[amountSign];
            bottomRow = new Int32[amountSign];
            rightCol = new Int32[amountSign];

            for (Int32 i = 0, k = 0; i < amountSign * 4 && k < amountSign; i++, k++)
            {
                if (!Int32.TryParse(data[i], out topRow[k]))
                {
                    throw new FormatException("Upper X coordinate could not be read from the
test data set");
                }
                else if (!Int32.TryParse(data[++i], out leftCol[k]))
                {
                    throw new FormatException("Left Y coordinate could not be read from the
test data set");
                }
                else if (!Int32.TryParse(data[++i], out bottomRow[k]))
                {
                    throw new FormatException("Lower X coordinate could not be read from the
test data set");
                }
                else if (!Int32.TryParse(data[++i], out rightCol[k]))
                {
                    throw new FormatException("Right Y coordinate could not be read from the
test data set");
                }
            }
        }
```

c) обчислення точності розпізнавання.

Лістинг 3. Метод відповідно повертає точність. Містить параметр з модифікатором out, тобто в результаті цього методу буде повернена змінна precision, яка відповідає за точність розпізнавання. Другий параметр типу Int32 вказує на кількість істинних дорожніх знаків.

```
        public Double DetermPrecision(out Int32 wrongCountSigns, Int32 amountSign)
                {
                    Int32 amountRightRect = 0;

                    for (Int32 i = 0; amountSign >= roadSignsRecog.Length ?
                        i < roadSignsRecog.Length : i < amountSign; i++)
                    {
                        Double fault = (Double)(bottomRow[i] - topRow[i]) * 0.1;

                        if (Math.Abs(roadSignsRecog[i].Top - topRow[i]) <= fault &&
Math.Abs(roadSignsRecog[i].Left - leftCol[i]) <= fault &&
                        Math.Abs(roadSignsRecog[i].Bottom - bottomRow[i]) <= fault &&
Math.Abs(roadSignsRecog[i].Right - rightCol[i]) <= fault)
                        {
                            amountRightRect++;
                        }

                    }

                    Double precision = (Double)(amountRightRect / amountSign * 100.0);
                    if(amountSign >= roadSignsRecog.Length)
                    {
                        wrongCountSigns = roadSignsRecog.Length - amountRightRect;
                    }
                    else
                    {
                        wrongCountSigns = 0;
                    }
```

```
                return precision;
                   }
```

d) обведення областей, де ймовірно розташовані дорожні знаки.

Лістинг 4. Приклад створення схожого методу див. додаток А. Також пояснення цього методу можна знайти за посиланням [2].

```
public Double RecSigns(System.Windows.Controls.Image pictureBox, Image<Bgr, Byte> image,
Double scaleFactor, String fileXML, Int32 minNeighbors)
        {
            try
            {
                CascadeClassifier cascadeClassifier = new CascadeClassifier(fileXML);

                Bitmap bitmap = image.ToBitmap();
                Image<Bgr, Byte> grayImage = new Image<Bgr, Byte>(bitmap);

                DateTime start = new DateTime();
                start = DateTime.Now;
                roadSignsRecog = cascadeClassifier.DetectMultiScale(grayImage, scaleFactor,
minNeighbors);
                TimeSpan ms = DateTime.Now.Subtract(start);

                foreach (Rectangle roadSign in roadSignsRecog)
                {
                    using (Graphics graphics = Graphics.FromImage(bitmap))
                    {
                        using (Pen pen = new Pen(Color.Blue, 4))
                        {
                            graphics.DrawRectangle(pen, roadSign);
                        }
                    }
                }
                grayImage.Bitmap = bitmap;
                pictureBox.Source = Helper.LoadBitmap(bitmap);

                return ms.TotalMilliseconds;
            }
            catch (Exception ex)
            {
                throw new Exception(ex.Message);
            }
           }
```

CascadeClassifier - клас, який містить метод DetectMultiScale. Опис цього методу міститься тут [3].

Для обчислення часу виявлення областей використовується структура DateTime. Саме вона вважається однією з найбільш точних способів для виявлення часу виконання рядків коду. Увесь код цього класу розташований у додатку Б.

5. Реалізація коду головного вікна

Лістинг 5. Варто зробити акцент на обробник події натискання на пункт SelectImage зі смужки меню, оскільки нам потрібно зчитувати файли будь-якого формату зображень та сам WPF робить непростим завантаження у контейнер Image. Решту подій не мають особливостей. Увесь код основного вікна можна подивитись у додатку Б.

```
private void SelectImage_Click(object sender, RoutedEventArgs e)
        {
            openImage = new OpenFileDialog();
            findImage = openImage.ShowDialog();
            if (findImage == true)
            {
                image = new Image<Bgr, Byte>(openImage.FileName);
                pictureBox1.Source = Helper.LoadBitmap(image.ToBitmap());
            }
         }
```

Лістинг 6. Реалізація класу Helper. Взята з [4]

```
public class Helper
    {
        [System.Runtime.InteropServices.DllImport("gdi32.dll")]
        private static extern Boolean DeleteObject(IntPtr handle);

        public static BitmapSource LoadBitmap(Bitmap source)
        {
            IntPtr intPtr = source.GetHbitmap();
            BitmapSource bitmapSource =
System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap(intPtr, IntPtr.Zero,
                Int32Rect.Empty, BitmapSizeOptions.FromEmptyOptions());
            DeleteObject(intPtr);

            return bitmapSource;
        }
       }
```

Клас-атрибут DllImport включає атрибутний метод, який представлений некерованою динамічною бібліотекою(dynamic-link library (DLL)) в даному випадку метод DeleteObject, який є зовнішнім (extern), міститься в "gdi32.dll". Метод System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap описується тут [5].

6. Реалізація коду діалогового вікна

При виборі певного RadioButton змінюється статична змінна fileXML і minNeighbors в класі RoadSignsRecognise. fileXML використовується, як назва .xml-файлу і передається у параметр конструктора класу CascadeClassifier. Змінна minNeighbors потрібна для різного групування сусідніх прямокутників через різні розмірності зразків та їхньої кількості (додаток В).

Джерела

1. Стаття про роботу з класом StreamReader: https://programming-lessons.xyz/c-sharp/chtenie-tekstovyh-fajlov-klass-streamreader/
2. Відео-пояснення методу RecSigns: https://www.youtube.com/watch?v=IBG5IQQGHg4&list=PLH3y3SWteZd3LSZ0N3_dGnY3n2hYjLn7b
3. Стаття про метод CascadeClassifier.DetectMultiScale: http://www.emgu.com/wiki/files/2.4.2/document/html/944b73ce-a95c-5547-3651-12b691fdeb46.htm
4. Як загрузити в контейнер Image зображення у WPF: https://stackoverflow.com/questions/31607014/bitmap-graphics-createbitmapsourcefromhbitmap-memory-leak
5. Опис методу CreateBitmapSourceFromHBitmap: https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.interop.imaging.createbitmapsourcefromhbitmap?view=netcore-3.1.

Додатки

Додаток А

```
private void faceRecognize()
{
    string name;
    //take a frame
    using (var imageFrame = capture.QueryFrame().ToImage<Bgr, byte>())
    {
        if (imageFrame != null)
        {
            //convert to gray to improve the prediction
            var grayFrame = imageFrame.Convert<Gray, byte>();

            //Finds rectangular regions (face); the second param its the scale > 1 slowest
process but accurated prediction
            var faces = cascadeClassifier.DetectMultiScale(grayFrame, 1.3, 6, new
Size((grayFrame.Size.Width / 4), (grayFrame.Size.Height / 4)));

            //Predict the frame taked. If theres a sample of the face in db it will return
the username, if not will say John Doe
            name = (facePredict(grayFrame) > 0) ? da.GetUsername(facePredict(grayFrame)) :
"John Doe";

            foreach (var face in faces)
            {
                //draw a box at the face
                imageFrame.Draw(face, new Bgr(Color.Green), 2);
                //put text bellow the box
                CvInvoke.PutText(imageFrame, name, new Point(face.Location.X + 10,
face.Location.Y - 10), Emgu.CV.CvEnum.FontFace.HersheyComplex, 1.0, new Bgr(0, 255,
0).MCvScalar);
            }
            imageBox1.Image = imageFrame;
        }
    }
}
private void button1_Click(object sender, EventArgs e)
{
    userPicture = capture.QueryFrame().ToImage<Gray, byte>();

    //Finds rectangular regions (face)
    var faces = cascadeClassifier.DetectMultiScale(userPicture, 1.3, 6, new
Size((userPicture.Size.Width / 4), (userPicture.Size.Height / 4)));

    foreach (var face in faces)
    {
        //resize sample
        faceImageThumb = userPicture.Copy(face).Resize(64, 64, Emgu.CV.CvEnum.Inter.Cubic);
    }

    //add to db and notify user
    MessageBox.Show(da.AddSample(textBox1.Text, ConvertImageToByte(faceImageThumb)), "Face",
MessageBoxButtons.OK);
}
```

```
using Emgu.CV;
using Emgu.CV.Structure;
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows;

namespace ViolaJonesWPF
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>

    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        internal static String FileXML { get; set; }
        internal static Double ScaleFactor { private get; set; }
        internal static Int32 MinNeighbors { private get; set; }

        private readonly RecogniseRoadSigns recognise = new RecogniseRoadSigns();

        private OpenFileDialog openImage, openDataSet;
        private Image<Bgr, Byte> image;

        private void SelectImage_Click(object sender, RoutedEventArgs e)
        {
            openImage = new OpenFileDialog();
            Boolean? findImage = openImage.ShowDialog();
            if (findImage == true)
            {
                image = new Image<Bgr, Byte>(openImage.FileName);
                pictureBox1.Source = Helper.LoadBitmap(image.ToBitmap());
            }
        }

        private void CompulatePrecision_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                FileInfo file = new FileInfo(openImage.FileName);
                Int32 amountSign;

                List<String> data = new List<String>();
                if (openImage.FileName != "" && openDataSet.FileName != "")
                {
                    Boolean rightRead = recognise.ReadTextFile(openDataSet.FileName,
file.Name, out amountSign, data);
                    if (rightRead == false)
                    {
                        MessageBox.Show("Тестовий набір даних неправильно записаний",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
                        return;
                    }
                }
                else if (openDataSet.FileName == "")
                {
                    MessageBox.Show("Виберіть, будь ласка, тестовий набір даних", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
```

```
                }
                else
                {
                    return;
                }

                recognise.IsRightRecord(data, amountSign);
                Int32 wrongCountSigns;
                Double precision = recognise.DetermPrecision(out wrongCountSigns,
amountSign);
                if (wrongCountSigns != 0)
                {
                    MessageBox.Show($"Точніть розпізнавання: {precision}%", "Точніть
розпізнавання", MessageBoxButton.OK, MessageBoxImage.Information);
                }
                else
                {
                    MessageBox.Show($"Точніть розпізнавання: {precision}%. Обведено зайво
{wrongCountSigns} областей",
                        "Точніть розпізнавання", MessageBoxButton.OK,
MessageBoxImage.Information);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Помилка", MessageBoxButton.OK,
MessageBoxImage.Error);
            }
        }

        private void SelectTestSet_Click(object sender, RoutedEventArgs e)
        {
            openDataSet = new OpenFileDialog();
            openDataSet.Filter = "TXT|*.txt";
            openDataSet.ShowDialog();
        }

        private void SelectXML_Click(object sender, RoutedEventArgs e)
        {
            ChoiceXML dialogXML = new ChoiceXML();
            dialogXML.ShowDialog();
        }

        private void Recognise_Click(object sender, RoutedEventArgs e)
        {
            Double timeRecog;
            try
            {
                timeRecog = recognise.RecSigns(pictureBox1, image, ScaleFactor, FileXML,
MinNeighbors);
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message, "Помилка", MessageBoxButton.OK,
MessageBoxImage.Error);
                return;
            }
            MessageBox.Show($"Час розпізнавання об\'єктів: {timeRecog} мілісекунд", "Час
розпізнавання", MessageBoxButton.OK, MessageBoxImage.Information);
        }
    }
}
```

```
using System.Windows;

namespace ViolaJonesWPF
{
    /// <summary>
    /// Логика взаимодействия для ChoiceXML.xaml
    /// </summary>
    public partial class ChoiceXML : Window
    {
        public ChoiceXML()
        {
            InitializeComponent();
        }

        private void RB_15_5000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 15 -h 15, 5000pos, 1neg).xml";
        }

        private void RB_15_7000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 15 -h 15, 7000pos, 1neg).xml";
        }

        private void RB_25_5000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 25 -h 25, 5000pos, 1neg).xml";
        }

        private void RB_25_7000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 25 -h 25, 7000pos, 1neg).xml";
        }

        private void RB_25_10000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 25 -h 25, 10000pos, 1neg).xml";
        }

        private void RB_25_15000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 25 -h 25, 15000pos, 1neg).xml";
        }

        private void RB_35_5000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 35 -h 35, 5000pos, 1neg).xml";
        }

        private void RB_35_7000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 35 -h 35, 7000pos, 1neg).xml";
        }

        private void RB_25_5000_3000_Checked(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = @"HaarCascade\cascade(-w 25 -h 25, 5000pos, 3000neg).xml";
        }

        private void Select_Click(object sender, RoutedEventArgs e)
        {
            if(!MainWindow.FileXML.Contains("3000neg"))
            {
                MainWindow.MinNeighbors = 43;
                MainWindow.ScaleFactor = 1.1;
```

```
            }
            else
            {
                MainWindow.MinNeighbors = 6;
                MainWindow.ScaleFactor = 1.3;
            }
            Close();
        }

        private void Cancel_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.FileXML = "";
            Close();
        }

        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            MainWindow.FileXML = "";
        }
    }
}
```