Capítulo 4. Implementación del Sistema

Para el desarrollo del prototipo se utilizó el lenguaje C# de la plataforma *Visual Studio .NET*, a continuación se mostrarán las partes de código más importantes.

4.1 Librerías Utilizadas

Se descargaron las librerías AForge .NET y OpenCV y se agregaron al proyecto (Figura 10).

- http://www.aforgenet.com/
- http://opencv.org/

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using System.IO;
using System.Diagnostics;
using AForge.Video.VFW;
```

Figura 10. Librerías utilizadas para el desarrollo del Proyecto.

4.2 Instanciación de objetos y Carga de Etiquetas

Se instancia el objeto *face* del tipo HaarCascade, donde se carga el archivo en formato XML con las características *Haar-Like* las cuales permitirán realizar la detección facial, se instancian los objetos *AVIWriter* donde se almacenará el video en formato AVI (Figura 11).

```
//Load haarcascades for face detection
face = new HaarCascade("haarcascade_frontalface_default.xml");

// instantiate AVI writer, use WMV3 codec
writer = new AVIWriter("wmv3");

// create new AVI file and open it
writer.Open(Application.StartupPath + "/VideoGrabber/video.avi",
imageBoxFrameGrabber.Size.Width, imageBoxFrameGrabber.Size.Height);
            // instantiate AVI writer, use WMV3 codec
writerOptimized = new AVIWriter("wmv3");
// create new AVI file and open it
writerOptimized.Open(Application.StartupPath + "/VideoGrabber/optimizedVideo.avi",
imageBoxSalida.Size.Width, imageBoxSalida.Size.Height);
```

Figura 11. Instanciación de objetos, detección facial y almacenamiento en video.

Para agilizar la carga de etiquetas de la base de conocimiento, se utilizó un archivo en formato TXT, cada etiqueta está separada por un símbolo especial (Figura 12).

```
try
{
    //Load of previous trainned faces and labels for each image
    string Labelsinfo = File.ReadAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt");
    string[] Labels = Labelsinfo.Split('%');
    NumLabels = Convert.ToInt16(Labels[0]);
    ContTrain = NumLabels;
    string LoadFaces;

    for (int tf = 1; tf < NumLabels+1; tf++)
    {
        LoadFaces = "face" + tf + ".bmp";
        trainingImages.Add(new Image<Gray, byte>(Application.StartupPath + "/TrainedFaces/" + LoadFaces));
        labels.Add(Labels[tf]);
    }
}
catch(Exception e)
{
    MessageBox.Show("La base de conocimiento está vacia, agregar un rostro", "Trained faces load",
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
```

Figura 12. Carga de base de conocimiento.

4.3 Reconocimiento Facial e Implementación de Criterios de Almacenamiento

Se obtiene el fotograma capturado por la cámara, para realizar la detección facial se hace un redimensionamiento (320 x 240 pixeles) del fotograma de entrada para mejorar el rendimiento del prototipo, ya que el tiempo de procesamiento es directamente proporcional al tamaño en pixeles de la imagen de entrada. Posteriormente, se convierte el fotograma redimensionado a escala de grises y se aplican las características de *Haar* para realizar la detección facial, si en el fotograma se detecta un rostro, se dibuja un cuadro rojo para enmarcar el rostro y se inicia el proceso de identificación facial a través de los *eigenfaces*, si el rostro es identificado se dibuja la etiqueta asociada a ese rostro. Finalmente se asignan los criterios de almacenamiento con base al resultado del proceso anterior (reconocimiento facial) y se inicia el proceso para enviar la secuencia de fotogramas para su almacenamiento.

```
//Get the current frame form capture device
currentFrame      =      grabber.QueryFrame().Resize(imageBoxFrameGrabber.Size.Width,
imageBoxFrameGrabber.Size.Height, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

//Convert it to Grayscale
gray = currentFrame.Convert<Gray, Byte>();

//Face Detector
MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
                  face,
                  1.2,
                  10,
                  Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
                  new Size(20, 20));

foreach (MCvAvgComp f in facesDetected[0])
{
    t = t + 1;
    result   =   currentFrame.Copy(f.rect).Convert<Gray,   byte>().Resize(100,   100,
    Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
    //draw the face detected in the 0th (gray) channel with red color
    currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);

    if (trainingImages.ToArray().Length != 0)
    {
        MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);
```

```
        //Eigen face recognizer
        EigenObjectRecognizer recognizer = new EigenObjectRecognizer(
                                trainingImages.ToArray(),
                                labels.ToArray(),
                                3000,
                                ref termCrit);

        name = recognizer.Recognize(result);
        //Draw the label for each face detected and recognized
        currentFrame.Draw(name, ref font, new Point(f.rect.X - 2, f.rect.Y - 2), new
        Bgr(Color.Silver));

}
                //If is there a face, set the criteria based on recognition
                if (recording==false)
                {
                    //UnRecognized face
                    if (name == "")
                    {
                        countSeconds = 5;
                    }
                    //Recognized face
                    else
                    {
                        countSeconds = 3;
                    }
                    recording = true;
                }

                NamePersons[t-1] = name;
                NamePersons.Add("");

                //Set the number of faces detected on the scene
                label3.Text = facesDetected[0].Length.ToString();
            }

            if (recording)
                VideoGrabberBgrOptimized(currentFrame);

            t = 0;

            //Names concatenation of persons recognized
            for (int nnn = 0; nnn < facesDetected[0].Length; nnn++)
            {
                names = names + NamePersons[nnn] + ", ";
            }
            //Show the faces procesed and recognized
            imageBoxFrameGrabber.Image = currentFrame;
            label4.Text = names;
            names = "";
            //Clear the list(vector) of names
            NamePersons.Clear();
            GC.Collect();
        }
```

Figura 13. Módulo reconocimiento facial e implementación de criterios de almacenamiento.

4.4 Agregar Rostro a Base de Conocimiento

Para agregar un rostro en la base de conocimiento (Figura 14), se redimensiona el fotograma y se convierte la imagen a escala de grises, se realiza la detección del rostro, se corta en un tamaño 100 x 100 pixeles y se almacena en el disco duro, finalmente se agrega la etiqueta en el archivo de etiquetas.

```
try
{
//Trained face counter
ContTrain = ContTrain + 1;

//Get a gray frame from capture device
gray        =        grabber.QueryGrayFrame().Resize(imageBoxFrameGrabber.Size.Width,
 imageBoxFrameGrabber.Size.Height, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

//Face Detector
MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
               face,
               1.2,
               10,
               Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
               new Size(20, 20));

//Action for each element detected
foreach (MCvAvgComp f in facesDetected[0])
{
       TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();
       break;
}

//resize face detected image for force to compare the same size with the
//test image with cubic interpolation type method
TrainedFace = result.Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
               trainingImages.Add(TrainedFace);
               labels.Add(textBox1.Text);

//Show face added in gray scale
imageBox1.Image = TrainedFace;
//Write the number of trained faces in a file text for further load
File.WriteAllText(Application.StartupPath    +    "/TrainedFaces/TrainedLabels.txt",
trainingImages.ToArray().Length.ToString() + "%");

//Write the labels of trained faces in a file text for further load
for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
{
   trainingImages.ToArray()[i       -       1].Save(Application.StartupPath       +
   "/TrainedFaces/face" + i + ".bmp");
   File.AppendAllText(Application.StartupPath   +   "/TrainedFaces/TrainedLabels.txt",
   labels.ToArray()[i - 1] + "%");
}
```

```
MessageBox.Show(textBox1.Text + " se agrego a la base de conocimiento", "Training OK",
MessageBoxButtons.OK, MessageBoxIcon.Information);
}
catch
{
 MessageBox.Show("Primero se debe habilitar la detección facial", "Training Fail",
 MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
```

Figura 14. Modulo agregar rostro a base de conocimiento.

Como resultado de la implementación del sistema, se construyó un prototipo el cual es capaz de reconocer e identificar rostros, lo que permite implementar criterios de almacenamiento, que definirán la cantidad de fotogramas que se almacenarán en una secuencia de video.

El prototipo se probará en diferentes condiciones y escenarios, lo que permitirá obtener resultados cualitativos y cuantitativos, los cuales se documentarán en el capítulo siguiente.