Emgu CV Code samples

1. Camera Capture and Gray Scale Conversion

```
//namspace for emgu cv
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.Util;

namespace CameraCapture
{
   public partial class CameraCapture : Form
   {
      private Capture _capture = null;  //Capture images from either camera or video
file.
      private bool _captureInProgress;  //bool variable to keep track of the capture
status

      public CameraCapture()
      {
         InitializeComponent();
         try
         {
            _capture = new Capture();       //create instance of the class capture
            _capture.ImageGrabbed += ProcessFrame;   //call Process Frame Function
         }
         catch (NullReferenceException excpt)
         {
            MessageBox.Show(excpt.Message);
         }
      }

      private void ProcessFrame(object sender, EventArgs arg)
      {
         //get frame from the Camera
         Image<Bgr, Byte> frame = _capture.RetrieveBgrFrame();

         //convert the frame to gray scale
         Image<Gray, Byte> grayFrame = frame.Convert<Gray, Byte>();

        //show the resulted images in the imageboxes
         captureImageBox.Image = frame;
         grayscaleImageBox.Image = grayFrame;
      }


   }
}
```

2. Face and eye detection using camera

```
//Capture images from either camera or video file
Capture _capture = new Capture();

Image<Bgr, Byte> frame = _capture.RetrieveBgrFrame();      //get frame from the camera

DetectFace.Detect(frame, "haarcascade_frontalface_default.xml", "haarcascade_eye.xml",
faces, eyes, out detectionTime);

//Draw faces and eyes detected
 foreach (Rectangle face in faces)
    frame.Draw(face, new Bgr(Color.Red), 2);

 foreach (Rectangle eye in eyes)
    frame.Draw(eye, new Bgr(Color.Blue), 2);


public static void Detect(Image<Bgr, Byte> image, String faceFileName, String
eyeFileName)
{
  //Read the HaarCascade files
  using (CascadeClassifier face = new CascadeClassifier(faceFileName))
  using (CascadeClassifier eye = new CascadeClassifier(eyeFileName))
  {
    //Convert it to Grayscale
    using (Image<Gray, Byte> gray = image.Convert<Gray, Byte>())
    {
      //Detect the faces  from the gray scale image and store the locations as rectangle
      //The first dimensional is the channel
      //The second dimension is the index of the rectangle in the specific channel

      Rectangle[] facesDetected = face.DetectMultiScale(
              gray,
              1.1, //scale factor
              10,  //min neighbors
              new Size(20, 20),  //min size
              Size.Empty);       //max size

     foreach (Rectangle f in facesDetected)
     {
       //Set the region of interest on the faces (ROI)
        gray.ROI = f;

        foreach (Rectangle e in eyesDetected)
        {
           Rectangle eyeRect = e;
           eyeRect.Offset(f.X, f.Y);
           //the best match for the left eye was located at an offset of 168 rows
           //and 248 columns, and at an offset of 184 rows and 250
           //columns for the right eye (based on 512 × 512 images).
           eyes.Add(eyeRect);
          }
        }
     }
```

3. Motion Detection

```
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.VideoSurveillance;
using Emgu.Util;

private Capture _capture = new Capture();
private MotionHistory _motionHistory;


if (_capture != null) //if camera capture has been successfully created
{
   _motionHistory = new MotionHistory(
    1.0, //in second, the duration of motion history you wants to keep
    0.05, //in second, maxDelta for cvCalcMotionGradient
    0.5); //in second, minDelta for cvCalcMotionGradient

  _capture.ImageGrabbed += ProcessFrame;
  _capture.Start();
 }

private void ProcessFrame()
{
   using (Image<Bgr, Byte> image = _capture.RetrieveBgrFrame())
   using (MemStorage storage = new MemStorage()) //create storage for motion components
   {
      _forgroundDetector.Update(image);

       //update the motion history
       _motionHistory.Update(_forgroundDetector.ForegroundMask);

//get a copy of the motion mask and enhance its color

    _motionHistory.Mask.MinMax(out minValues, out maxValues, out minLoc, out maxLoc);
    Image<Gray, Byte> motionMask = _motionHistory.Mask.Mul(255.0 / maxValues[0]);

        //create the motion image
        Image<Bgr, Byte> motionImage = new Image<Bgr, byte>(motionMask.Size);

       //display the motion pixels in blue (first channel)
        motionImage[0] = motionMask;

       //Draw each individual motion in red
        DrawMotion(motionImage, comp.rect, angle, new Bgr(Color.Red));
  }
   //Display the image of the motion
    motionImageBox.Image = motionImage;
   }
 }
```