Kalibrasi Form.cs

```
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 Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.VideoSurveillance;
using Emgu.Util;
using Emgu.CV.CvEnum;
using Emgu.CV.Features2D;
using Emgu.CV.Cvb;
using Emgu.CV.ML;
using Emgu.CV.ML.Structure;

namespace Arya
{
   public partial class CekCitra : Form
   {
      #region inisialisasi citra
      private Capture capture;
      private Image<Bgr, byte> imgFrame;
      private Image<Hsv, byte> imgSource;
      private Image<Ycc, byte> imgSource2;
      private Image<Gray, byte> imgThreshold;
```

```
private Image<Gray, byte> imgThreshold2;
int HueLow = 0;
int HueHigh = 0;
int SatLow = 0;
int SatHigh = 0;
int ValLow = 0;
int ValHigh = 0;
int YLow = 0;
int YHigh = 0;
int CrLow = 0;
int CrHigh = 0;
int CbLow = 0;
int CbHigh = 0;
#endregion

public CekCitra()
{
    InitializeComponent();
    initialise_capture();
    //default range
    trackLHue.Value = 0;
    trackHHue.Value = 255;
    trackLSat.Value = 55;
    trackHSat.Value = 255;
    trackLVal.Value = 30;
    trackHVal.Value = 255;

    //default range
    trackLY.Value = 45;
    trackHY.Value = 230;
    trackLCr.Value = 130;
```

```
    trackHCr.Value = 160;
    trackLCb.Value = 100;
    trackHCb.Value = 150;
}

//Camera Start Stop
public void initialise_capture()
{

    capture = new Capture();
    capture.QueryFrame();
    //Initialize the FrameGraber event
    Application.Idle += new EventHandler(FrameGrabber);

}

void FrameGrabber(object sender, EventArgs arg)
{
    #region Range HSV
    HueLow = trackLHue.Value;
    HueHigh = trackHHue.Value;
    SatLow = trackLSat.Value;
    SatHigh = trackHSat.Value;
    ValLow = trackLVal.Value;
    ValHigh = trackHVal.Value;
    lblLHue.Text = "Low Hue = " + HueLow;
    lblHHUe.Text = "High Hue = " + HueHigh;
    lblLSat.Text = "Low Sat = " + SatLow;
    lblHSat.Text = "High Sat = " + SatHigh;
    lblLVal.Text = "Low Val = " + ValLow;
    lblHVal.Text = "High Val = " + ValHigh;
```

```
        #endregion
        #region Range YCrCb
        YLow = trackLY.Value;
        YHigh = trackHY.Value;
        CrLow = trackLCr.Value;
        CrHigh = trackHCr.Value;
        CbLow = trackLCb.Value;
        CbHigh = trackHCb.Value;
        lblLY.Text = "Low Y = " + YLow;
        lblHY.Text = "High Y = " + YHigh;
        lblLCr.Text = "Low Cr = " + CrLow;
        lblHCr.Text = "High Cr = " + CrHigh;
        lblLCb.Text = "Low Cb = " + CbLow;
        lblHCb.Text = "High Cb = " + CbHigh;
        #endregion

        #region Elemen
        StructuringElementEx rect_12 = new StructuringElementEx(10, 10, 5, 5,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
        StructuringElementEx rect_6 = new StructuringElementEx(6, 6, 3, 3,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
        StructuringElementEx rect_3 = new StructuringElementEx(4, 4, 2, 2,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
        #endregion

        //mengaktifkan video streaming webcam
        imgFrame = capture.QueryFrame();

        //merubah frame ke bit
        Bitmap bmpOri = imgFrame.ToBitmap();
        Image<Bgr, byte> FrameOri = new Image<Bgr, byte>(bmpOri);
```

```
        //proses threshold
        imgSource = FrameOri.Convert<Hsv, byte>();
        imgSource2 = FrameOri.Convert<Ycc, byte>();
        imgThreshold = new Image<Gray, byte>(imgSource2.Size); //convert ycc
to gray
        imgThreshold = new Image<Gray, byte>(imgSource.Size); //convert hsv
to gray

        if (imgThreshold != null && cBCitraHSV.Checked && imgBoxCitra !=
null) //Citra HSV
        {
            imgThreshold = new Image<Gray, byte>(imgSource.Size); //convert
hsv to gray
            imgThreshold = imgSource.InRange(new Hsv(HueLow, SatLow,
ValLow), new Hsv(HueHigh, SatHigh, ValHigh));
            imgSource = imgSource.ThresholdBinary(new Hsv(50, 50, 50), new
Hsv(255, 255, 255));

            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }

        if (imgThreshold != null && cBCitraYCrCb.Checked && imgBoxCitra !=
null) //Citra YCrCb
        {
```

```
            imgThreshold = new Image<Gray, byte>(imgSource2.Size); //convert
ycc to gray
            imgThreshold = imgSource2.InRange(new Ycc(YLow, CrLow,
CbLow), new Ycc(YHigh, CrHigh, CbHigh));
            imgSource2 = imgSource2.ThresholdBinary(new Ycc(50, 50, 50), new
Ycc(255, 255, 255));

            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }

        imgBoxOri.Image = FrameOri.Flip(FLIP.HORIZONTAL);
    }

    private void btnTutup_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void btnTentang_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Fungsi Kalibrasi Disini Untuk Mengetahui Nilai
Ruang Warna Citra YCbCr Dan HSV", "Tentang Kalibrasi",
MessageBoxButtons.OK);
    }
  }
  }
```

Ruang Warna YCbCr Dan HSV.cs

```
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.Runtime.InteropServices;
using DirectShowLib;
using Emgu.CV.UI;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using Emgu.CV.VideoSurveillance;
using Emgu.Util;
using Emgu.CV.ML;
using System.IO;
using System.Drawing.Imaging;
using System.Xml;
using System.Threading;
using System.Xml.Serialization;

namespace Arya
{
   public partial class Training_Form : Form
   {
      #region Variables
```

```
        //Camera specific
        Capture capture;

        //Images for finding object
        Image<Bgr, byte> imgFrame;
        Image<Gray, byte> result = null;
        Image<Gray, byte> imgThreshold = null;

        //Untuk Kotak Deteksi
        MCvBox2D box;
        Rectangle rect;
        MemStorage storage = new MemStorage();

        //For aquiring 10 images in a row
        List<Image<Gray, byte>> resultImages = new List<Image<Gray, byte>>();
        int resultsList = 0;
        int numImages = 10;
        bool record = false;

        //Saving Jpg
        List<Image<Gray, byte>> ImagestoWrite = new List<Image<Gray,
byte>>();
        EncoderParameters ENC_Parameters = new EncoderParameters(1);
        EncoderParameter ENC = new
EncoderParameter(System.Drawing.Imaging.Encoder.Quality, 100);
        ImageCodecInfo Image_Encoder_JPG;

        //Saving XAML Data file
        List<string> NamestoWrite = new List<string>();
        List<string> NamesforFile = new List<string>();
        XmlDocument docu = new XmlDocument();
```

```
//KNN
int resizedImageWidth = 100;
int resizedImageheight = 150;
int K = 5;
int classes = 5;
int train_Samples = 50;
KNearest Knn;

//Variables
Form1 mainForm;
#endregion

public Training_Form()
{
   InitializeComponent();
   ENC_Parameters.Param[0] = ENC;
   Image_Encoder_JPG = GetEncoder(ImageFormat.Jpeg);
   initialise_capture();
}

//Camera Start Stop
public void initialise_capture()
{

      capture = new Capture();
      capture.QueryFrame();
      //Initialize the FrameGraber event
      Application.Idle += new EventHandler(FrameGrabber);

}
```

```
private void stop_capture()
{
    Application.Idle -= new EventHandler(FrameGrabber);
    if (capture != null)
    {
        capture.Dispose();
    }
    //Initialize the FrameGraber event
}

void FrameGrabber(object sender, EventArgs arg)
{
    #region Elemen
    StructuringElementEx rect_12 = new StructuringElementEx(10, 10, 5, 5,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
    StructuringElementEx rect_6 = new StructuringElementEx(6, 6, 3, 3,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
    StructuringElementEx rect_3 = new StructuringElementEx(4, 4, 2, 2,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
    #endregion

    //mengaktifkan video streaming webcam
    imgFrame = capture.QueryFrame().Resize(320, 240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

    //merubah frame ke bit
    Bitmap bmpOri = imgFrame.ToBitmap();
    Image<Bgr, byte> FrameOri = new Image<Bgr, byte>(bmpOri);

    //Convert
```

```
            Image<Ycc, byte> imgSourceYCrCb = FrameOri.Convert<Ycc,
byte>();
            Image<Ycc, byte> imgSourceYCrCb1 = FrameOri.Convert<Ycc,
byte>();
            Image<Hsv, byte> imgSourceHSV = FrameOri.Convert<Hsv, byte>();
            Image<Hsv, byte> imgSourceHSV1 = FrameOri.Convert<Hsv, byte>();
            imgThreshold = new Image<Gray, byte>(imgSourceYCrCb.Size);
//convert ycc to gray
            imgThreshold = new Image<Gray, byte>(imgSourceHSV.Size);
//convert hsv to gray

            Image<Gray, byte>[] channelsycrcb = imgSourceYCrCb1.Split();
            Image<Gray, byte> y = channelsycrcb[0];
            Image<Gray, byte> cb = channelsycrcb[1];
            Image<Gray, byte> cr = channelsycrcb[2];



           Image<Gray, byte>[] channelshsv = imgSourceHSV1.Split();
           Image<Gray, byte> h = channelshsv[0];
           Image<Gray, byte> s = channelshsv[1];
           Image<Gray, byte> v = channelshsv[2];

            #region Terang YCrCb
         if (cBTYCrCb.Checked && imgThreshold != null && imgBoxY != null
&& imgBoxCr != null && imgBoxCb != null && imgBoxYCrCb != null)
         {
            imgThreshold = new Image<Gray, byte>(imgSourceYCrCb.Size);
//convert ycc to gray
            imgThreshold = imgSourceYCrCb.InRange(new Ycc(135, 110, 40),
new Ycc(230, 160, 150));
```

```
            imgSourceYCrCb = imgSourceYCrCb.ThresholdBinary(new Ycc(50,
50, 50), new Ycc(255, 255, 255));
            y = imgThreshold.AbsDiff(y);
            cb = imgThreshold.AbsDiff(cb);
            cr = imgThreshold.AbsDiff(cr);
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxYCrCb.Image = imgSourceYCrCb.Flip(FLIP.HORIZONTAL);
            imgBoxY.Image = y.Flip(FLIP.HORIZONTAL);
            imgBoxCr.Image = cr.Flip(FLIP.HORIZONTAL);
            imgBoxCb.Image = cb.Flip(FLIP.HORIZONTAL);
            imgBoxThreshold.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

            #region Normal YCrCb
        if (cBNYCrCb.Checked && imgThreshold != null && imgBoxY != null
&& imgBoxCr != null && imgBoxCb != null && imgBoxYCrCb != null)
            {
            imgThreshold = new Image<Gray, byte>(imgSourceYCrCb.Size);
//convert ycc to gray
            imgThreshold = imgSourceYCrCb.InRange(new Ycc(45, 130, 100),
new Ycc(230, 160, 150));
            imgSourceYCrCb = imgSourceYCrCb.ThresholdBinary(new Ycc(50,
50, 50), new Ycc(255, 255, 255));
            y = imgThreshold.AbsDiff(y);
            cb = imgThreshold.AbsDiff(cb);
            cr = imgThreshold.AbsDiff(cr);
            //citra dilatasi dan erosi
```

```
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxYCrCb.Image = imgSourceYCrCb.Flip(FLIP.HORIZONTAL);
            imgBoxY.Image = y.Flip(FLIP.HORIZONTAL);
            imgBoxCr.Image = cr.Flip(FLIP.HORIZONTAL);
            imgBoxCb.Image = cb.Flip(FLIP.HORIZONTAL);
            imgBoxThreshold.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
            }
        #endregion

            #region Redup YCrCb
        if (cBRYCrCb.Checked && imgThreshold != null && imgBoxY != null
&& imgBoxCr != null && imgBoxCb != null && imgBoxYCrCb != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSourceYCrCb.Size);
//convert ycc to gray
            imgThreshold = imgSourceYCrCb.InRange(new Ycc(32,0,0), new
Ycc(255,255,255));
            imgSourceYCrCb = imgSourceYCrCb.ThresholdBinary(new Ycc(50,
50, 50), new Ycc(255, 255, 255));
            y = imgThreshold.AbsDiff(y);
            cb = imgThreshold.AbsDiff(cb);
            cr = imgThreshold.AbsDiff(cr);
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxYCrCb.Image = imgSourceYCrCb.Flip(FLIP.HORIZONTAL);
            imgBoxY.Image = y.Flip(FLIP.HORIZONTAL);
            imgBoxCr.Image = cr.Flip(FLIP.HORIZONTAL);
```

```
            imgBoxCb.Image = cb.Flip(FLIP.HORIZONTAL);
            imgBoxThreshold.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

            #region Terang HSV
        if (cBTHSV.Checked && imgThreshold != null && imgBoxH != null &&
imgBoxS != null && imgBoxV != null && imgBoxHSV != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSourceHSV.Size);
//convert hsv to gray
            imgThreshold = imgSourceHSV.InRange(new Hsv(0, 0, 0), new
Hsv(255, 55, 255));
            imgSourceHSV = imgSourceHSV.ThresholdBinary(new Hsv(50, 50,
50), new Hsv(255, 255, 255));
            h = imgThreshold.AbsDiff(h);
            s = imgThreshold.AbsDiff(s);
            v = imgThreshold.AbsDiff(v);
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_3, 2);

            imgBoxHSV.Image = imgSourceHSV.Flip(FLIP.HORIZONTAL);
            imgBoxH.Image = h.Flip(FLIP.HORIZONTAL);
            imgBoxS.Image = s.Flip(FLIP.HORIZONTAL);
            imgBoxV.Image = v.Flip(FLIP.HORIZONTAL);
            imgBoxThreshold.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

            #region Normal HSV
```

```
        if (cBNHSV.Checked && imgThreshold != null && imgBoxH != null
&& imgBoxS != null && imgBoxV != null && imgBoxHSV != null)
            {
            imgThreshold = new Image<Gray, byte>(imgSourceHSV.Size);
//convert hsv to gray
            imgThreshold = imgSourceHSV.InRange(new Hsv(0, 0, 0), new
Hsv(80, 255, 255));
            imgSourceHSV = imgSourceHSV.ThresholdBinary(new Hsv(50, 50,
50), new Hsv(255, 255, 255));
            h = imgThreshold.AbsDiff(h);
            s = imgThreshold.AbsDiff(s);
            v = imgThreshold.AbsDiff(v);
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxHSV.Image = imgSourceHSV.Flip(FLIP.HORIZONTAL);
            imgBoxH.Image = h.Flip(FLIP.HORIZONTAL);
            imgBoxS.Image = s.Flip(FLIP.HORIZONTAL);
            imgBoxV.Image = v.Flip(FLIP.HORIZONTAL);
            imgBoxThreshold.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
            }
        #endregion

            #region Redup HSV
        if (cBRHSV.Checked && imgThreshold != null && imgBoxH != null
&& imgBoxS != null && imgBoxV != null && imgBoxHSV != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSourceHSV.Size);
//convert hsv to gray
```

```
            imgThreshold = imgSourceHSV.InRange(new Hsv(0, 0, 30), new
Hsv(255, 255, 255));
            imgSourceHSV = imgSourceHSV.ThresholdBinary(new Hsv(50, 50,
50), new Hsv(255, 255, 255));
            h = imgThreshold.AbsDiff(h);
            s = imgThreshold.AbsDiff(s);
            v = imgThreshold.AbsDiff(v);
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxHSV.Image = imgSourceHSV.Flip(FLIP.HORIZONTAL);
            imgBoxH.Image = h.Flip(FLIP.HORIZONTAL);
            imgBoxS.Image = s.Flip(FLIP.HORIZONTAL);
            imgBoxV.Image = v.Flip(FLIP.HORIZONTAL);
            imgBoxThreshold.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

        Contour<Point> kuntur =
imgThreshold.FindContours(Emgu.CV.CvEnum.CHAIN_APPROX_METHOD.
CV_CHAIN_APPROX_SIMPLE,
Emgu.CV.CvEnum.RETR_TYPE.CV_RETR_TREE, storage);
            Contour<Point> kunturbesar = null;

            //ekstraksi kuntur besar
            double hasil1 = 0;
            double hasil2 = 0;
            while (kuntur != null)
            {
```

```
                hasil1 = kuntur.Area;
                if (hasil1 > hasil2)
                {
                    hasil2 = hasil1;
                    kunturbesar = kuntur;
                }
                kuntur = kuntur.HNext;
            }

            if (kunturbesar != null)
            {
                kunturbesar = kunturbesar.ApproxPoly(kunturbesar.Perimeter *
0.000015, storage);
                if (kunturbesar.BoundingRectangle.Width > 1 &&
kunturbesar.BoundingRectangle.Height > 1)
                {
                    box = kunturbesar.GetMinAreaRect();
                    rect = box.MinAreaRect();
                    FrameOri.Draw(rect, new Bgr(255, 0, 0), 1);
                }
            }

            if (record && resultImages.Count < numImages)
            {
                resultImages.Add(result);
                if (resultImages.Count == numImages)
                {
                    btnAddYCrCb.Enabled = true;
                    record = false;
                    Application.Idle -= new EventHandler(FrameGrabber);
                }
```

```
            }

            Image<Gray, byte> sizeROI = imgThreshold.Copy();
            sizeROI.ROI = new Rectangle(rect.Location.X, rect.Location.Y,
rect.Width, rect.Height);
            //CvInvoke.cvShowImage("ROI", sizeROI.Flip(FLIP.HORIZONTAL));

            imgBoxOri.Image = FrameOri.Flip(FLIP.HORIZONTAL);
            imgBoxROI.Image = sizeROI.Flip(FLIP.HORIZONTAL);



        }

        #region  Saving The Data
        private bool saveTrainingData(IImage Tangan_data)
        {
           try
           {
              //Random rand = new Random(); untuk label acak dalam num++ diganti
rand.next()
              int num = 0;
              bool file_create = true;
              string tanganName = "Hasil_" + statusImages.Text + "_" +
num++.ToString() + ".jpg";
              while (file_create)
              {
                 if (!File.Exists(Application.StartupPath + "/Trained/" +
tanganName))
                 {
                    file_create = false;
                 }
```

```
                else
                {
                    tanganName = "Hasil_" + statusImages.Text + "_" +
num++.ToString() + ".jpg";
                }
            }

            //Menentukan Lokasi Gambar tanganName
            if (Directory.Exists(Application.StartupPath + "/Trained/"))
                {
                    Tangan_data.Save(Application.StartupPath + "/Trained/" +
tanganName);
                }
                else
                {
                    Directory.CreateDirectory(Application.StartupPath + "/Trained/");
                    Tangan_data.Save(Application.StartupPath + "/Trained/" +
tanganName);
                }

            if (File.Exists(Application.StartupPath +
"/Trained/TrainedLabels.xml"))
            {
                //File.AppendAllText(Application.StartupPath +
"/Trained/TrainedLabels.txt", mtxTrainData + "\n\r");

                bool loading = true;
                while (loading)
                {
                    try
                    {
```

```
                docu.Load(Application.StartupPath +
"/Trained/TrainedLabels.xml");
                loading = false;
            }
            catch
            {
                docu = null;
                docu = new XmlDocument();
                Thread.Sleep(10);
            }
        }

        //Get the root element
        XmlElement root = docu.DocumentElement;

        XmlElement Hand_D = docu.CreateElement("Tangan");
        XmlElement name_D = docu.CreateElement("Status");
        XmlElement file_D = docu.CreateElement("File");

        //Add the values for each nodes
        name_D.InnerText = statusImages.Text;
        file_D.InnerText = tanganName;

        //Construct the status element
        Hand_D.AppendChild(name_D);
        Hand_D.AppendChild(file_D);

        //Add the element to the end of the root element
        root.AppendChild(Hand_D);

        //Save the document
```

```
                docu.Save(Application.StartupPath + "/Trained/TrainedLabels.xml");
            }
            else
            {
                FileStream FS_Tangan = File.OpenWrite(Application.StartupPath +
"/Trained/TrainedLabels.xml");
                using (XmlWriter writer = XmlWriter.Create(FS_Tangan))
                {
                    writer.WriteStartDocument();
                    writer.WriteStartElement("Hands_For_Training");

                    writer.WriteStartElement("Tangan");
                    writer.WriteElementString("Status", statusImages.Text);
                    writer.WriteElementString("File", tanganName);
                    writer.WriteEndElement();

                    writer.WriteEndElement();
                    writer.WriteEndDocument();

                }
                FS_Tangan.Close();
            }
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }

    private ImageCodecInfo GetEncoder(ImageFormat format)
```

```
        {
            ImageCodecInfo[] codecs = ImageCodecInfo.GetImageDecoders();
            foreach (ImageCodecInfo codec in codecs)
            {
                if (codec.FormatID == format.Guid)
                {
                    return codec;
                }
            }
            return null;
        }
        #endregion

        //Add the image to training data
        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (resultImages.Count == numImages)
            {
                if (!saveTrainingData(imgBoxOri.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                if (!saveTrainingData(imgBoxThreshold.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not saved",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                if (!saveTrainingData(imgBoxY.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                if (!saveTrainingData(imgBoxCr.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

```
            if (!saveTrainingData(imgBoxCb.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxYCrCb.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not saved",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxROI.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);


        }
        else
        {
            stop_capture();
            if (!saveTrainingData(imgBoxOri.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxThreshold.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not saved",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxY.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxCr.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxCb.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

```
            if (!saveTrainingData(imgBoxYCrCb.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not saved",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxROI.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            initialise_capture();
        }
    }

    private void btnTutup_Click(object sender, EventArgs e)
    {

        this.Close();

    }

    private void btnAddHSV_Click(object sender, EventArgs e)
    {
        if (resultImages.Count == numImages)
        {
            if (!saveTrainingData(imgBoxOri.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxThreshold.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not saved",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxH.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

```
            if (!saveTrainingData(imgBoxS.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxV.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxHSV.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxROI.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);

        }
        else
        {
            stop_capture();
            if (!saveTrainingData(imgBoxOri.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxThreshold.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not saved",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxH.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxS.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

```
            if (!saveTrainingData(imgBoxV.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxHSV.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            if (!saveTrainingData(imgBoxROI.Image)) MessageBox.Show("Error",
"Error in saving file info. Training data not saved", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            initialise_capture();
          }
       }

       private void btnTentang_Click(object sender, EventArgs e)
       {
          MessageBox.Show("Perbandingan YCbCr Dan HSV Dalam Pendeteksian
Tangan Pada Virtual Mouse \nUntuk Mengetahui Perbedaan Citra Yang Terbaik
Dalam 3 Kondisi Pencahayaan Yaitu Terang, Normal, Redup", "Tentang
Penelitian Jurnal", MessageBoxButtons.OK);
       }
   }
}
```

Main Form.cs

```
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 Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.VideoSurveillance;
using Emgu.Util;
using Emgu.CV.CvEnum;
using Emgu.CV.Features2D;
using Emgu.CV.Cvb;
using Emgu.CV.ML;
using Emgu.CV.ML.Structure;
using System.Runtime.InteropServices;
using DirectShowLib;
using System.IO;

namespace Arya
{
   public partial class Form1 : Form
   {
      #region inisialisasi
      private Capture capture;
      private Image<Bgr, byte> imgFrame;
```

```
        private Image<Hsv, byte> imgSource;
        private Image<Ycc, byte> imgSource2;
        private Image<Gray, byte> imgThreshold;
        private Image<Gray, byte> imgThreshold2;
        private bool prosescapture = false;
        int selectedCam = 0;
        //Convex Hull & Convexity Defact
        int finger_num = 0;
        Seq<Point> Hull;
        Seq<MCvConvexityDefect> defects;
        MCvConvexityDefect[] defectArray;
        MCvBox2D box;
        Rectangle rect;
        MemStorage penyimpanan = new MemStorage();
        //Knn
        int K = 1;
        int size = 150;
        int classes = 100;
        KNearest knn;
        #endregion

        #region Interface dan Perintah Mouse
        //interface perangkat mouse
        [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention =
CallingConvention.StdCall)]
        static extern void mouse_event(uint dwFlags, uint dx, uint dy, uint dwData,
int dwExtraInfo);

        // constants for the mouse_input() API function
        private const int MOUSEEVENTF_MOVE = 0x01;
        private const int MOUSEEVENTF_LEFTDOWN = 0x02;
```

```
        private const int MOUSEEVENTF_LEFTUP = 0x04;
        private const int MOUSEEVENTF_RIGHTDOWN = 0x08;
        private const int MOUSEEVENTF_RIGHTUP = 0x10;
        private const int MOUSEEVENTF_MIDDLEDOWN = 0x20;
        private const int MOUSEEVENTF_MIDDLEUP = 0x40;
        private const int MOUSEEVENTF_WHEEL = 0x800;
        private Type typeShell = null;
        private object objShell = Type.Missing;

        private void Move()
        {
           uint dX = Convert.ToUInt32(Cursor.Position.X);
           uint dY = Convert.ToUInt32(Cursor.Position.Y);
           mouse_event(MOUSEEVENTF_MOVE, dX, dY, 0, 0);
        }

        private void DoMouseClick()
        {
           uint dX = Convert.ToUInt32(Cursor.Position.X);
           uint dY = Convert.ToUInt32(Cursor.Position.Y);
           mouse_event(MOUSEEVENTF_LEFTDOWN |
MOUSEEVENTF_LEFTUP, dX, dY, 0, 0);
        }

        private void DoMouseRightClick()
        {
           uint dX = Convert.ToUInt32(Cursor.Position.X);
           uint dY = Convert.ToUInt32(Cursor.Position.Y);
           mouse_event(MOUSEEVENTF_RIGHTDOWN |
MOUSEEVENTF_RIGHTUP, dX, dY, 0, 0);
        }
```

```
        private void DoMouseScrollUp()
        {
            uint dX = Convert.ToUInt32(Cursor.Position.X);
            uint dY = Convert.ToUInt32(Cursor.Position.Y);
            mouse_event(MOUSEEVENTF_MIDDLEUP, dX, dY, 0, 0);
        }

        private void DoMouseScrollDown()
        {
            uint dX = Convert.ToUInt32(Cursor.Position.X);
            uint dY = Convert.ToUInt32(Cursor.Position.Y);
            mouse_event(MOUSEEVENTF_MIDDLEDOWN, dX, dY, 0, 0);
        }
        #endregion

        public Form1()
        {
            InitializeComponent();
            ShowDesktop();
            //untuk settingan webcam usb
            DsDevice[] CameraDevice =
DsDevice.GetDevicesOfCat(FilterCategory.VideoInputDevice);
            for (int i = 0; i < CameraDevice.Length; i++)
            {
                cmbCamName.Items.Add("[" + i + "]: " + CameraDevice[i].Name);
            }
        }

        void ProsesFrameGestureTangan(object sender, EventArgs arg)
        {
```

```
        #region Elemen
        StructuringElementEx rect_12 = new StructuringElementEx(10, 10, 5, 5,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
        StructuringElementEx rect_6 = new StructuringElementEx(6, 6, 3, 3,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
        StructuringElementEx rect_3 = new StructuringElementEx(4, 4, 2, 2,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
        #endregion

        //mengaktifkan video streaming webcam
        imgFrame = capture.QueryFrame();

        //merubah frame ke bit
        Bitmap bmpOri = imgFrame.ToBitmap();
        Image<Bgr, byte> FrameOri = new Image<Bgr, byte>(bmpOri);

        //proses threshold
        imgSource = FrameOri.Convert<Hsv, byte>();
        imgSource2 = FrameOri.Convert<Ycc, byte>();
        imgThreshold = new Image<Gray, byte>(imgSource2.Size); //convert ycc
to gray
        imgThreshold = new Image<Gray, byte>(imgSource.Size); //convert hsv
to gray

        #region Terang YCrCb
        if (cBTYCrCb.Checked && imgThreshold != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSource2.Size); //convert
ycc to gray
            imgThreshold = imgSource2.InRange(new Ycc(135, 110, 40), new
Ycc(230, 160, 150));
```

```
            imgSource2 = imgSource2.ThresholdBinary(new Ycc(50, 50, 50), new
Ycc(255, 255, 255));
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxDisplay.Image = imgSource2.Flip(FLIP.HORIZONTAL);
            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

        #region Normal YCrCb
        if (cBNYCrCb.Checked && imgThreshold != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSource2.Size); //convert
ycc to gray
            imgThreshold = imgSource2.InRange(new Ycc(45, 130, 100), new
Ycc(230, 160, 150));
            imgSource2 = imgSource2.ThresholdBinary(new Ycc(50, 50, 50), new
Ycc(255, 255, 255));
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxDisplay.Image = imgSource2.Flip(FLIP.HORIZONTAL);
            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

        #region Redup YCrCb
        if (cBRYCrCb.Checked && imgThreshold != null)
```

```
        {
            imgThreshold = new Image<Gray, byte>(imgSource2.Size); //convert
ycc to gray
            imgThreshold = imgSource2.InRange(new Ycc(32, 0, 0), new Ycc(255,
255, 255));
            imgSource2 = imgSource2.ThresholdBinary(new Ycc(50, 50, 50), new
Ycc(255, 255, 255));
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxDisplay.Image = imgSource2.Flip(FLIP.HORIZONTAL);
            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

        #region Terang HSV
        if (cBTHSV.Checked && imgThreshold != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSource.Size); //convert
hsv to gray
            imgThreshold = imgSource.InRange(new Hsv(0, 0, 0), new Hsv(255,
55, 255));
            imgSource = imgSource.ThresholdBinary(new Hsv(50, 50, 50), new
Hsv(255, 255, 255));
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_3, 2);

            imgBoxDisplay.Image = imgSource.Flip(FLIP.HORIZONTAL);
            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
```

```
        }
        #endregion

        #region Normal HSV
        if (cBNHSV.Checked && imgThreshold != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSource.Size); //convert
hsv to gray
            imgThreshold = imgSource.InRange(new Hsv(0, 0, 0), new Hsv(80,
255, 255));
            imgSource = imgSource.ThresholdBinary(new Hsv(50, 50, 50), new
Hsv(255, 255, 255));
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);
            imgBoxDisplay.Image = imgSource.Flip(FLIP.HORIZONTAL);
            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

        #region Redup HSV
        if (cBRHSV.Checked && imgThreshold != null)
        {
            imgThreshold = new Image<Gray, byte>(imgSource.Size); //convert
hsv to gray
            imgThreshold = imgSource.InRange(new Hsv(0, 0, 30), new Hsv(255,
255, 255));
            imgSource = imgSource.ThresholdBinary(new Hsv(50, 50, 50), new
Hsv(255, 255, 255));
            //citra dilatasi dan erosi
            CvInvoke.cvErode(imgThreshold, imgThreshold, rect_12, 1);
```

```
            CvInvoke.cvDilate(imgThreshold, imgThreshold, rect_6, 2);

            imgBoxDisplay.Image = imgSource.Flip(FLIP.HORIZONTAL);
            imgBoxCitra.Image = imgThreshold.Flip(FLIP.HORIZONTAL);
        }
        #endregion

        //flip gambar
        imgThreshold = imgThreshold.Flip(FLIP.HORIZONTAL);
        FrameOri = FrameOri.Flip(FLIP.HORIZONTAL);

        //ekstraksi kuntur
        Contour<Point> kuntur = imgThreshold.FindContours();
        Contour<Point> kunturbesar = null;

        //ekstraksi kuntur besar
        double hasil1 = 0;
        double hasil2 = 0;
        while (kuntur != null)
        {
            hasil1 = kuntur.Area;
            if (hasil1 > 18000 && hasil1 < 74000)
            {
                kunturbesar = kuntur;
            }
            kuntur = kuntur.HNext;
        }

        if (kunturbesar != null && cBConvexHull.Checked) //
kunturbesar.BoundingRectangle.Width > 20 &&
kunturbesar.BoundingRectangle.Height > 20)
```

```
        {
            finger_num = 0;
            kunturbesar = kunturbesar.ApproxPoly(kunturbesar.Perimeter * 0.0025,
penyimpanan);
            Hull =
kunturbesar.GetConvexHull(ORIENTATION.CV_CLOCKWISE);
            box = kunturbesar.GetMinAreaRect();

            PointF[] points = box.GetVertices();
            Point[] ps = new Point[points.Length];

            FrameOri.Draw(kunturbesar, new Bgr(Color.Yellow), 2);
            FrameOri.DrawPolyline(Hull.ToArray(), true, new Bgr(0, 0, 256), 2);

            if (cBConvexityDefacts.Checked)
            {
               for (int i = 0; i < points.Length; i++)
                  ps[i] = new Point((int)points[i].X, (int)points[i].Y);

               Seq<Point> filteredhull = new Seq<Point>(penyimpanan);
               for (int i = 0; i < Hull.Total; i++)
               {
                  if (Math.Sqrt(Math.Pow(Hull[i].X - Hull[i + 1].X, 2) +
Math.Pow(Hull[i].Y - Hull[i + 1].Y, 2)) > box.size.Width / 10)
                  {
                     filteredhull.Push(Hull[i]);
                  }
               }

               for (int i = 0; i < filteredhull.Total; i++)
               {
```

```
                PointF nilaihull = new PointF((float)filteredhull[i].X,
                                        (float)filteredhull[i].Y);
                CircleF bulathull = new CircleF(nilaihull, 4);
                //FrameOri.Draw(bulathull, new Bgr(Color.Aquamarine), 2);
            }

            defects = kunturbesar.GetConvexityDefacts(penyimpanan,
ORIENTATION.CV_CLOCKWISE);
            defectArray = defects.ToArray();

            for (int i = 0; i < defects.Total; i++)
            {
                PointF starPoint = new PointF((float)defectArray[i].StartPoint.X,
(float)defectArray[i].StartPoint.Y);
                PointF depthPoint = new
PointF((float)defectArray[i].DepthPoint.X, (float)defectArray[i].DepthPoint.Y);
                PointF endPoint = new PointF((float)defectArray[i].EndPoint.X,
(float)defectArray[i].EndPoint.Y);

                LineSegment2D garisAwalTengah = new
LineSegment2D(defectArray[i].StartPoint, defectArray[i].DepthPoint);
                LineSegment2D garisTengahAkhir = new
LineSegment2D(defectArray[i].DepthPoint, defectArray[i].EndPoint);

                CircleF startCircle = new CircleF(starPoint, 5f);
                CircleF depthCircle = new CircleF(depthPoint, 5f);
                CircleF endCircle = new CircleF(endPoint, 5f);

                if ((startCircle.Center.Y < box.center.Y || depthCircle.Center.Y <
box.center.Y) && (startCircle.Center.Y < depthCircle.Center.Y) &&
(Math.Sqrt(Math.Pow(startCircle.Center.X - depthCircle.Center.X, 2) +
```

```
Math.Pow(startCircle.Center.Y - depthCircle.Center.Y, 2)) > box.size.Height /
6.5))
                {
                   finger_num++;
                   imgFrame.Draw(garisAwalTengah, new Bgr(Color.Green), 2);
                }

                FrameOri.Draw(startCircle, new Bgr(Color.DarkBlue), 3);
                FrameOri.Draw(depthCircle, new Bgr(Color.Yellow), 5);
                CircleF boxMiddleCircle = new CircleF(box.center, 5f);
                FrameOri.Draw(boxMiddleCircle, new Bgr(Color.Red), 1);
             }

             if (cBKnn.Checked)
             {
                #region KNN
                Matrix<float> trainData = new Matrix<float>(classes, 2);
                Matrix<float> trainClasses = new Matrix<float>(classes, 1);

                Matrix<float> data1 = trainData.GetRows(0, classes >> 1, 1);
                data1.SetRandNormal(new MCvScalar(200), new
MCvScalar(50));
                Matrix<float> data2 = trainData.GetRows(classes >> 1, classes, 1);
                data2.SetRandNormal(new MCvScalar(300), new
MCvScalar(50));

                if (finger_num == 5 || finger_num == 4)
                {
                   lblStatus.Text = "Gerak Pointer";
                }
```

```
                if (finger_num == 1 || finger_num == 2)
                {
                   lblStatus.Text = "Klik Kiri";
                }

                if (finger_num == 3)
                {
                   lblStatus.Text = "Klik Kanan";
                }

                Matrix<float> datakelas1 = trainClasses.GetRows(0, classes >> 1,
1);
                datakelas1.SetValue(2);
                Matrix<float> datakelas2 = trainClasses.GetRows(classes >> 1,
classes, 1);
                datakelas2.SetValue(2);

                KNearest knn = new KNearest(trainData, trainClasses, null, false,
K);
                #endregion
            }
            MCvFont font = new
MCvFont(FONT.CV_FONT_HERSHEY_DUPLEX, 2d, 2d);
            FrameOri.Draw(finger_num.ToString(), ref font, new Point(10, 60),
new Bgr(Color.White));
        }

    }

    #region proses kalibrasi layar dan memasukkan perintah mouse
    //Mengsingkronkan Koordinat Kuntur
```

```
            MCvMoments moment = new MCvMoments();
            try
            {
               moment = kunturbesar.GetMoments();
            }
            catch(NullReferenceException expt)
            {
               return;
            }
            CvInvoke.cvMoments(kunturbesar, ref moment, 0);
            double m_00 = CvInvoke.cvGetSpatialMoment(ref moment, 0, 0);
            double m_10 = CvInvoke.cvGetSpatialMoment(ref moment, 1, 0);
            double m_01 = CvInvoke.cvGetSpatialMoment(ref moment, 0, 1);
            int current_X = Convert.ToInt32(m_10 / m_00) / 10;
            int current_Y = Convert.ToInt32(m_01 / m_00) / 10;
            int curX = current_X * 11;
            int curY = current_Y * 11;

            if (cBConvexityDefacts.Checked)
            {
               lblPosisi.Text = "Koordinat : " + curX + ", " + curY;
            }

            if (cBFungsiMouse.Checked)
            {
               if (finger_num == 5 || finger_num == 4) //finger_num == 1 ||
finger_num == 2 || finger_num == 3 || finger_num == 4 ||
               {
                  Cursor.Position = new Point(1366 / 640 * (curX - 100), 768 / 480 *
(curY - 100));
```

```
            }

            if (finger_num == 1 || finger_num == 2)
            {
                DoMouseClick();
            }

            if (finger_num == 3)
            {
                DoMouseRightClick();
            }

            if (kunturbesar == null)
            {
                finger_num = 0;
            }
        }
        #endregion
        //mengsingkronkan dengan box
        imgBoxOri.Image = FrameOri;
    }

    #region Menu
    private void cmbCamName_SelectedIndexChanged(object sender,
EventArgs e)
    {
        selectedCam = cmbCamName.SelectedIndex;
        try
        {
            capture = new Emgu.CV.Capture(selectedCam);
```

```
capture.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.CV_CAP_PROP_F
PS, 30);

capture.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.CV_CAP_PROP_F
RAME_WIDTH, 1280);

capture.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.CV_CAP_PROP_F
RAME_HEIGHT, 720);
        }
        catch (NullReferenceException excpt)
        {
            MessageBox.Show(excpt.Message);
        }
    }

    private void btnGT_Click(object sender, EventArgs e)
    {
        if (capture != null)
        {
            if (prosescapture)
            {
                prosescapture = true;
                Application.Idle -= ProsesFrameGestureTangan;
            }
            else
            {
                prosescapture = false;
                Application.Idle += ProsesFrameGestureTangan;
            }
            prosescapture = !prosescapture;
```

```
    }
}

private void Form1_FormClosed(object sender, FormClosedEventArgs arg)
{
    if (capture != null)
    {
        capture.Dispose();
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    try
    {
        capture = new Capture();

    }
    catch (NullReferenceException except)
    {
        return;
    }
}

private void btnTrain_Click(object sender, EventArgs e)
{
    capture.Stop();
    //OpenForm
    Training_Form TF = new Training_Form();
    TF.Show();
}
```

```
        private void btnCekCitra_Click(object sender, EventArgs e)
        {
            capture.Stop();
            //OpenForm
            CekCitra CC = new CekCitra();
            CC.Show();
        }

        private void btnAbout_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Judul Tugas Akhir : Implementasi K-Nearest
Neighbor (KNN) Untuk Tracking Gerakan Tangan Berbasis Webcam
\nPembimbing 1 : Sopian Soim \nPembimbing 2 : Irawan Hadi \nDibuat Oleh Tri
Arya Nugraha \nNim : 061340351632 \nPoliteknik Negeri Sriwijaya", "Tentang
Tugas Akhir", MessageBoxButtons.OK);
        }

        private void btnTutup_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void ShowDesktop()
        {
            typeShell = Type.GetTypeFromProgID("Shell.Application");
            objShell = Activator.CreateInstance(typeShell);
            typeShell.InvokeMember("MinimizeAll",
System.Reflection.BindingFlags.InvokeMethod, null, objShell, null);
        }
```

```
        private void UnShowDesktop()
        {
            typeShell = Type.GetTypeFromProgID("Shell.Application");
            objShell = Activator.CreateInstance(typeShell);
            typeShell.InvokeMember("UndoMinimizeAll",
System.Reflection.BindingFlags.InvokeMethod, null, objShell, null);
        }

        #endregion

    }
}
```