



Computer Vision

Hands on Lab



September 2013

For the latest information, please see bluejack.binus.ac.id



Information in this document, including URL and other Internet Web site references, is subject to change without notice. This document supports a preliminary release of software that may be changed substantially prior to final commercial release, and is the proprietary information of Binus University.

This document is for informational purposes only. BINUS UNIVERSITY MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

The entire risk of the use or the results from the use of this document remains with the user. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Binus University.

Binus University may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Binus University, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2011 Binus University. All rights reserved.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Table of Content

CHAPTER 01	INTRODUCTION TO EMGUCV	1
CHAPTER 02	IMAGE PROCESSING	10
CHAPTER 03	EDGE DETECTION	31
CHAPTER 04	SHAPE DETECTION.....	44
CHAPTER 05	PATTERN RECOGNITION AND CLASSIFICATION.....	62
REFERENCES:	81



OVERVIEW

Chapter 1

- Introduction to EmguCV dan OpenCV
Mempelajari mengenai *library* dari OpenCV dan cara menginstall dan menggunakan EmguCV di C#.

Chapter 2

- Image Processing
Mempelajari jenis pemrosesan citra yang sederhana, meliputi *smoothing*, *grayscale*, dan *thresholding*.

Chapter 3

- Edge Detection
Melakukan pemrosesan *image* untuk dapat mendeteksi sisi, menggunakan *canny*, *sobel* dan *laplacian edge detection*.

Chapter 4

- Shape Detection
Melakukan pendeteksian objek-objek sederhana, meliputi garis, lingkaran, dan persegi.

Chapter 5

- Pattern Recognition and Classification
Mengetahui suatu langkah awal pada *pattern recognition*, seperti deteksi wajah (*face detection*).

Chapter 01



Introduction to EmguCV

SOFTWARE LABORATORY CENTER

1.1. EmguCV

EmguCV adalah suatu .Net *wrapper* yang *cross platform* yang digunakan untuk OpenCV *image processing library*. Dengan menggunakan EmguCV, fungsi-fungsi OpenCV dapat digunakan untuk bahasa pemrograman .Net yang kompatibel seperti: **C#, VB, C++**, dan lain sebagainya. EmguCV dapat jalan di berbagai sistem operasi, seperti: **Windows, Linux, Mac OS X, iPhone, iPad**, dan **Android**.

1.2. OpenCV

OpenCV adalah sebuah *library* yang digunakan untuk memproses gambar yang dapat berjalan secara *multiplatform*.

Pengaplikasian **OpenCV** mencakup:

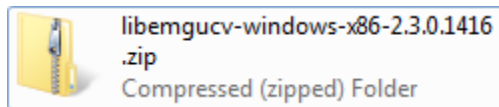
- *2D and 3D feature toolkits*
- *Ego-motion*
- *Face Recognition*
- *Gesture Recognition*
- *Human-Computer Interface (HCI)*
- *Mobile Robotics*
- *Motion Understanding*
- *Object Identification*
- *Segmentation and Recognition*
- *Stereopsis Stereo Vision: depth perception from 2 cameras*
- *Structure from Motion (SFM)*
- *Motion Tracking*



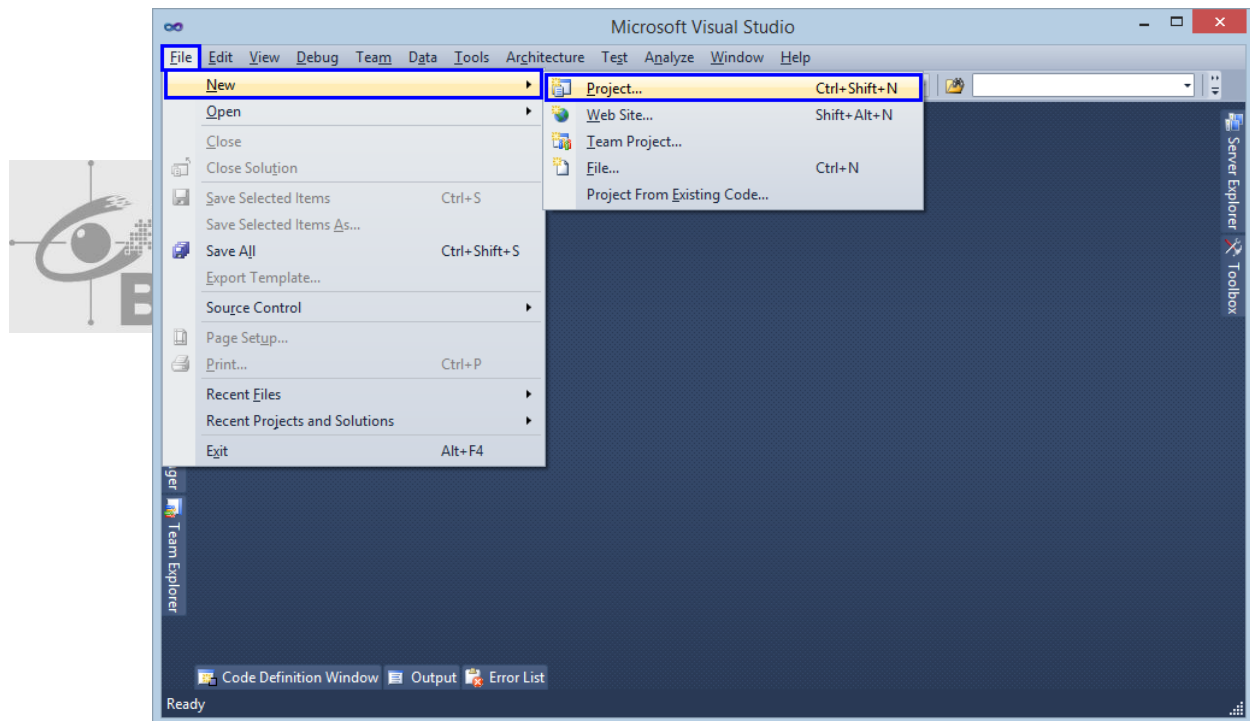
1.3. Instalasi EmguCV

EmguCV dapat di-*download* secara gratis melalui *website* resminya pada *link* berikut ini: <http://www.emgu.com/> (versi yang digunakan pada pembelajaran ini adalah **EmguCV 2.3.0**). Langkah-langkah untuk instalasi **EmguCV** dan menghubungkannya dengan **Microsoft Visual Studio 2010 C#** adalah sebagai berikut:

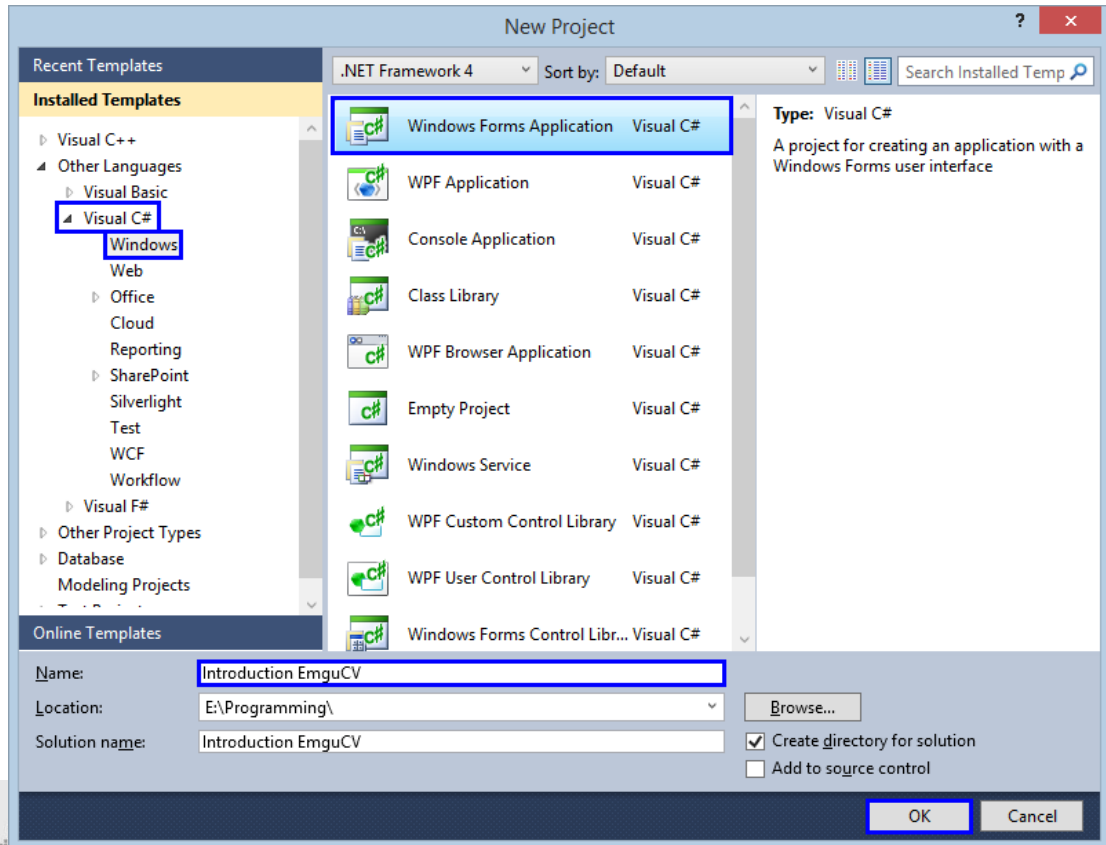
1. *Download* dan *extract*-lah **libemgucv-windows-x86-2.3.0.1416.zip**.



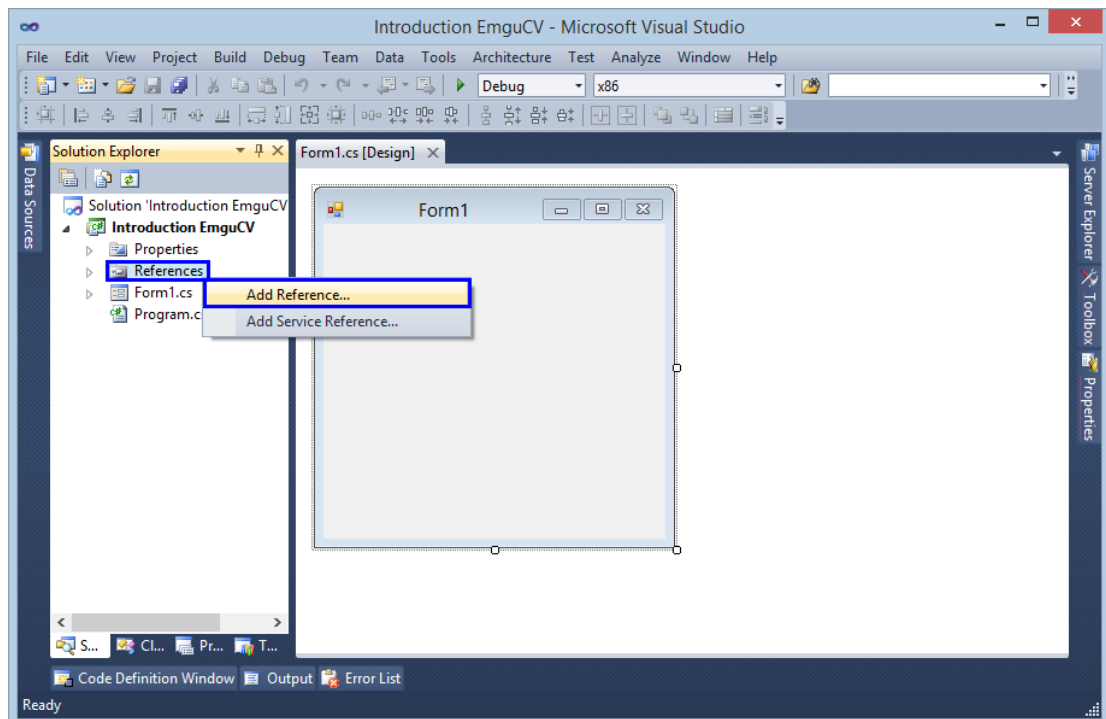
2. Buka **Microsoft Visual Studio 2010 C#** dan buatlah suatu *project* baru dengan cara mengklik menu **File > New > Project** atau menekan **Ctrl + Shift + N**.



3. Pilih *template* **Visual C# > Windows > Windows Forms Application**. Beri nama, misalnya **“Introduction EmguCV”**. Lalu, klik tombol **OK**.



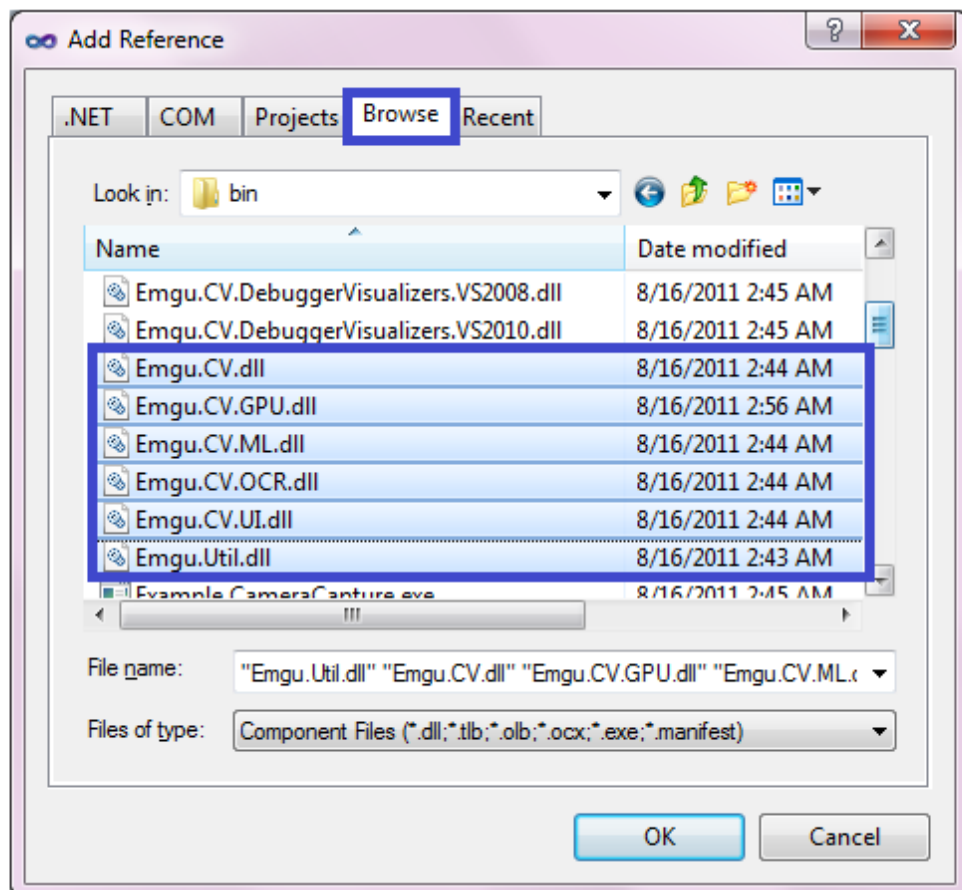
4. Pada **Solution Explorer**, klik kanan di *folder References* dan klik **Add Reference**.



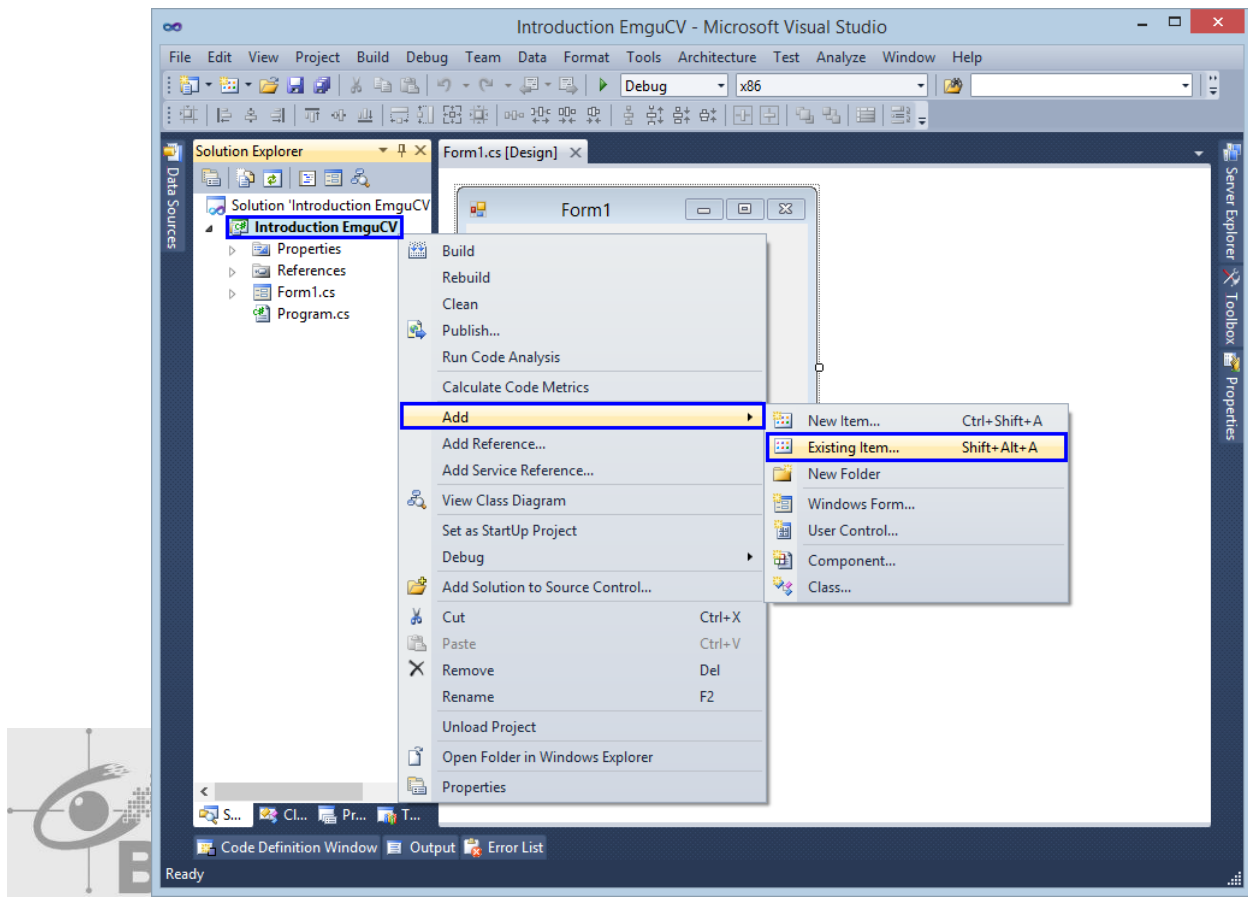
5. Pilih tab **Browse**, lalu pilih hasil *extract libemgucv-windows-x86-2.3.0.1416* yang di *folder libemgucv-windows-x86-2.3.0.1416 > bin* dan pilih file “.dll” berikut ini:

1. Emgu.CV.dll
2. Emgu.CV.GPU.dll
3. Emgu.CV.ML.dll
4. Emgu.CV.OCR.dll
5. Emgu.CV.UI.dll
6. Emgu.Util.dll

Kemudian, klik tombol OK.



6. Pada **Solution Explorer**, klik kanan di **Project**, klik **Add** dan klik **Existing Item**.



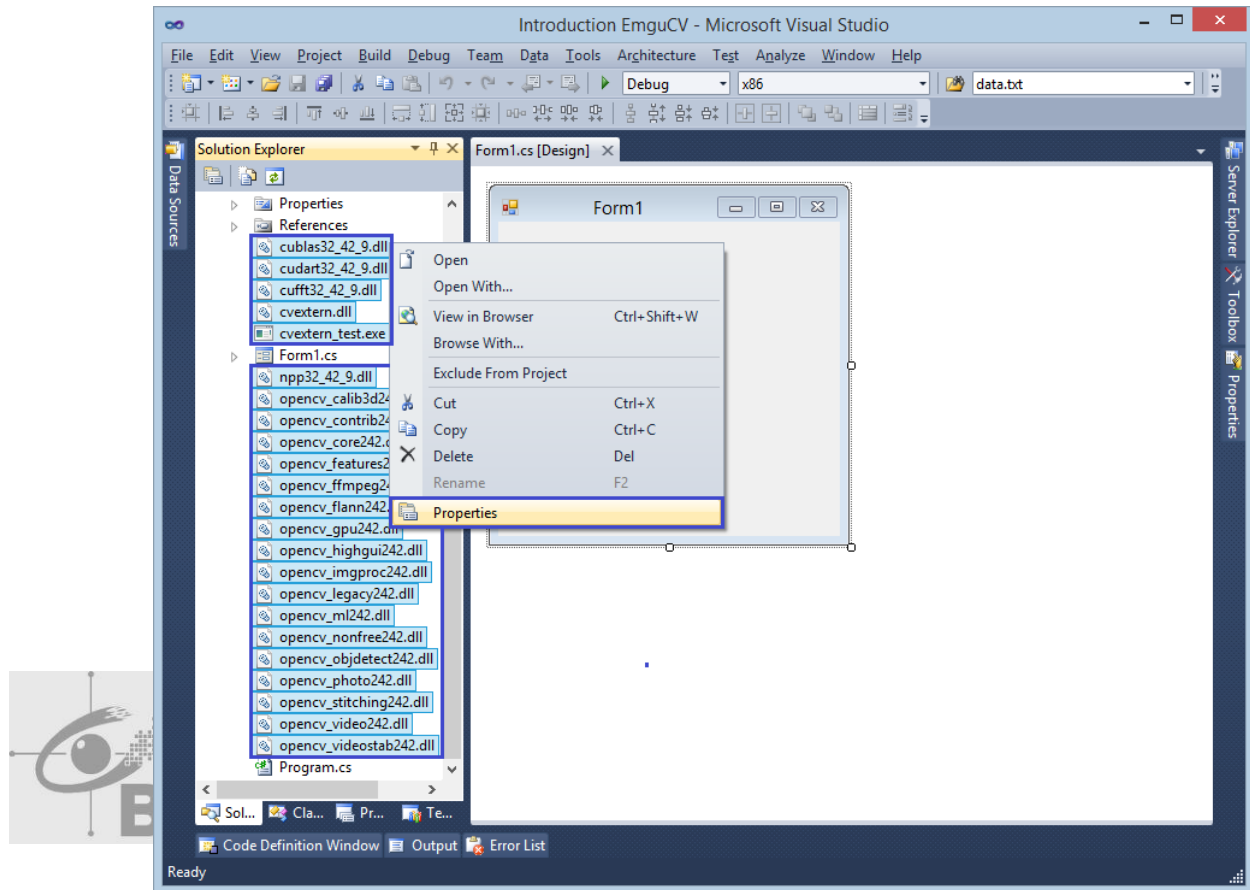
7. Pilih hasil *extract libemgucv-windows-x86-2.3.0.1416* yang di *folder libemgucv-windows-x86-2.3.0.1416 > bin*, dan pilih “.dll” berikut ini:

1. cudart32_40_17.dll
2. cufft32_40_17.dll
3. cvextern.dll
4. cvextern_gpu.dll
5. npp32_40_17.dll
6. opencv_calib3d231.dll
7. opencv_contrib231.dll
8. opencv_core231.dll
9. opencv_features2d231.dll
10. opencv_ffmpeg.dll
11. opencv_flann231.dll
12. opencv_gpu231.dll
13. opencv_highgui231.dll
14. opencv_imgproc231.dll
15. opencv_legacy231.dll
16. opencv_ml231.dll
17. opencv_objdetect231.dll
18. opencv_video231.dll
19. ZedGraph.dll

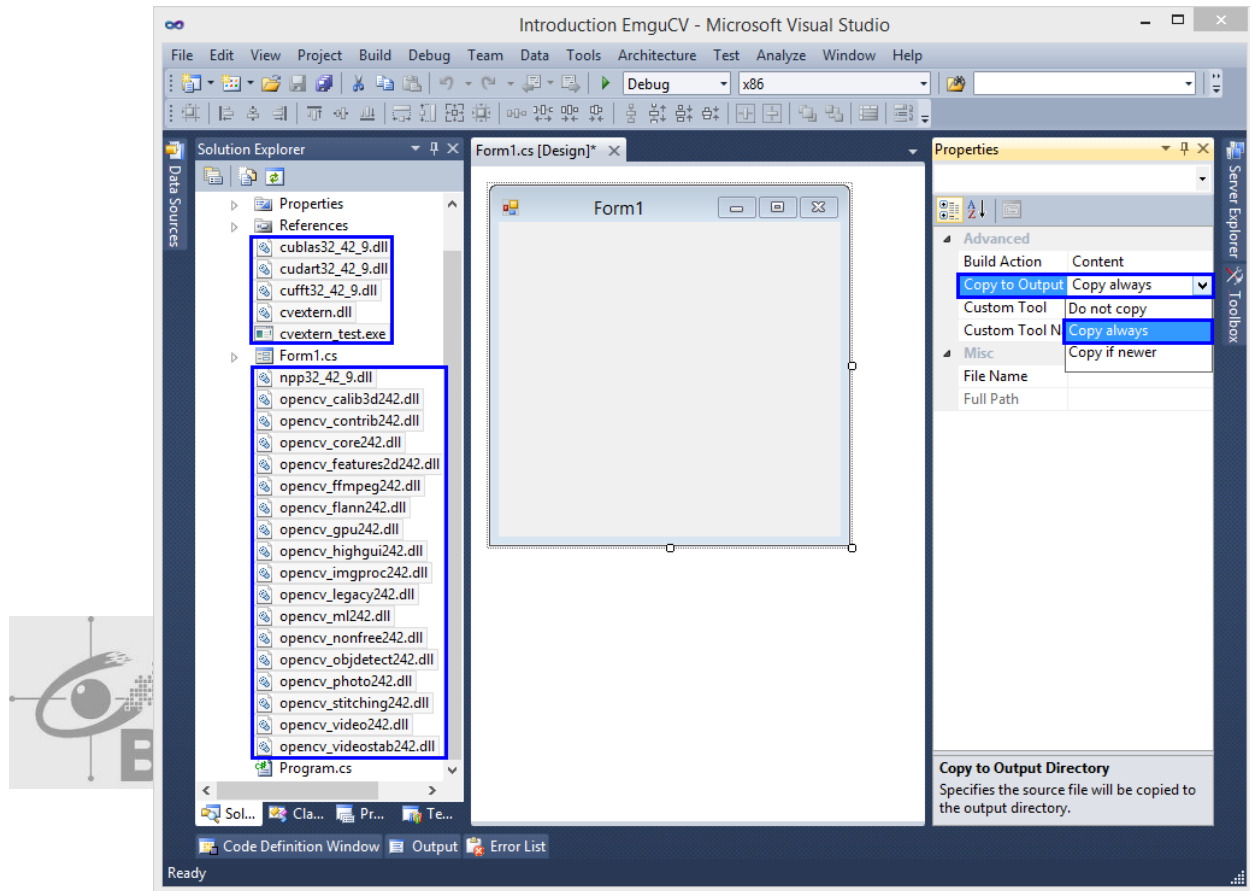
Lalu Pilih **Add**.



8. Setelah itu, *select* semua *file* yang baru saja dimasukkan. Lalu klik kanan pada *file* tersebut, pilih **Properties**.



9. Pada **Properties** yang bagian **Copy to Output**, pilih **Copy always** agar ketika *project* di-*debug* atau di-*release* maka semua *file* yang di-*select* akan ter-*copy* secara otomatis ke dalam *folder debug* atau *release*.



10. Setelah semua langkah dilakukan, maka **EmguCV** pada **Microsoft Visual Studio 2010 C#** telah selesai dikonfigurasi dan telah dapat digunakan.

Chapter 02

Image Processing



UNIVERSITY

SOFTWARE LABORATORY CENTER

2.1. Grayscale

Dalam komputasi, suatu citra digital *grayscale* atau *greyscale* adalah suatu citra dimana nilai dari setiap *pixel* merupakan *sample* tunggal. Citra yang ditampilkan terdiri atas warna “**abu-abu**”, bervariasi di warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra *grayscale* berbeda dengan citra “hitam-putih”. Dimana pada konteks komputer, citra hitam putih hanya terdiri atas 2 warna saja yaitu “**hitam**” dan “**putih**” saja.

Pada citra *grayscale* warna bervariasi antara hitam dan putih, tetapi variasi warna diantaranya sangat banyak. Citra *grayscale* seringkali merupakan perhitungan dari intensitas cahaya pada setiap *pixel* pada spektrum elektromagnetik *single band*. Citra *grayscale* disimpan dalam format 8 bit untuk setiap *sample pixel*, yang memungkinkan sebanyak 256 intensitas. Format ini sangat membantu dalam pemrograman karena manipulasi bit yang tidak terlalu banyak. Pada aplikasi lain seperti pada aplikasi *medical imaging* dan *remote sensing* biasa juga digunakan format 10, 12 maupun 16 bit.

2.2. Threshold

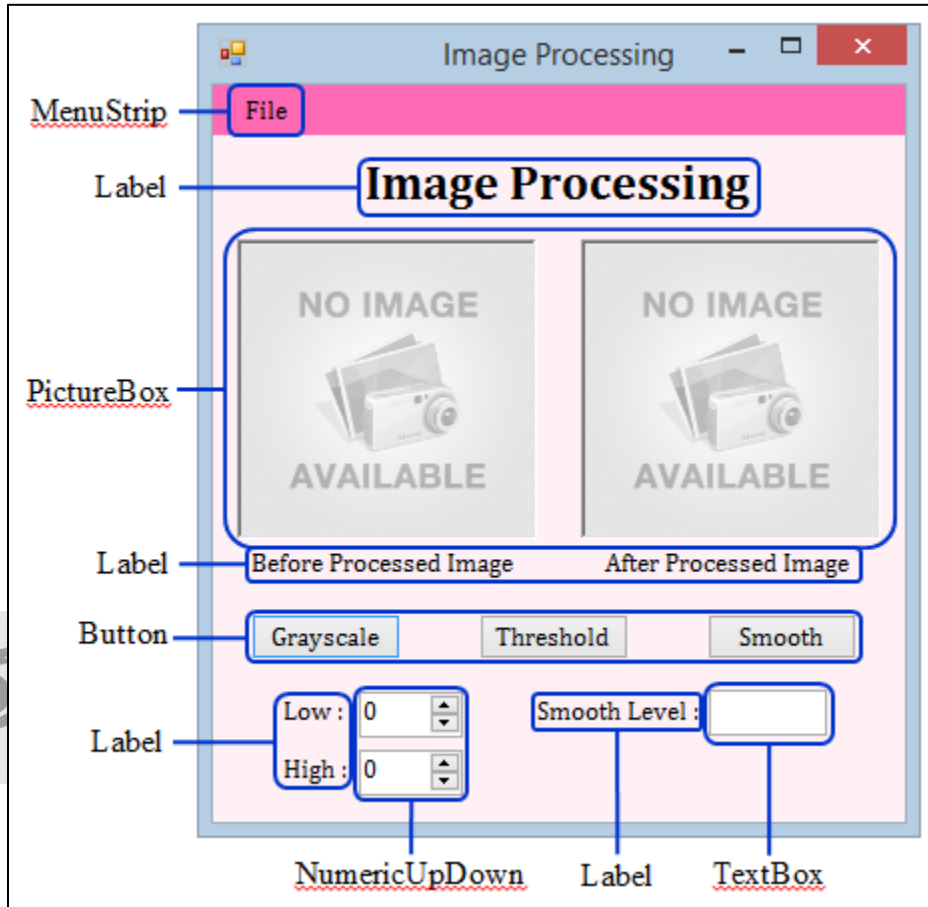
Threshold merupakan suatu operasi untuk mengubah gambar *gray-scale* ataupun gambar berwarna menjadi gambar biner berdasarkan pada nilai *threshold* tertentu. Adapun gambar biner merupakan gambar yang hanya memiliki dua nilai intensitas warna untuk setiap *pixel*-nya, yaitu nilai **255 (warna putih)** dan **0 (warna hitam)**.

2.3. Smoothing

Smoothing (disebut juga *blur*) adalah operasi pengolahan citra sederhana yang sering digunakan. Ada banyak alasan untuk menghaluskan, tetapi biasanya dilakukan untuk mengurangi *noise* (gangguan) pada *image*. *Smoothing* juga penting ketika kita ingin mengurangi resolusi dari suatu gambar.

2.4. Exercise

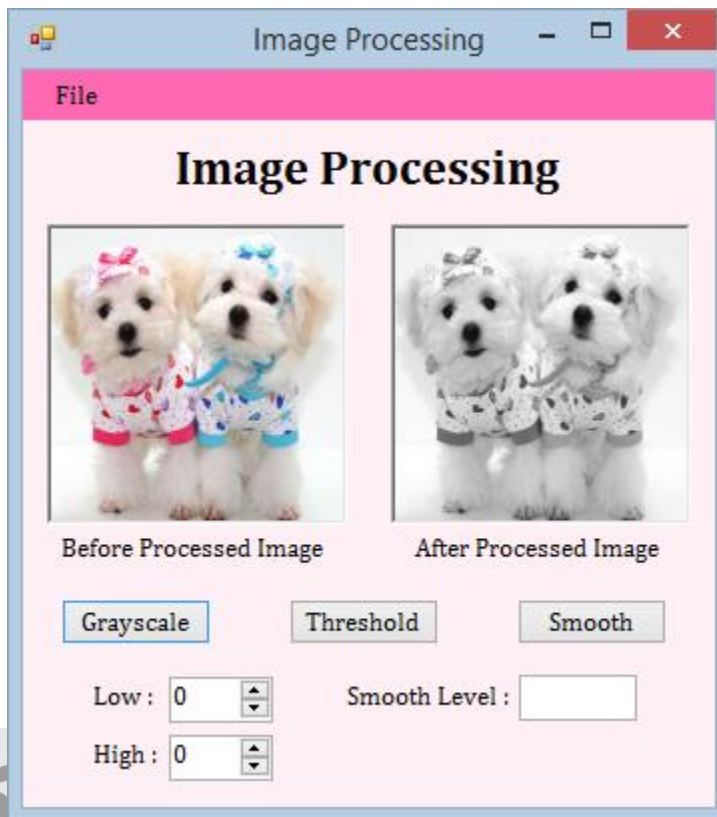
Buatlah sebuah program untuk melakukan operasi *grayscale*, *threshold*, dan *smooth* (*median blur*), seperti gambar berikut ini:



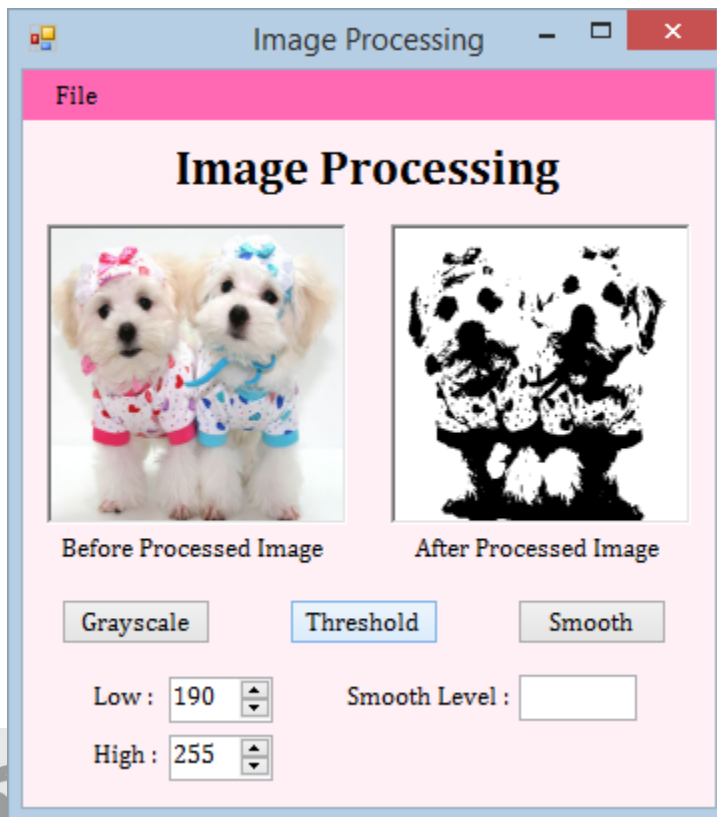
Keterangan:

Pada menu *strip* terdapat menu **File** yang terdiri atas 2 menu *items*, yakni: **Load Image** (*shortcut CTRL + I*) dan **Exit** (*shortcut CTRL + E*).

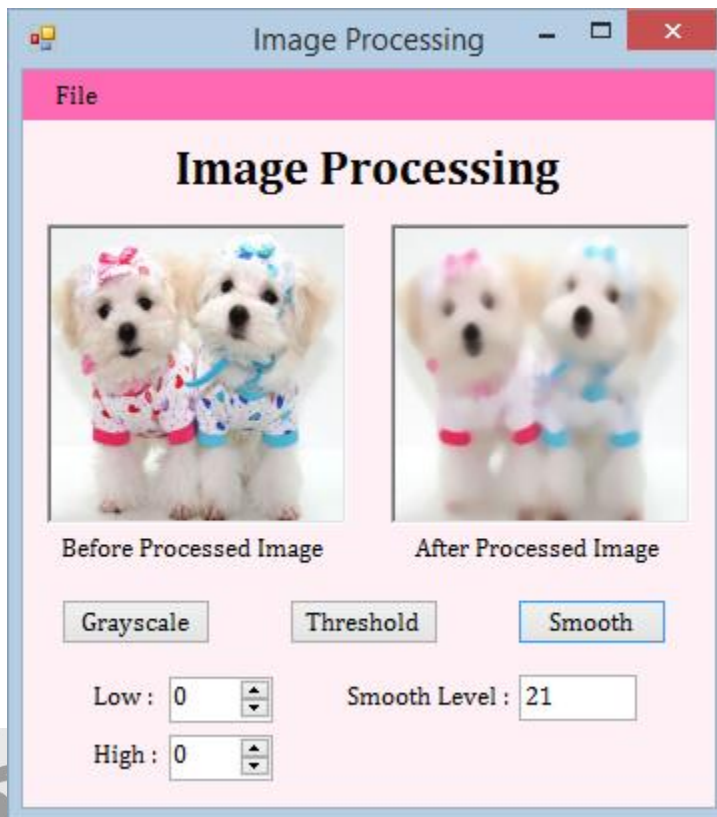
Hasil dari **Grayscale**:



Hasil dari **Threshold**:



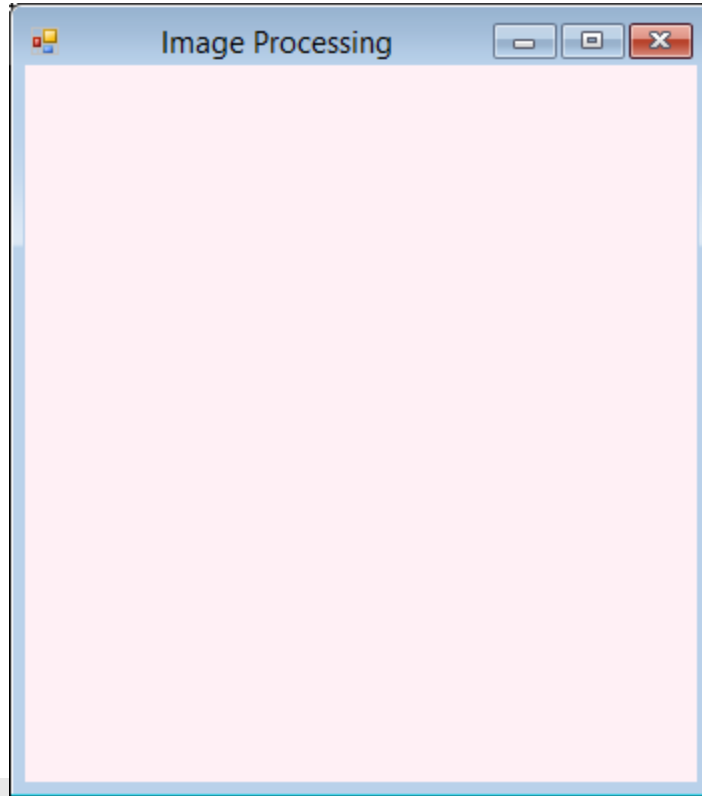
Hasil dari **Smooth (Median Blur)**:



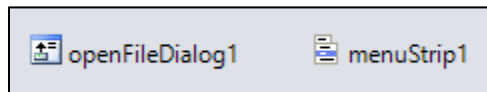
Jawaban:

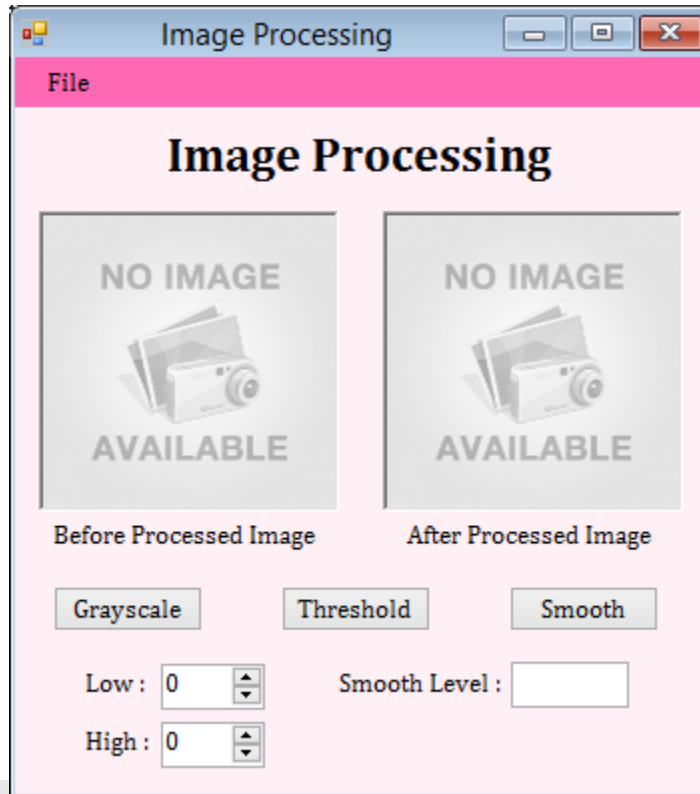
Berikut merupakan langkah-langkah pengerjaan:

1. Buat *project* baru dari **Microsoft Visual Studio 2010 C#**. Cara membuat *project* baru dan mengkonfigurasi **EmguCV** masih sama seperti yang telah kita pelajari pada **Chapter I**.
2. Lalu, ubah **Size form** menjadi *width* = 352 dan *height* = 397, ganti **Text** menjadi “*Image Processing*”, dan ubah **BackColor** menjadi “*LavenderBlush*”.

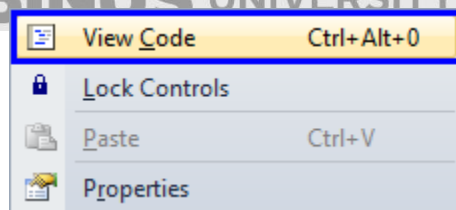


3. Masukkan semua komponen yang dibutuhkan, seperti: **MenuStrip**, **OpenFileDialog**, **Label**, **PictureBox**, **Button**, **NumericUpDown**, dan **TextBox**. Lalu, atur *properties* yang diperlukan untuk setiap komponen.





4. Klik kanan pada *form* dan pilih **View Code**.



5. *Import namespace* yang dibutuhkan untuk **EmguCV**.

```
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
```

Penjelasan:

Untuk menggunakan fungsi-fungsi **EmguCV**, diperlukan *namespace* berikut:

- *Namespace* **Emgu.CV**, berguna sebagai *wrapper* dari OpenCV *image processing function*.

- Namespace **Emgu.CV.Structure**, berguna sebagai *wrapper* dari OpenCV *structure*.
- Namespace **Emgu.CV.CvEnum**, berguna sebagai *wrapper* untuk OpenCV *enumeration*.

6. Buat *variable global* yang akan digunakan untuk *image processing*.

```
Image<Bgr, Byte> ori, editWithColor;
Image<Gray, Byte> edit;
```

Penjelasan:

Image adalah suatu *class* untuk **IplImage** dan mendefinisikan **TColor** sebagai tipe warna untuk **IplImage** dan **TDepth** sebagai *depth* dari **IplImage**.

Bgr adalah suatu *class* yang mendefinisikan warna *blue green red*.

Gray adalah suatu *class* yang mendefinisikan warna *gray*.

Byte adalah suatu *class* yang mendefinisikan jenis *depth* dalam ukuran *byte*.

7. Tambahkan *event click* pada *menu item Load Image*.

```
private void loadImageToolStripMenuItem_Click(object sender, EventArgs e)
{
    DialogResult result = openFileDialog1.ShowDialog();

    if (result == System.Windows.Forms.DialogResult.OK)
    {
        originalImg.Image = Image.FromFile(openFileDialog1.FileName);

        ori = new Image<Bgr, byte>(new Bitmap(originalImg.Image));
        edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
        editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
    }
}
```

Penjelasan:

DialogResult merupakan representasi hasil dari *form* yang digunakan sebagai *dialog box*.

```
DialogResult result = openFileDialog1.ShowDialog();
```

Digunakan untuk menampilkan **OpenFileDialog** dan menampung hasilnya ke **DialogResult**.

```
if (result == System.Windows.Forms.DialogResult.OK)
```

Digunakan untuk memastikan apakah *user* telah mengklik tombol **OK** pada **OpenFileDialog** atau belum. Jika *user* telah mengklik tombol **OK** maka program akan mengeksekusi baris selanjutnya.

```
originalImg.Image = Image.FromFile(openFileDialog1.FileName);
```

Digunakan untuk menampilkan *image* yang dipilih dari **OpenFileDialog** ke **PictureBox**.

```
ori = new Image<Bgr, byte>(new Bitmap(originalImg.Image));
```

Digunakan untuk membuat *object* dari **Image** <**Bgr**, **byte**> yang berisi gambar awal.

```
edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
```

Digunakan untuk membuat *object* dari **Image** dengan warna **Gray** yang ukurannya disesuaikan dengan gambar awal.

```
editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
```

Digunakan untuk membuat *object* **Image** dengan warna **Bgr** yang ukurannya disesuaikan dengan gambar awal.

8. Tambahkan *event click* pada **Grayscale** button.

```
private void grayScaleBtn_Click(object sender, EventArgs e)
{
    CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);

    editedImg.Image = edit.ToBitmap();
}
```

Penjelasan:

Fungsi ini digunakan untuk memproses gambar awal menjadi gambar *grayscale*. Gambar hasil proses akan ditampung di variabel yang berbeda sehingga gambar awal tidak akan berubah.

```
CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
```

Digunakan untuk melakukan konversi warna gambar dari **Bgr** ke **Gray**. Fungsi **cvCvtColor()** ini memiliki parameter sebagai berikut:

- 1) **Source:** *image* awal yang akan dikonversi warnanya (variabel awal dengan tipe data **Image <Bgr, byte>**).
- 2) **Destination:** target *image* hasil konversi (variabel tujuan dengan tipe data **Image <Gray, byte>**).
- 3) **COLOR_CONVERSION code:** kode konversi *image*. Table tetapan dari **color conversion code**, antara lain:

Kode Color Conversion	Keterangan
CV_BGR2BGRA	Convert BGR color to BGRA color
CV_RGB2RGBA	Convert RGB color to RGBA color
CV_BGRA2BGR	Convert BGRA color to BGR color
CV_RGBA2RGB	Convert RGBA color to RGB color
CV_BGR2RGBA	Convert BGR color to RGBA color
CV_RGB2BGRA	Convert RGB color to BGRA color
CV_RGBA2BGR	Convert RGBA color to BGR color
CV_BGRA2RGB	Convert BGRA color to RGB color
CV_BGR2RGB	Convert BGR color to RGB color
CV_RGB2BGR	Convert RGB color to BGR color
CV_BGRA2RGBA	Convert BGRA color to RGBA color
CV_RGBA2BGRA	Convert RGBA color to BGRA color
CV_BGR2GRAY	Convert BGR color to GRAY color

CV_RGB2GRAY	Convert RGB color to GRAY color
CV_GRAY2BGR	Convert GRAY color to BGR color
CV_GRAY2RGB	Convert GRAY color to RGB color
CV_GRAY2BGRA	Convert GRAY color to BGRA color
CV_GRAY2RGBA	Convert GRAY color to RGBA color
CV_BGRA2GRAY	Convert BGRA color to GRAY color
CV_RGBA2GRAY	Convert RGBA color to GRAY color
CV_BGR2BGR565	Convert BGR color to BGR565 color
CV_RGB2BGR565	Convert RGB color to BGR565 color
CV_BGR5652BGR	Convert BGR565 color to BGR color
CV_BGR5652RGB	Convert BGR565 color to RGB color
CV_BGRA2BGR565	Convert BGRA color to BGR565 color
CV_RGBA2BGR565	Convert RGBA color to BGR565 color
CV_BGR5652BGRA	Convert BGR565 color to BGRA color
CV_BGR5652RGBA	Convert BGR565 color to RGBA color
CV_GRAY2BGR565	Convert GRAY color to BGR565 color
CV_BGR5652GRAY	Convert BGR565 color to GRAY color
CV_BGR2BGR555	Convert BGR color to BGR555 color
CV_RGB2BGR555	Convert RGB color to BGR555 color
CV_BGR5552BGR	Convert BGR555 color to BGR color
CV_BGR5552RGB	Convert BGR555 color to RGB color
CV_BGRA2BGR555	Convert BGRA color to BGR555 color
CV_RGBA2BGR555	Convert RGBA color to BGR555 color
CV_BGR5552BGRA	Convert BGR555 color to BGRA color
CV_BGR5552RGBA	Convert BGR555 color to RGBA color
CV_GRAY2BGR555	Convert GRAY color to BGR555 color
CV_BGR5552GRAY	Convert BGR555 color to GRAY color
CV_BGR2XYZ	Convert BGR color to XYZ color
CV_RGB2XYZ	Convert RGB color to XYZ color
CV_XYZ2BGR	Convert XYZ color to BGR color
CV_XYZ2RGB	Convert XYZ color to RGB color
CV_BGR2YCrCb	Convert BGR color to YCrCb color
CV_RGB2YCrCb	Convert RGB color to YCrCb color
CV_YCrCb2BGR	Convert YCrCb color to BGR color
CV_YCrCb2RGB	Convert YCrCb color to RGB color
CV_BGR2HSV	Convert BGR color to HSV color
CV_RGB2HSV	Convert RGB color to HSV color
CV_BGR2Lab	Convert BGR color to Lab color
CV_RGB2Lab	Convert RGB color to Lab color

CV_BayerBG2BGR	Convert BayerBG color to BGR color
CV_BayerGB2BGR	Convert BayerGB color to BGR color
CV_BayerRG2BGR	Convert BayerRG color to BGR color
CV_BayerGR2BGR	Convert BayerGR color to BGR color
CV_BayerBG2RGB	Convert BayerBG color to BGR color
CV_BayerGB2RGB	Convert BayerRG color to BGR color
CV_BayerRG2RGB	Convert BayerRG color to RGB color
CV_BayerGR2RGB	Convert BayerGR color to RGB color
CV_BGR2Luv	Convert BGR color to Luv color
CV_RGB2Luv	Convert RGB color to Luv color
CV_BGR2HLS	Convert BGR color to HLS color
CV_RGB2HLS	Convert RGB color to HLS color
CV_HSV2BGR	Convert HSV color to BGR color
CV_HSV2RGB	Convert HSV color to RGB color
CV_Lab2BGR	Convert Lab color to BGR color
CV_Lab2RGB	Convert Lab color to RGB color
CV_Luv2BGR	Convert Luv color to BGR color
CV_Luv2RGB	Convert Luv color to RGB color
CV_HLS2BGR	Convert HLS color to BGR color
CV_HLS2RGB	Convert HLS color to RGB color
CV_BayerBG2BGR_VNG	Convert BayerBG pattern to BGR color using VNG
CV_BayerGB2BGR_VNG	Convert BayerGB pattern to BGR color using VNG
CV_BayerRG2BGR_VNG	Convert BayerRG pattern to BGR color using VNG
CV_BayerGR2BGR_VNG	Convert BayerGR pattern to BGR color using VNG
CV_BayerBG2RGB_VNG	Convert BayerBG pattern to RGB color using VNG
CV_BayerGB2RGB_VNG	Convert BayerGB pattern to RGB color using VNG
CV_BayerRG2RGB_VNG	Convert BayerRG pattern to RGB color using VNG
CV_BayerGR2RGB_VNG	Convert BayerGR pattern to RGB color using VNG
CV_BGR2HSV_FULL	Convert BGR to HSV
CV_RGB2HSV_FULL	Convert RGB to HSV
CV_BGR2HLS_FULL	Convert BGR to HLS
CV_RGB2HLS_FULL	Convert RGB to HLS
CV_HSV2BGR_FULL	Convert HSV color to BGR color
CV_HSV2RGB_FULL	Convert HSV color to RGB color
CV_HLS2BGR_FULL	Convert HLS color to BGR color
CV_HLS2RGB_FULL	Convert HLS color to RGB color
CV_LBGR2Lab	Convert sBGR color to Lab color
CV_LRGB2Lab	Convert sRGB color to Lab color
CV_LBGR2Luv	Convert sBGR color to Luv color

CV_LRGB2Luv	Convert sRGB color to Luv color
CV_Lab2LBGR	Convert Lab color to sBGR color
CV_Lab2LRGB	Convert Lab color to sRGB color
CV_Luv2LBGR	Convert Luv color to sBGR color
CV_Luv2LRGB	Convert Luv color to sRGB color
CV_BGR2YUV	Convert BGR color to YUV
CV_RGB2YUV	Convert RGB color to YUV
CV_YUV2BGR	Convert YUV color to BGR
CV_YUV2RGB	Convert YUV color to RGB
CV_COLORCVT_MAX	The max number, do not use

```
editedImg.Image = edit.ToBitmap();
```

Digunakan untuk menampilkan gambar hasil proses *grayscale* pada **PictureBox**.

9. Tambahkan *event click* pada **Threshold** button.



```
private void thresholdBtn_Click(object sender, EventArgs e)
{
    CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);

    CvInvoke.cvThreshold(edit, edit, (double)lowValue.Value,
                        (double)highValue.Value, THRESH.CV_THRESH_BINARY);

    editedImg.Image = edit.ToBitmap();
}
```

Penjelasan:

Fungsi ini digunakan untuk melakukan proses *thresholding* pada gambar awal dengan *value* berdasarkan **NumericUpDown** *lowValue* dan *highValue* yang dimasukkan *user*.

```
CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
```

Digunakan untuk melakukan konversi warna gambar dari **Bgr** ke **Gray**.

Catatan: Dalam melakukan proses *thresholding*, untuk mendapatkan hasil *threshold* yang lebih baik maka gambar harus dijadikan *grayscale* terlebih dahulu.

```
CvInvoke.cvThreshold(edit, edit, (double)lowValue.Value,
                    (double)highValue.Value, THRESH.CV_THRESH_BINARY);
```

Digunakan untuk melakukan proses *thresholding* pada gambar yang telah di *grayscale* sebelumnya. Fungsi **cvThreshold()** memiliki parameter sebagai berikut:

- 1) **Source**: gambar awal yang akan dilakukan *thresholding*.
- 2) **Destination**: gambar tujuan hasil dari *thresholding*.
- 3) **double lowest_value**: nilai terendah *thresholding*.
- 4) **double highest_value**: nilai tertinggi *thresholding*.
- 5) **THRESH type**: tipe dari *thresholding*. **Threshold** memiliki beberapa tipe, antara lain:


Tipe Threshold	Keterangan
CV_THRESH_BINARY	value = value > threshold ? max_value : 0
CV_THRESH_BINARY_INV	value = value > threshold ? 0 : max_value
CV_THRESH_TRUNC	value = value > threshold ? threshold : value
CV_THRESH_TOZERO	value = value > threshold ? value : 0
CV_THRESH_TOZERO_INV	value = value > threshold ? 0 : value
CV_THRESH_OTSU	menggunakan algoritma Otsu untuk memilih threshold value yang optimal.

Tipe-tipe dari *thresholding* inilah, yang akan digunakan sebagai algoritma dari perhitungan besaran nilai *pixel* dari masing-masing titik pada gambar.

```
editedImg.Image = edit.ToBitmap();
```

Digunakan untuk menampilkan gambar hasil proses *thresholding* pada **PictureBox**.

10. Tambahkan *event click* pada **Smooth button**.



```
private void smoothBtn_Click(object sender, EventArgs e)
{
    if(levelValue.Text.Equals("") || levelValue.Text==null)
        MessageBox.Show("The Smooth Level Value must be filled!", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    else{
        try
        {
            int levelVal = Int32.Parse(levelValue.Text);

            if (levelVal % 2 == 0)
                MessageBox.Show("The Smooth Level Value must be odd number!", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            else
            {
                CvInvoke.cvSmooth(ori, editWithColor, SMOOTH_TYPE.CV_MEDIAN,
                    levelVal, levelVal, 0, 0);

                editedImg.Image = editWithColor.ToBitmap();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("The Smooth Level Value must be numeric!", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

Penjelasan:

Fungsi ini digunakan untuk melakukan proses *smoothing* pada gambar dengan *value* berdasarkan **TextBox** levelValue yang dimasukkan *user*.

```
if(levelValue.Text.Equals("") || levelValue.Text==null)
```

Digunakan untuk melakukan pengecekan apakah **TextBox** levelValue kosong atau tidak. Jika levelValue kosong maka program akan mengeksekusi baris selanjutnya, tetapi jika levelValue tidak kosong maka program akan mengeksekusi baris yang di dalam **else**.

```
int levelVal = Int32.Parse(levelValue.Text);
```

Digunakan untuk melakukan konversi dari tipe data **String** menjadi **Int32** (*integer*).

```
if (levelVal % 2 == 0)
```

Digunakan untuk melakukan pengecekan apakah levelVal bilangan ganjil atau genap. Jika levelVal adalah bilangan genap maka program akan mengeksekusi baris selanjutnya, tetapi jika levelVal merupakan bilangan ganjil maka program akan mengeksekusi baris yang di dalam **else**.

```
CvInvoke.cvSmooth(ori, editWithColor, SMOOTH_TYPE.CV_MEDIAN,
                  levelVal, levelVal, 0, 0);
```

Digunakan untuk melakukan *smoothing* terhadap gambar. Fungsi **cvSmooth()** memiliki parameter sebagai berikut:

- 1) **Source:** *image* awal yang akan di-*smoothing* (variabel awal dengan tipe data **Image <Bgr, byte>**).
- 2) **Destination:** target *image* hasil *smoothing* (variabel tujuan dengan tipe data **Image <Bgr, byte>**).
- 3) **SMOOTH_TYPE type:** tipe dari *smoothing*. **Smooth** memiliki beberapa tipe, antara lain:

Tipe Smooth	Keterangan
CV_BLUR_NO_SCALE	(<i>simple blur with no scaling</i>) – penjumlahan semua sekeliling <i>pixel</i> dari param1 x param2. Jika nilai sekeliling dari <i>pixel</i> berbeda, maka akan di <i>precompute integral image</i> dengan menggunakan fungsi cvIntegral .
CV_BLUR	(<i>simple blur</i>) – penjumlahan semua sekeliling <i>pixel</i> dari param1 x param 2 yang diikuti perubahan ukuran dengan rumus berikut: $\frac{1}{\text{param1} \times \text{param2}}$
CV_GAUSSIAN	(<i>gaussian blur</i>) – membelit gambar dengan param1 x param2 <i>gaussian kernel</i> .
CV_MEDIAN	(<i>median blur</i>) – mencari median dari sekeliling <i>pixel</i> param1 x param2, dengan contoh sekeliling <i>pixel</i> adalah <i>square</i> .
CV_BILATERAL	(<i>bilateral filter</i>) – penerapan <i>filtering bilateral</i> matriks 3x3 dengan <i>color sigma</i> = param1 dan <i>space sigma</i> = param2.

- 4) **int param1**: parameter pertama dari operasi *smoothing*, harus berupa angka ganjil agar sekeliling *pixel* yang digunakan untuk operasi *smoothing simetris* dengan *pixel*.
- 5) **int param2**: parameter kedua dari operasi *smoothing* (untuk *simple blur / simple blur with no scaling* dan *gaussian blur*, jika $\text{param2} = 0$ maka $\text{param2} = \text{param1}$).
- 6) **double param3**: parameter ketiga dari operasi *smoothing* atau disebut sigma1 untuk arah horizontal (untuk *gaussian kernel*, parameter ini dapat menentukan sigma *gaussian [standard derivation]*. Jika $\text{param3} = 0$, maka sigma akan dihitung dari *size kernel* dengan rumus berikut:

$$\text{Sigma} = (\text{n}/2 - 1) * 0.3 + 0.8$$

dimana:

$\text{n} = \text{param1}$ untuk *horizontal kernel*.

$\text{n} = \text{param2}$ untuk *vertical kernel*.

Catatan: Performa untuk *kernel* yang kecil (matriks 3x3 hingga 7x7) akan lebih baik dengan sigma *standard*.

Jika param3 bukan 0, tetapi param1 dan param2 yang bernilai 0 maka *size kernel* akan dihitung dari sigma [agar operasi cukup akurat]).

- 7) **double param4**: parameter keempat dari operasi *smoothing* atau disebut sigma2 untuk arah vertikal (untuk *gaussian kernel* yang *non-square*, parameter ini dapat digunakan untuk menentukan sigma *gaussian* yang berbeda dari param3 secara vertikal).

```
editedImg.Image = editWithColor.ToBitmap();
```

Digunakan untuk menampilkan gambar hasil proses *smoothing* pada **PictureBox**.

11. Tambahkan *event click* pada *menu item* Exit.

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Penjelasan:

Fungsi untuk keluar dari aplikasi.

Berikut ini adalah keseluruhan *code* dari *code* di atas:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;

namespace Introduction_EmguCV
{
    public partial class Form1 : Form
    {
        Image<Bgr, Byte> ori, editWithColor;
        Image<Gray, Byte> edit;

        public Form1()
        {
            InitializeComponent();
        }

        private void loadImageToolStripMenuItem_Click(object sender, EventArgs e)
        {
            DialogResult result = openFileDialog1.ShowDialog();

            if (result == System.Windows.Forms.DialogResult.OK)
            {
                originalImg.Image = Image.FromFile(openFileDialog1.FileName);

                ori = new Image<Bgr, byte>(new Bitmap(originalImg.Image));
                edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
                editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
            }
        }
    }
}
```



```
private void grayScaleBtn_Click(object sender, EventArgs e)
{
    CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);

    editedImg.Image = edit.ToBitmap();
}

private void thresholdBtn_Click(object sender, EventArgs e)
{
    CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);

    CvInvoke.cvThreshold(edit, edit, (double)lowValue.Value,
        (double)highValue.Value, THRESH.CV_THRESH_BINARY);

    editedImg.Image = edit.ToBitmap();
}

private void smoothBtn_Click(object sender, EventArgs e)
{
    if(levelValue.Text.Equals("") || levelValue.Text==null)
        MessageBox.Show("The Smooth Level Value must be filled!", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);

    else{
        try
        {
            int levelVal = Int32.Parse(levelValue.Text);

            if (levelVal % 2 == 0)
                MessageBox.Show("The Smooth Level Value must be odd number!", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);

            else
            {
                CvInvoke.cvSmooth(ori, editWithColor, SMOOTH_TYPE.CV_MEDIAN,
                    levelVal, levelVal, 0, 0);

                editedImg.Image = editWithColor.ToBitmap();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("The Smooth Level Value must be numeric!", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
}
```

Chapter 03

Edge Detection



UNIVERSITY

SOFTWARE LABORATORY CENTER

3.1. Canny Edge Detection

Canny Edge Detection adalah sebuah metode pengenalan garis/tepi yang akan menghasilkan garis-garis *pixel* pada ujung (tepi) dari suatu daerah yang tebal. *Canny Edge Detection Thinning Algorithm* merupakan metode yang melakukan deteksi terhadap ujung (tepi) dari suatu *image*.

Memiliki keunggulan karena:

1. *Smooth image* dengan **Gaussian** mengoptimalkan *trade-off* antara *noise filtering* dan *edge localization*.
2. Menghitung **Gradient Magnitude** menggunakan pendekatan parsial derivatif 2x2 *filter*.
3. *Edges by line* menerapkan penekanan *non-maximal* dengan besarnya *gradient*.
4. Deteksi tepi dengan **thresholding** ganda.

3.2. Sobel Edge Detection

Sobel edge detection adalah salah satu metode dalam *image processing* yang berguna untuk mendeteksi tepi (*edge*) suatu objek dalam gambar digital. *Edge* dapat terjadi karena adanya perubahan atau perbedaan (*gradient*) nilai *pixel* yang cukup berpengaruh antara suatu *pixel* terhadap *pixel* yang berada di sekitarnya.

Secara umum, **sobel edge detection** dibedakan menjadi dua macam, yakni:

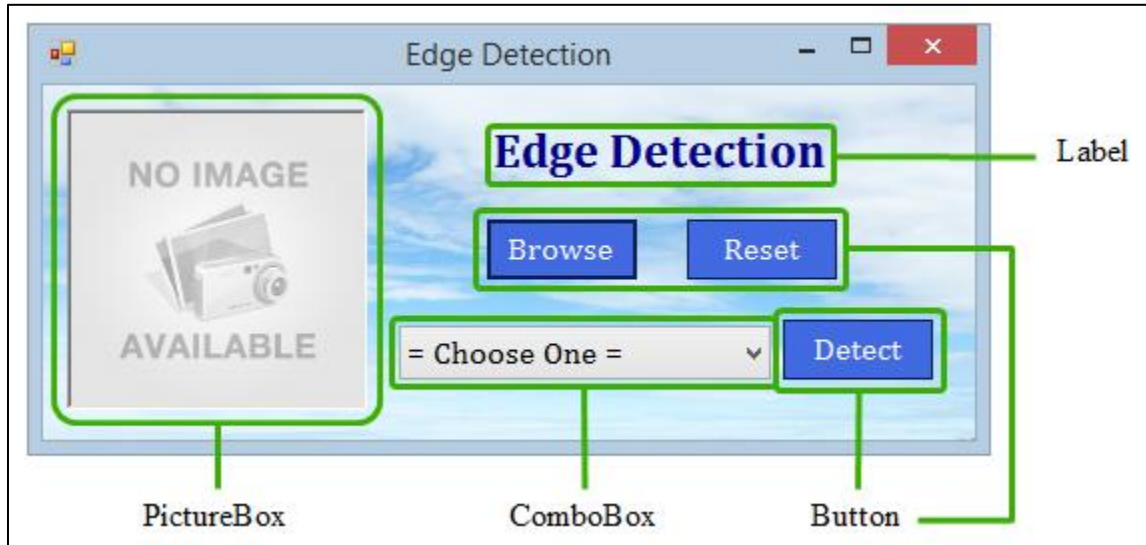
- 1) **Sobel horizontal**: dimana pencarian *edge* dilakukan searah sumbu x gambar.
- 2) **Sobel vertikal**: dimana pencarian *edge* dilakukan searah sumbu y gambar.

3.3. Laplace Edge Detection

Laplace edge detection menggunakan *matrix Laplace 5x5*. Dimana *matrix Laplace 5x5* yang digunakan adalah *convolution mask* untuk mendekati turunan kedua, tidak seperti metode **Sobel** yang mendekati *gradient*. Dan bukannya 2 *mask*, 3x3 *matrix Sobel*, satu untuk arah x dan y, **Laplace** menggunakan 15x5 *masker* untuk turunan 2 baik diarah x dan y. Namun, karena pelindung yang mendekati ukuran turunan kedua pada gambar, maka mereka sangat peka terhadap *noise*.

3.4. Exercise

Buatlah sebuah program untuk melakukan operasi *canny*, *sobel*, dan *laplace edge detection*, seperti gambar berikut ini:



Keterangan:

Pada **ComboBox** terdapat 4 *items*, yakni: “= Choose One =”, “Canny Detection”, “Sobel Detection”, dan “Laplace Detection”.

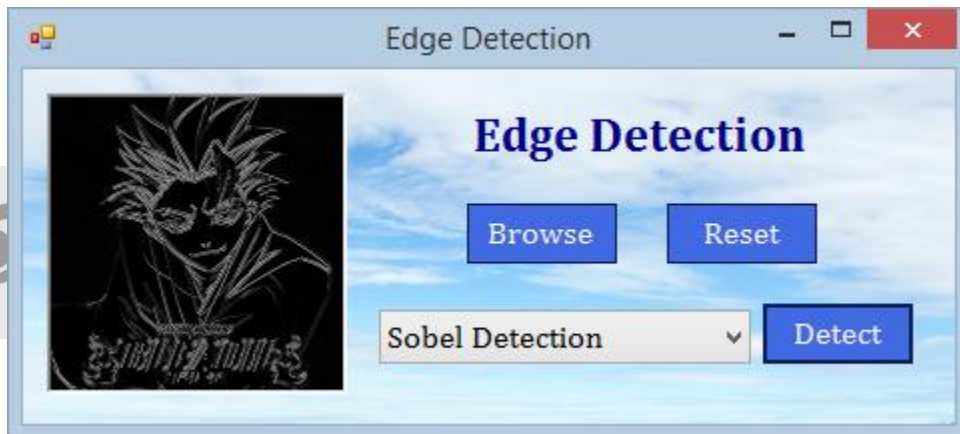
Setelah **Browse Image**:



Hasil dari **Canny Edge Detection**:



Hasil dari **Sobel Edge Detection** :



Hasil dari **Laplace Edge Detection** :



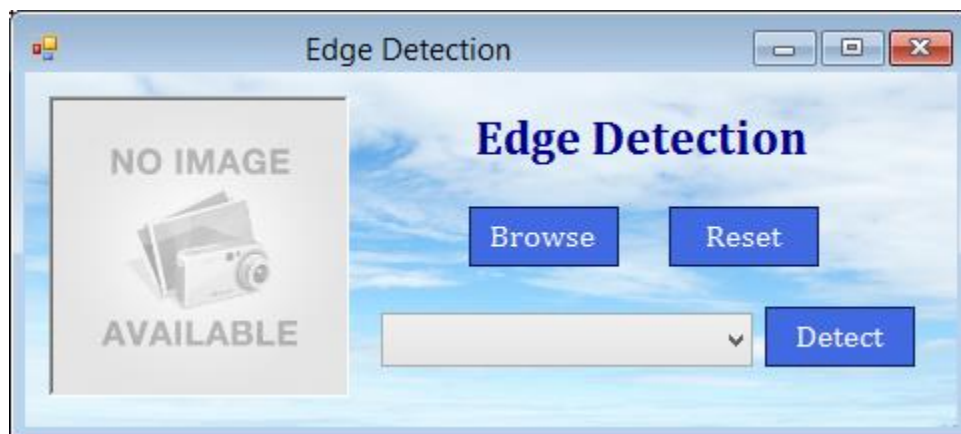
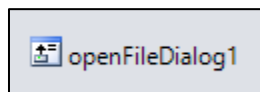
Jawaban:

Berikut merupakan langkah-langkah pengerjaan:

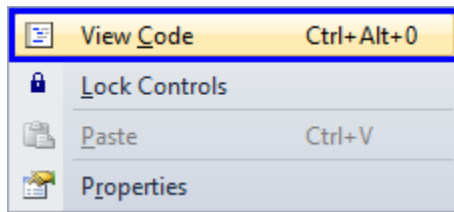
1. Buat *project* baru dari **Microsoft Visual Studio 2010 C#**. Cara membuat *project* baru dan mengkonfigurasi **EmguCV** masih sama seperti yang telah kita pelajari pada **Chapter I**.
2. Lalu, ubah **Size form** menjadi *width* = 482 dan *height* = 216, ganti **Text** menjadi “*Edge Detection*”, ubah **BackgroundImage** dengan *image* yang diperlukan, dan ubah **BackgroundImageLayout** menjadi “*Stretch*”.



3. Masukkan semua komponen yang dibutuhkan, seperti: **OpenFileDialog**, **Label**, **PictureBox**, **Button**, dan **ComboBox**. Lalu, atur *properties* yang diperlukan untuk setiap komponen.



4. Klik kanan pada *form* dan pilih **View Code**.



5. *Import namespace* yang dibutuhkan untuk **EmguCV**.

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
```

Penjelasan:

Untuk menggunakan fungsi-fungsi **EmguCV**, diperlukan *namespace* berikut:

- *Namespace* **Emgu.CV**, berguna sebagai *wrapper* dari OpenCV *image processing function*.
- *Namespace* **Emgu.CV.CvEnum**, berguna sebagai *wrapper* untuk OpenCV *enumeration*.
- *Namespace* **Emgu.CV.Structure**, berguna sebagai *wrapper* dari OpenCV *structure*.

6. Buat *variable global* yang akan digunakan.

```
Image<Bgr, Byte> ori, editWithColor;
Image<Gray, Byte> edit;
```

Penjelasan:

Image adalah suatu *class* untuk **IplImage** dan mendefinisikan **TColor** sebagai tipe warna untuk **IplImage** dan **TDepth** sebagai *depth* dari **IplImage**.

Bgr adalah suatu *class* yang mendefinisikan warna *blue green red*.

Gray adalah suatu *class* yang mendefinisikan warna *gray*.

Byte adalah suatu *class* yang mendefinisikan jenis *depth* dalam ukuran *byte*.


7. Tambahkan *set properties* **ComboBox** pada **Constructor Form**.

```
public Form1()
{
    InitializeComponent();
    choiceComboBox.SelectedIndex = 0;
}
```

Penjelasan:

Code di atas digunakan untuk melakukan *set* pada *properties* **ComboBox** sehingga `choiceComboBox` akan menampilkan isi *item* indeks ke-0 ketika program dijalankan.

8. Tambahkan *event click* pada **Browse button**.



```
private void button1_Click(object sender, EventArgs e)
{
    DialogResult result = openFileDialog1.ShowDialog();

    if (result == System.Windows.Forms.DialogResult.OK)
    {
        imgPictureBox.Image = Image.FromFile(openFileDialog1.FileName);

        ori = new Image<Bgr, byte>(new Bitmap(imgPictureBox.Image));
        edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
        editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
    }
}
```

Penjelasan:

DialogResult merupakan representasi hasil dari *form* yang digunakan sebagai *dialog box*.

```
DialogResult result = openFileDialog1.ShowDialog();
```

Digunakan untuk menampilkan **OpenFileDialog** dan menampung hasilnya ke **DialogResult**.

```
if (result == System.Windows.Forms.DialogResult.OK)
```

Digunakan untuk memastikan apakah *user* telah mengklik tombol **OK** pada **OpenFileDialog** atau belum. Jika *user* telah mengklik tombol **OK** maka program akan mengeksekusi baris selanjutnya.

```
imgPictureBox.Image = Image.FromFile(openFileDialog1.FileName);
```

Digunakan untuk menampilkan *image* yang dipilih dari **OpenFileDialog** ke **PictureBox**.

```
ori = new Image<Bgr, byte>(new Bitmap(imgPictureBox.Image));
```

Digunakan untuk membuat *object* dari **Image** <**Bgr**, **byte**> yang berisi gambar awal.

```
edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
```

Digunakan untuk membuat *object* dari **Image** dengan warna **Gray** yang ukurannya disesuaikan dengan gambar awal.

```
editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
```

Digunakan untuk membuat *object* **Image** dengan warna **Bgr** yang ukurannya disesuaikan dengan gambar awal.

9. Tambahkan *event click* pada **Detect** button.

```
private void detectBtn_Click(object sender, EventArgs e)
{
    if (choiceComboBox.SelectedIndex == 0)
    {
        MessageBox.Show("Please choose an action first!!!", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
    Image<Gray, float> edit2, result;
    switch (choiceComboBox.SelectedIndex)
    {
        case 1:
            CvInvoke.cvCanny(edit, edit, 100, 30, 3);
            imgPictureBox.Image = edit.ToBitmap();
            break;
        case 2:
            edit2 = edit.Convert<Gray, float>();
            result = edit2.Sobel(1, 0, 5);
            imgPictureBox.Image = result.ToBitmap();
            break;
        case 3:
            edit2 = edit.Convert<Gray, float>();
            result = edit2.Laplace(5);
            imgPictureBox.Image = result.ToBitmap();
            break;
    }
}
```

Penjelasan:

Fungsi ini digunakan untuk melakukan proses *edge detection* pada gambar dengan tipe *detection* berdasarkan pilihan *user* pada **ComboBox**.

```
if (choiceComboBox.SelectedIndex == 0)
```

Digunakan untuk melakukan pengecekan apakah *item* yang sedang dipilih *user* pada **ComboBox** merupakan *item* indeks ke-0 atau bukan. Jika iya, maka program akan menampilkan pesan dan keluar dari fungsi tanpa menjalankan *code* setelah pengecekan.

```
CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
```

Digunakan untuk melakukan konversi warna gambar dari **Bgr** ke **Gray**.

Catatan: Dalam melakukan proses *edge detection*, untuk mendapatkan hasil terbaik maka gambar harus dijadikan *grayscale* terlebih dahulu.

```
CvInvoke.cvCanny(edit, edit, 100, 30, 3);
```

Digunakan untuk melakukan *canny edge detection* pada gambar. Fungsi **cvCanny()** memiliki parameter sebagai berikut:

- 1) **Image:** *image* awal yang akan dilakukan *canny detection*.
- 2) **Edges:** *image* untuk menyimpan *edges* yang ditemukan oleh fungsi.
- 3) **double threshold1:** nilai *threshold* pertama dari *canny detection* atau disebut juga nilai *threshold* terendah. Biasanya memiliki perbandingan 1:2 atau 1:3 dengan nilai *threshold* terbesar.
- 4) **double threshold2:** nilai *threshold* kedua dari *canny detection* atau disebut juga nilai *threshold* terbesar.
- 5) **int aperture_size:** besaran *matrix* untuk parameter besaran matrik **canny** yang digunakan (nilai *default* adalah 3. Nilai lain yang dapat digunakan adalah 1, 3, 5, dan 7).

```
edit2 = edit.Convert<Gray, float>();
```

Digunakan untuk melakukan pengubahan *depth* gambar dari *byte* menjadi *float* karena sobel dan laplace membutuhkan *depth* gambar yang lebih dalam yaitu 16 bit sedangkan *byte* hanya 8 bit.

```
result = edit2.Sobel(1, 0, 5);
```

Digunakan untuk melakukan *sobel edge detection* pada gambar dan ditampung ke dalam **result**. Fungsi **Sobel()** memiliki parameter sebagai berikut:

- 1) **int xorder:** nilai orde untuk turunan x (biasa *xorder* = 0 untuk keadaan normal dan *yorder* = 1, ataupun sebaliknya).
- 2) **int yorder:** nilai orde untuk turunan y (jika *yorder* = 1 untuk keadaan normal, maka *xorder* = 0, ataupun sebaliknya).

- 3) **int aperture_size**: ukuran perpanjangan kernel *sobel* (harus bernilai 1, 3, 5, atau 7). Dalam semua kasus kecuali untuk nilai `aperture_size = 1`, `aperture_size x aperture_size` kernel yang dipisahkan akan digunakan untuk menghitung turunan.

```
result = edit2.Laplace(5);
```

Digunakan untuk melakukan *laplace edge detection* pada gambar dan ditampung ke dalam *result*.. Fungsi **Laplace()** memiliki parameter sebagai berikut:

- 1) **int aperture_size**: besaran matriks untuk parameter besaran matriks **laplacian** yang digunakan (nilai yang dapat digunakan adalah 1, 3, 5, dan 7).

```
imgPictureBox.Image = edit.ToBitmap();
```

Digunakan untuk menampilkan gambar hasil proses *canny* pada **PictureBox**.



10. Tambahkan *event click* pada **Reset** button.

```
private void resetBtn_Click(object sender, EventArgs e)
{
    imgPictureBox.Image = ori.ToBitmap();
}
```

Penjelasan:

Fungsi ini digunakan untuk menampilkan *image* awal pada **PictureBox**.

Berikut ini adalah keseluruhan *code* dari *code* di atas:

```
Image<Bgr, byte> ori, editWithColor;
Image<Gray, byte> edit;

public Form1()
{
    InitializeComponent();
    choiceComboBox.SelectedIndex = 0;
}
```



```
private void button1_Click(object sender, EventArgs e)
{
    DialogResult result = openFileDialog1.ShowDialog();

    if (result == System.Windows.Forms.DialogResult.OK)
    {
        imgPictureBox.Image = Image.FromFile(openFileDialog1.FileName);

        ori = new Image<Bgr, byte>(new Bitmap(imgPictureBox.Image));
        edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
        editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
    }
}

private void detectBtn_Click(object sender, EventArgs e)
{
    if (choiceComboBox.SelectedIndex == 0)
    {
        MessageBox.Show("Please choose an action first!!!", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;
    }

    CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
    Image<Gray, float> edit2, result;
    switch (choiceComboBox.SelectedIndex)
    {
        case 1:
            CvInvoke.cvCanny(edit, edit, 100, 30, 3);
            imgPictureBox.Image = edit.ToBitmap();
            break;
        case 2:
            edit2 = edit.Convert<Gray, float>();
            result = edit2.Sobel(1, 0, 5);
            imgPictureBox.Image = result.ToBitmap();
            break;
        case 3:
            edit2 = edit.Convert<Gray, float>();
            result = edit2.Laplace(5);
            imgPictureBox.Image = result.ToBitmap();
            break;
    }
}
```

```
private void resetBtn_Click(object sender, EventArgs e)
{
    imgPictureBox.Image = ori.ToBitmap();
}
```

Chapter 04

Shape Detection

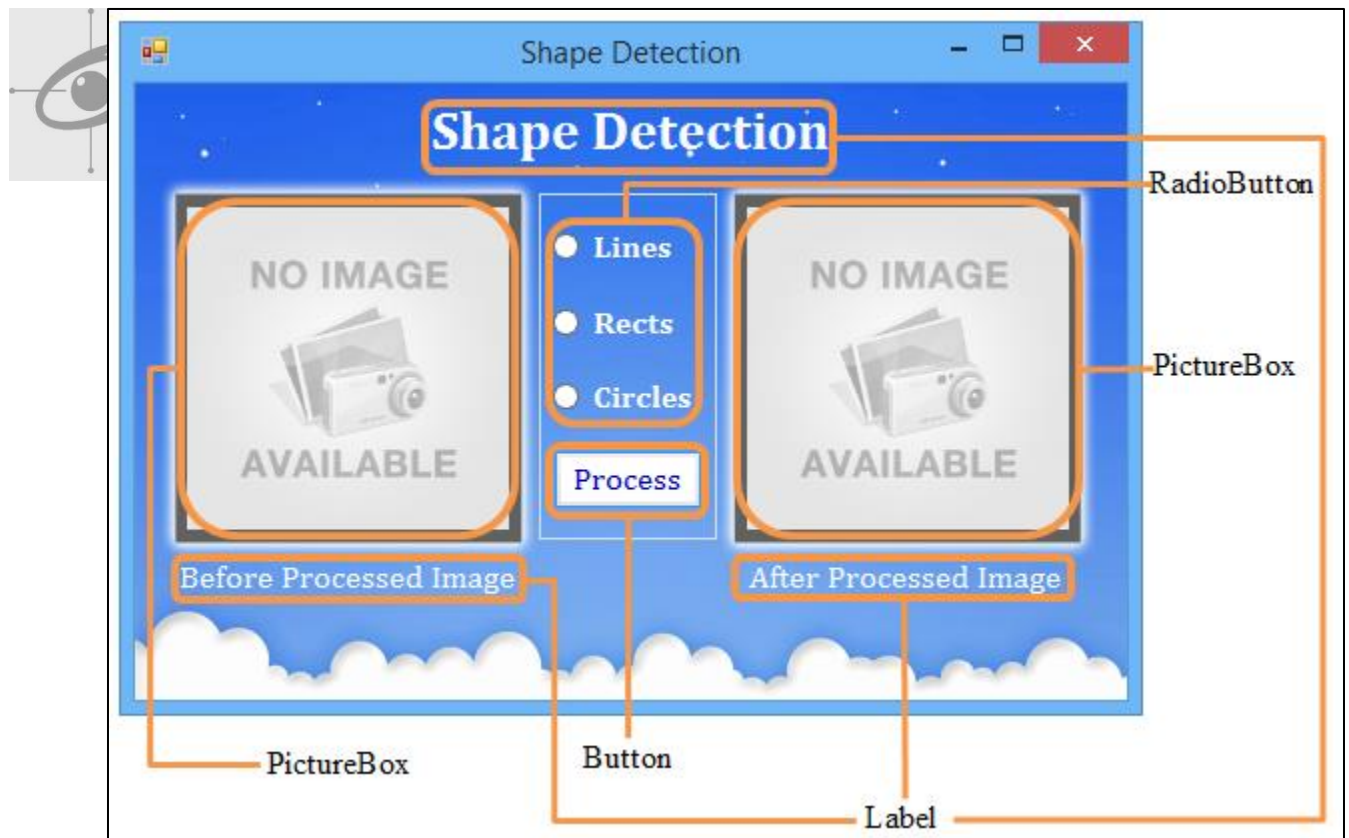


4.1. Shape Detection

Shape Detection umumnya mencari fitur *shape* yang efektif dan penting berdasarkan *shape boundary* atau *boundary* yang ditambahkan konten didalamnya. Berbagai fitur dapat dikembangkan untuk mendeteksi *shape*. Teknik yang banyak digunakan dalam mendeteksi *shape* yakni *shape signature*, *signature histogram*, *shape invariants*, *moments*, *curvature*, *shape context*, *shape matrix*, dan lain sebagainya. Pada umumnya, semua teknik representasi *shape* terbagi menjadi 2 kelompok, yakni berbasis **contours** dan berbasis **wilayah**. Pembelajaran ini akan membahas teknik berbasis *contours* menggunakan metode **Hough Transform** dalam mendeteksi *lines*, *circles*, and *rectangles* shape.

4.2. Exercise

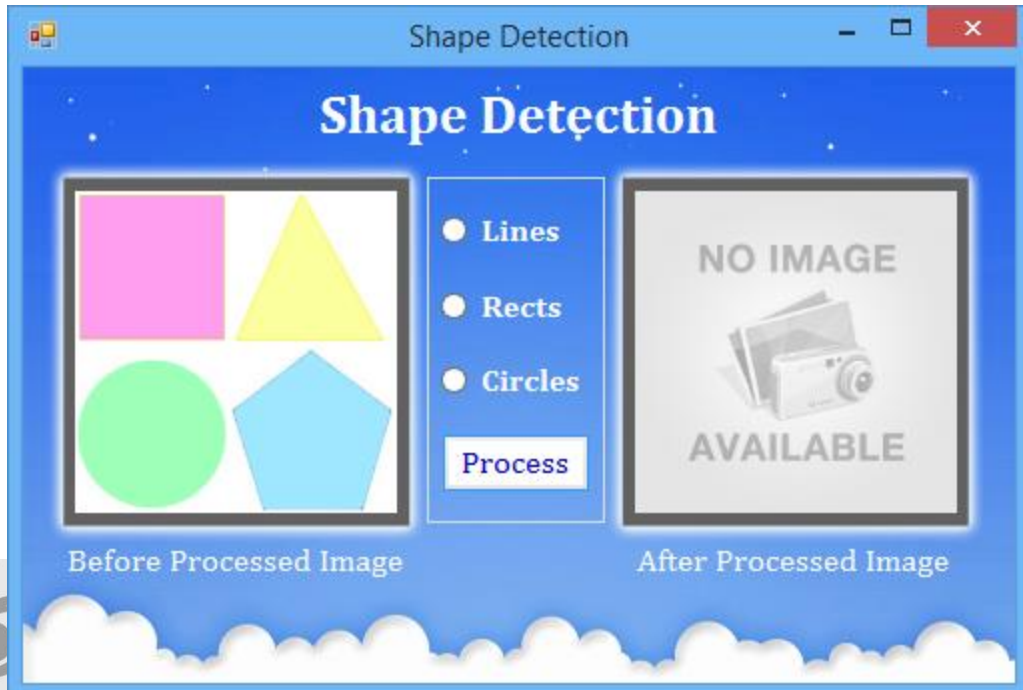
Buatlah sebuah program untuk melakukan operasi *lines*, *rectangles*, dan *circles* *shape detection*, seperti gambar berikut ini:



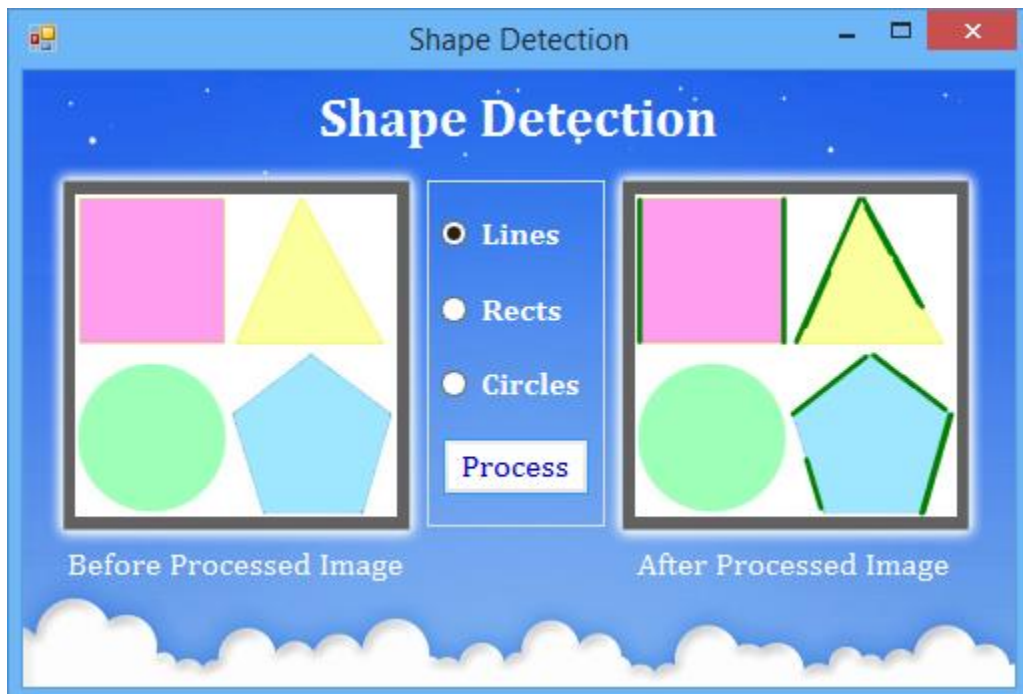
Keterangan:

Jika *image* pada **pictureBox** sebelah kiri (*before processed image*) di klik, maka *event load image* akan dijalankan.

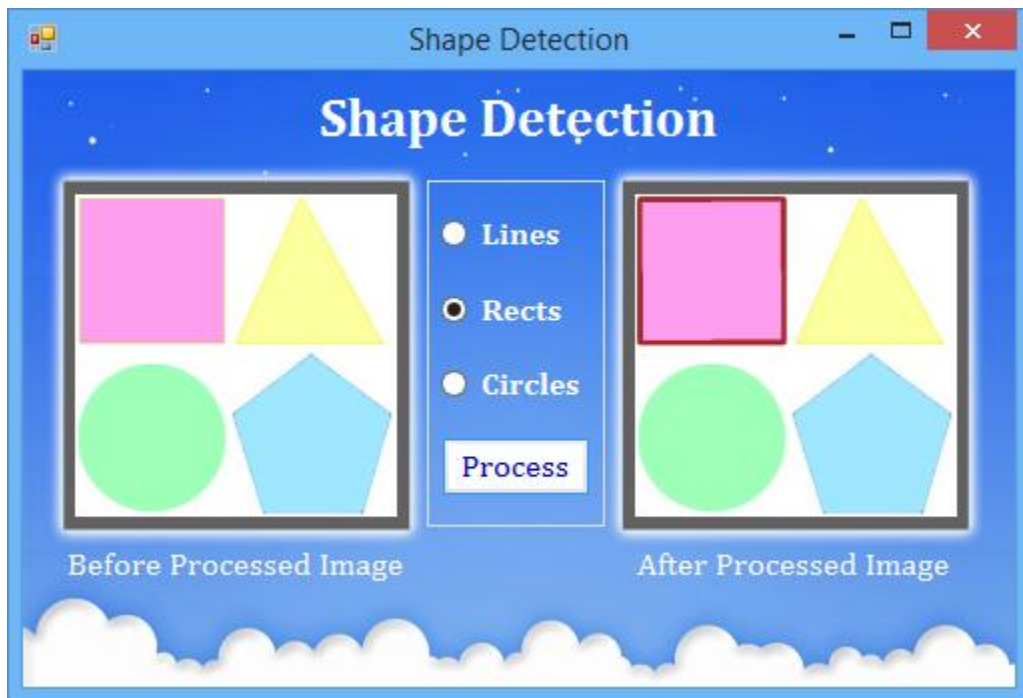
Setelah **Browse Image**:



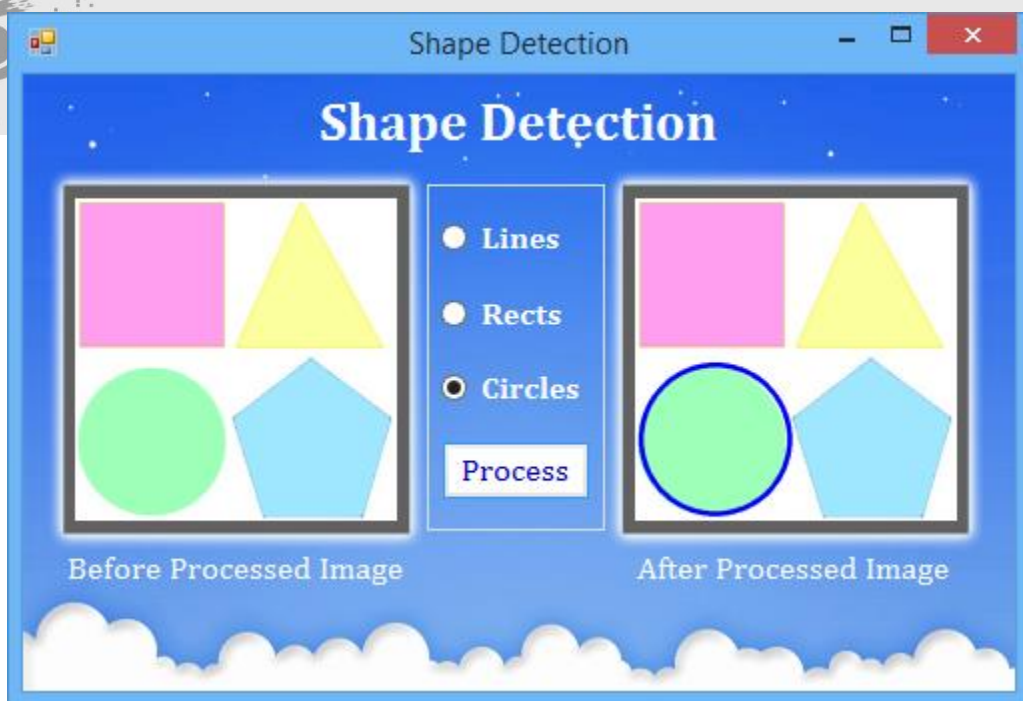
Hasil dari **Lines Detection**:



Hasil dari **Rectangles Detection** :



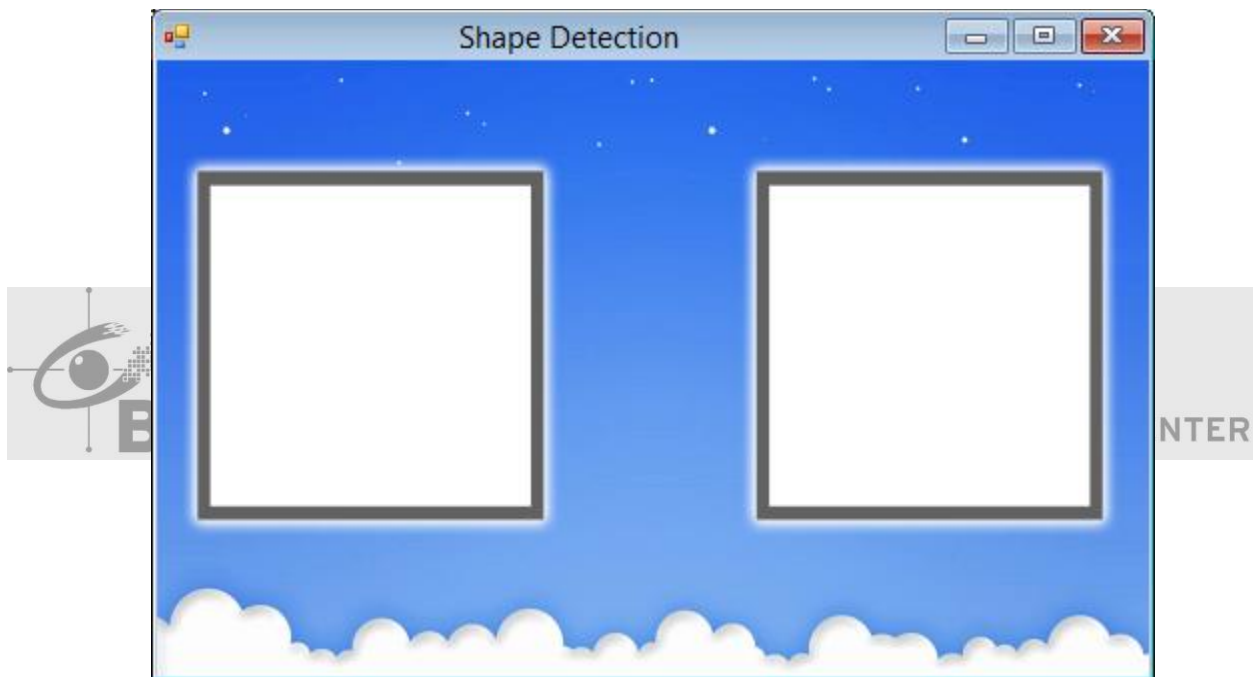
Hasil dari **Circles Detection** :



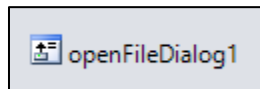
Jawaban:

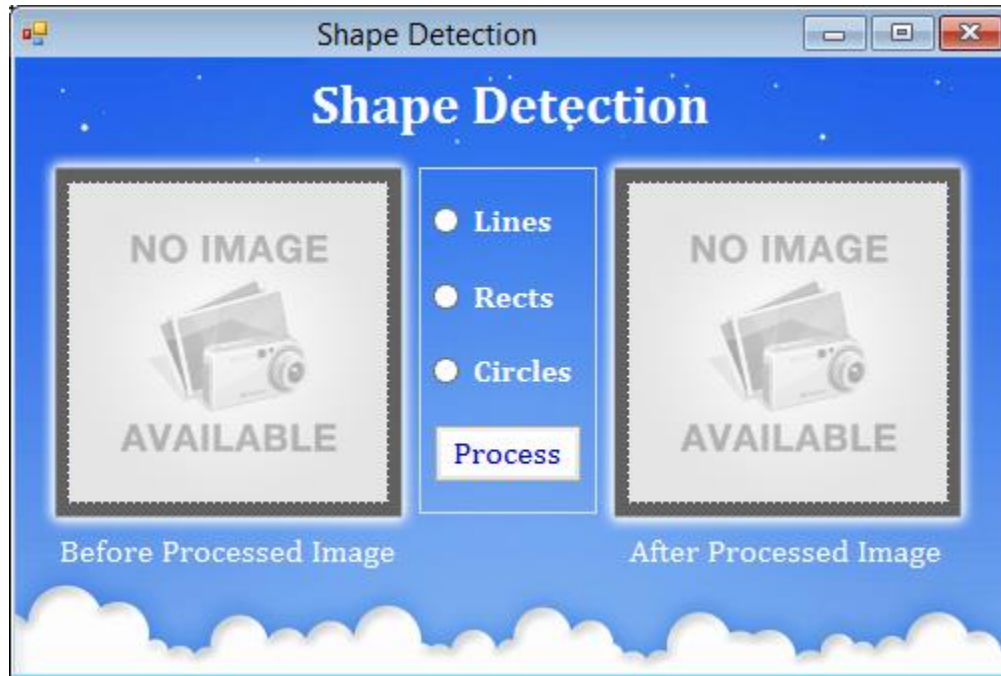
Berikut merupakan langkah-langkah pengerjaan:

1. Buat *project* baru dari **Microsoft Visual Studio 2010 C#**. Cara membuat *project* baru dan mengkonfigurasi **EmguCV** masih sama seperti yang telah kita pelajari pada **Chapter I**.
2. Lalu, ubah **Size form** menjadi *width* = 502 dan *height* = 337, ganti **Text** menjadi “*Shape Detection*”, ubah **BackgroundImage** dengan *image* yang diperlukan, dan ubah **BackgroundImageLayout** menjadi “*Stretch*”.

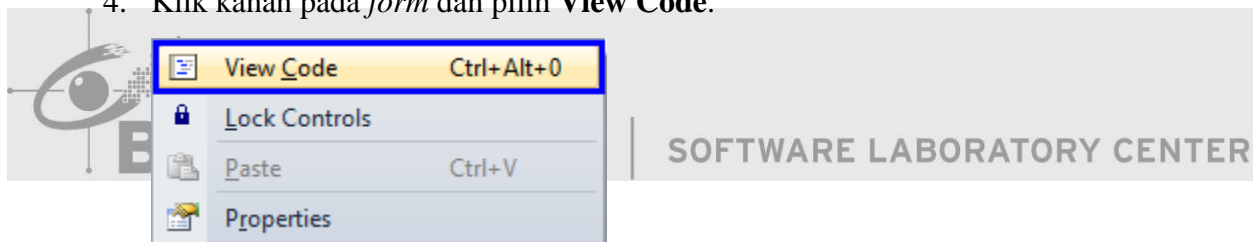


3. Masukkan semua komponen yang dibutuhkan, seperti: **OpenFileDialog**, **Label**, **PictureBox**, **GroupBox**, **RadioButton**, dan **Button**. Lalu, atur *properties* yang diperlukan untuk setiap komponen.





4. Klik kanan pada *form* dan pilih **View Code**.



5. *Import namespace* yang dibutuhkan untuk **EmguCV**.

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
```

Penjelasan:

Untuk menggunakan fungsi-fungsi **EmguCV**, diperlukan *namespace* berikut:

- *Namespace* **Emgu.CV**, berguna sebagai *wrapper* dari OpenCV *image processing function*.
- *Namespace* **Emgu.CV.CvEnum**, berguna sebagai *wrapper* untuk OpenCV *enumeration*.
- *Namespace* **Emgu.CV.Structure**, berguna sebagai *wrapper* dari OpenCV *structure*.

6. Buat *variable global* yang akan digunakan.

```
Image<Bgr, Byte> ori, editWithColor;
Image<Gray, Byte> edit;
```

Penjelasan:


Image adalah suatu *class* untuk **IplImage** dan mendefinisikan **TColor** sebagai tipe warna untuk **IplImage** dan **TDepth** sebagai *depth* dari **IplImage**.

Bgr adalah suatu *class* yang mendefinisikan warna *blue green red*.

Gray adalah suatu *class* yang mendefinisikan warna *gray*.

Byte adalah suatu *class* yang mendefinisikan jenis *depth* dalam ukuran *byte*.

7. Tambahkan *event click* pada **originalImage** pictureBox.



```
private void originalImage_Click(object sender, EventArgs e)
{
    DialogResult result = openFileDialog1.ShowDialog();

    if (result == System.Windows.Forms.DialogResult.OK)
    {
        originalImage.Image = Image.FromFile(openFileDialog1.FileName);

        ori = new Image<Bgr, byte>(new Bitmap(originalImage.Image));
        edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
        editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
    }
}
```

Penjelasan:

DialogResult merupakan representasi hasil dari *form* yang digunakan sebagai *dialog box*.

```
DialogResult result = openFileDialog1.ShowDialog();
```

Digunakan untuk menampilkan **OpenFileDialog** dan menampung hasilnya ke **DialogResult**.


```
if (result == System.Windows.Forms.DialogResult.OK)
```

Digunakan untuk memastikan apakah *user* telah mengklik tombol **OK** pada **OpenFileDialog** atau belum. Jika *user* telah mengklik tombol **OK** maka program akan mengeksekusi baris selanjutnya.

```
originalImage.Image = Image.FromFile(openFileDialog1.FileName);
```

Digunakan untuk menampilkan *image* yang dipilih dari **OpenFileDialog** ke **PictureBox**.

```
ori = new Image<Bgr, byte>(new Bitmap(originalImage.Image));
```

Digunakan untuk membuat *object* dari **Image** <**Bgr**, **byte**> yang berisi gambar awal.

```
edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
```

Digunakan untuk membuat *object* dari **Image** dengan warna **Gray** yang ukurannya disesuaikan dengan gambar awal.

```
editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
```

Digunakan untuk membuat *object* **Image** dengan warna **Bgr** yang ukurannya disesuaikan dengan gambar awal.

8. Tambahkan *event click* pada **Process button**.

```
private void processBtn_Click(object sender, EventArgs e)
{
    if (!linesRadioBtn.Checked &&
        !rectsRadioBtn.Checked &&
        !circlesRadioBtn.Checked)
    {
        MessageBox.Show("Action must be selected!", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
}
```



```

CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
CvInvoke.cvCanny(edit, edit, 150, 60, 3);

editWithColor = ori.Copy();

if (linesRadioBtn.Checked) //lines
{
    LineSegment2D[] lines=edit.HoughLinesBinary(2,Math.PI/45,90,50,10)[0];

    foreach (LineSegment2D l in lines)
    {
        editWithColor.Draw(l, new Bgr(Color.Green), 5);
    }
}
else if (rectsRadioBtn.Checked) //rectangles
{
    List<MCvBox2D> boxList = new List<MCvBox2D>();
    MemStorage storage = new MemStorage();

    for (Contour<Point> con=edit.FindContours();con!=null;con=con.HNext)
    {
        Contour<Point> con2 = con.ApproxPoly(con.Perimeter*0.16,storage);

        if (con.Area > 250)
        {
            if (con2.Total == 4)
            {
                bool rect = true;
                Point[] pts = con2.ToArray();
                LineSegment2D[] edges = PointCollection.PolyLine(pts, true);

                for (int i = 0; i < edges.Length; i++)
                {
                    double angle=Math.Abs(
                        edges[(i+1)%edges.Length].
                        GetExteriorAngleDegree(edges[i]));

                    if (angle < 80 || angle > 100)
                    {
                        rect = false;
                        break;
                    }
                }
                if (rect)
                {
                    boxList.Add(con2.GetMinAreaRect());
                }
            }
        }
    }
}

```



```

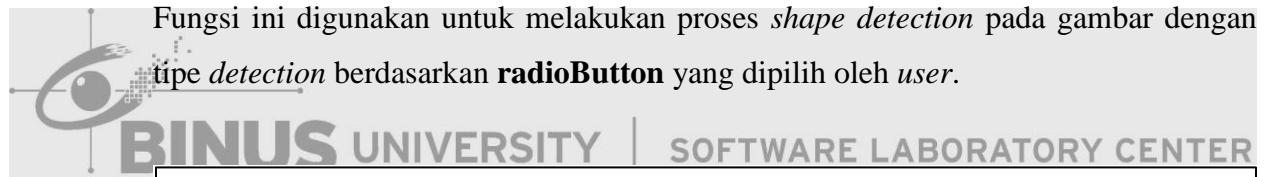
        foreach (MCvBox2D b in boxList)
        {
            editWithColor.Draw(b, new Bgr(Color.Brown), 5);
        }
    }
    else //circles
    {
        CircleF[] circle=edit.HoughCircles(new Gray(100),new Gray(80),
                                            20,400,50,0)[0];

        foreach (CircleF c in circle)
        {
            editWithColor.Draw(c, new Bgr(Color.Blue), 5);
        }
    }

    editedImage.Image = editWithColor.ToBitmap();
}
}
}

```

Penjelasan:



Fungsi ini digunakan untuk melakukan proses *shape detection* pada gambar dengan tipe *detection* berdasarkan **radioButton** yang dipilih oleh *user*.

```

if (!linesRadioBtn.Checked &&
    !rectsRadioBtn.Checked &&
    !circlesRadioBtn.Checked)

```

Digunakan untuk melakukan pengecekan apakah salah satu dari **radioButton** telah dipilih oleh *user* atau belum. Jika tidak ada satu pun **radioButton** yang terpilih, maka program akan menampilkan pesan dan keluar dari fungsi tanpa menjalankan *code* setelah pengecekan.

```

CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);

```

Digunakan untuk melakukan konversi warna gambar dari **Bgr** ke **Gray**.

Catatan: Dalam melakukan proses *shape detection*, gambar harus dijadikan *grayscale* terlebih dahulu.

```
CvInvoke.cvCanny(edit, edit, 150, 60, 3);
```

Digunakan untuk melakukan *canny edge detection* pada gambar.

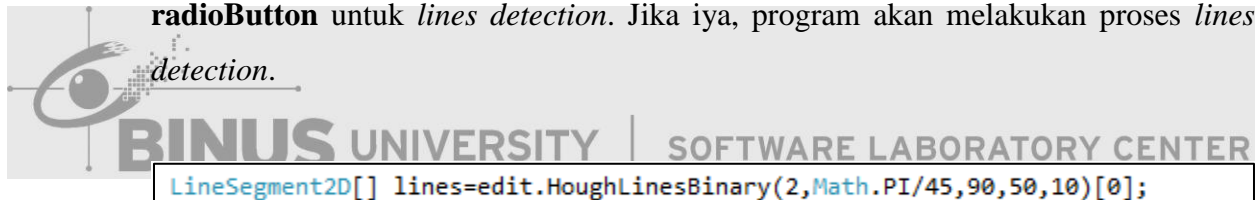
Catatan: Dalam melakukan proses *shape detection*, gambar harus telah dilakukan proses *canny* terlebih dahulu.

```
editWithColor = ori.Copy();
```

Digunakan untuk meng-copy image **ori** ke **editWithColor** sehingga, untuk menampilkan hasil proses *shape detection* dapat menggunakan **editWithColor** dan image **ori** tidak akan ada perubahan.

```
if (linesRadioBtn.Checked) //lines
```

Digunakan untuk melakukan pengecekan apakah **radioButton** yang dipilih adalah **radioButton** untuk *lines detection*. Jika iya, program akan melakukan proses *lines detection*.



```
LineSegment2D[] lines=edit.HoughLinesBinary(2,Math.PI/45,90,50,10)[0];
```

Digunakan untuk melakukan *lines detection* pada gambar kemudian ditampung pada **LineSegment2D[]**. Fungsi **HoughLinesBinary()** memiliki parameter berikut:

- 1) **double rhoResolution**: resolusi jarak dalam satuan piksel yang terkait.
- 2) **double thetaResolution**: resolusi sudut diukur dalam radian.
- 3) **int threshold**: nilai *threshold* (jika nilai akumulator yang sesuai lebih besar daripada nilai *threshold*, maka fungsi akan mengembalikan *line*).
- 4) **double minLineWidth**: nilai minimum *width* sebuah *line*.
- 5) **double gapBetweenLines**: nilai minimum jarak/*gap* antar *lines*.

```
foreach (LineSegment2D l in lines)
```

Digunakan untuk melakukan perulangan sebanyak jumlah isi dari *lines* agar menampilkan *lines* yang terdeteksi dengan sesuai.

```
editWithColor.Draw(1, new Bgr(Color.Green), 5);
```

Digunakan untuk menggambar *line* yang terdeteksi pada *image* **editWithColor**. Fungsi **Draw()** memiliki parameter sebagai berikut:

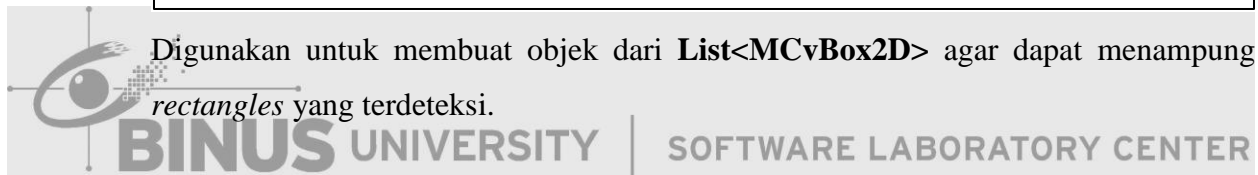
- 1) **LineSegment2D line**: segmen *line* yang ingin digambarkan.
- 2) **TColor color**: warna untuk gambar segmen *line*.
- 3) **int thickness**: ketebalan segmen *line* yang akan digambarkan.

```
else if (rectsRadioBtn.Checked) //rectangles
```

Digunakan untuk melakukan pengecekan apakah **radioButton** yang dipilih adalah **radioButton** untuk *rectangles detection*. Jika iya, program akan melakukan proses *rectangles detection*.

```
List<MCvBox2D> boxList = new List<MCvBox2D>();
```

Digunakan untuk membuat objek dari **List<MCvBox2D>** agar dapat menampung *rectangles* yang terdeteksi.



```
MemStorage storage = new MemStorage();
```

Digunakan untuk menampung hasil *sequence* yang digunakan dalam mendeteksi *rectangle*.

```
for (Contour<Point> con=edit.FindContours(); con!=null; con=con.HNext)
```

Digunakan untuk melakukan perulangan dalam mencari kontur yang menggunakan fungsi **FindContours()**.

```
Contour<Point> con2 = con.ApproxPoly(con.Perimeter*0.16, storage);
```

Digunakan untuk menentukan hasil *approximation curves* dengan menggunakan fungsi **ApproxPoly()** dan menampungnya ke **Contour<Point>**. Fungsi **ApproxPoly()** memiliki parameter sebagai berikut:

- 1) **double accuracy**: akurasi *approximation* yang diinginkan.
- 2) **int maxLevel**: maksimum level untuk nilai *approximation* kontur (bersifat opsional).
- 3) **MemStorage storage**: tempat hasil *sequence* yang digunakan.

```
if (con.Area > 250)
```

Digunakan untuk mengecek apakah area kontur yang terdeteksi lebih besar daripada 250. Jika tidak, maka tidak terdeteksi kontur yang diinginkan.

```
if (con2.Total == 4)
```

Digunakan untuk mengecek apakah terdeteksi *shape* dengan 4 sudut. Jika tidak, maka bukan *rectangle* yang terdeteksi.

```
bool rect = true;
```

Digunakan untuk variabel *flag* yang akan menentukan apakah yang terdeteksi adalah *rectangle* atau bukan.

```
Point[] pts = con2.ToArray();
```

Digunakan untuk menampung hasil *approximation curves*.

```
LineSegment2D[] edges = PointCollection.PolyLine(pts, true);
```

Digunakan untuk menentukan garis dari **Point[]**.

```
for (int i = 0; i < edges.Length; i++)
```

Digunakan untuk melakukan perulangan sebanyak jumlah isi **edges**.

```
double angle=Math.Abs(
    edges[(i+1)%edges.Length].
    GetExteriorAngleDegree(edges[i]));
```

Digunakan untuk menentukan sudut dari *rectangle* yang terdeteksi.

```
if (angle < 80 || angle > 100)
```

Digunakan untuk mengecek apakah sudut yang terdeteksi lebih kecil dari 80 atau lebih besar dari 100. Jika iya, maka *shape* yang terdeteksi bukanlah *rectangle*.

```
boxList.Add(con2.GetMinAreaRect());
```

Digunakan untuk memasukkan *rectangles* yang terdeteksi ke dalam **List<MCvBox2D>**.

```
foreach (MCvBox2D b in boxList)
```

Digunakan untuk melakukan perulangan sebanyak jumlah isi dari *boxList* agar menampilkan *rectangles* yang terdeteksi dengan sesuai.

```
editWithColor.Draw(b, new Bgr(Color.Brown), 5);
```

Digunakan untuk menggambarkan *rectangles* yang terdeteksi pada *image* **editWithColor**. Fungsi **Draw()** memiliki parameter sebagai berikut:

- 1) **MCvBox3D rectangle**: *rectangles* yang ingin digambarkan.
- 2) **TColor color**: warna untuk *rectangles* yang digambarkan.
- 3) **int thickness**: ketebalan *rectangles* yang digambarkan. Jika ketebalan kurang dari 1, maka *rectangle* yang digambarkan akan diisi penuh.

```
else //circles
```

Digunakan untuk melakukan pengecekan apakah **radioButton** yang dipilih adalah **radioButton** untuk *circles detection*. Jika iya, program akan melakukan proses *circles detection*.

```
CircleF[] circle=edit.HoughCircles(new Gray(100),new Gray(80),
                                   20,400,50,0)[0];
```

Digunakan untuk melakukan *circles detection* pada gambar kemudian ditampung pada **CircleF[]**. Fungsi **HoughCircles()** memiliki parameter berikut:

- 1) **TColor cannyThreshold**: nilai *threshold* tertinggi dari dua yang dilewati untuk *canny edge detector* (yang lebih rendah akan dua kali lebih kecil).
- 2) **TColor accumulatorThreshold**: *threshold* akumulator deteksi pada bagian tengah.
- 3) **double dp**: resolusi akumulator yang digunakan untuk mendeteksi titik tengah *circle*.
- 4) **double minDist**: jarak minimum antar titik tengah *circle* yang terdeteksi.
- 5) **int minRadius**: radius minimum untuk mencari *circle*.
- 6) **int maxRadius**: radius maksimum untuk mencari *circle*.

```
foreach (CircleF c in circle)
```

Digunakan untuk melakukan perulangan sebanyak jumlah isi dari *circle* agar menampilkan *circles* yang terdeteksi dengan sesuai.

```
editWithColor.Draw(c, new Bgr(Color.Blue), 5);
```

Digunakan untuk menggambarkan *circles* yang terdeteksi pada *image editWithColor*. Fungsi **Draw()** memiliki parameter sebagai berikut:

- 1) **CircleF circle**: *circles* yang ingin digambarkan.
- 2) **TColor color**: warna untuk *circles* yang digambarkan.
- 3) **int thickness**: ketebalan *circles* yang digambarkan. Jika ketebalan kurang dari 1, maka *circle* yang digambarkan akan diisi penuh.

```
editedImage.Image = editWithColor.ToBitmap();
```

Digunakan untuk menampilkan gambar hasil proses *shape detection* pada **editedImage** pictureBox.

Berikut ini adalah keseluruhan *code* dari *code* di atas:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;

namespace Chapter_4
{
    public partial class Form1 : Form
    {
        Image<Bgr, byte> ori, editWithColor;
        Image<Gray, byte> edit;

        public Form1()
        {
            InitializeComponent();
        }

        private void originalImage_Click(object sender, EventArgs e)
        {
            DialogResult result = openFileDialog1.ShowDialog();

            if (result == System.Windows.Forms.DialogResult.OK)
            {
                originalImage.Image = Image.FromFile(openFileDialog1.FileName);

                ori = new Image<Bgr, byte>(new Bitmap(originalImage.Image));
                edit = new Image<Gray, byte>(ori.Size.Width, ori.Size.Height);
                editWithColor = new Image<Bgr, byte>(ori.Size.Width, ori.Size.Height);
            }
        }

        private void processBtn_Click(object sender, EventArgs e)
        {
            if (!linesRadioBtn.Checked &&
                !rectsRadioBtn.Checked &&
                !circlesRadioBtn.Checked)
            {
                MessageBox.Show("Action must be selected!", "Error",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);

                return;
            }
        }
    }
}
```

```

CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
CvInvoke.cvCanny(edit, edit, 150, 60, 3);

editWithColor = ori.Copy();

if (linesRadioBtn.Checked) //lines
{
    LineSegment2D[] lines=edit.HoughLinesBinary(2,Math.PI/45,90,50,10)[0];

    foreach (LineSegment2D l in lines)
    {
        editWithColor.Draw(l, new Bgr(Color.Green), 5);
    }
}
else if (rectsRadioBtn.Checked) //rectangles
{
    List<MCvBox2D> boxList = new List<MCvBox2D>();
    MemStorage storage = new MemStorage();

    for (Contour<Point> con=edit.FindContours();con!=null;con=con.HNext)
    {
        Contour<Point> con2 = con.ApproxPoly(con.Perimeter*0.16,storage);

        if (con.Area > 250)
        {
            if (con2.Total == 4)
            {
                bool rect = true;
                Point[] pts = con2.ToArray();
                LineSegment2D[] edges = PointCollection.PolyLine(pts, true);

                for (int i = 0; i < edges.Length; i++)
                {
                    double angle=Math.Abs(
                        edges[(i+1)%edges.Length].
                        GetExteriorAngleDegree(edges[i]));

                    if (angle < 80 || angle > 100)
                    {
                        rect = false;
                        break;
                    }
                }

                if (rect)
                {
                    boxList.Add(con2.GetMinAreaRect());
                }
            }
        }
    }
}

```

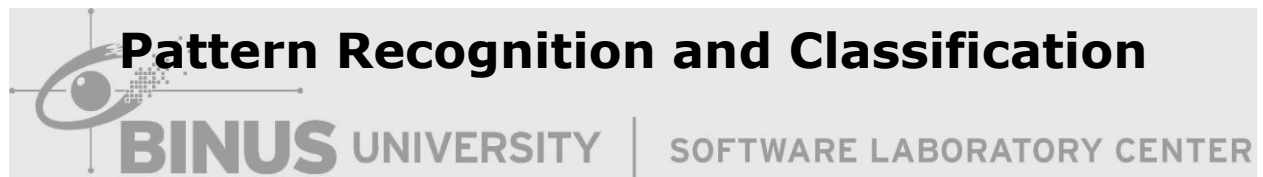


```
foreach (MCvBox2D b in boxList)
{
    editWithColor.Draw(b, new Bgr(Color.Brown), 5);
}
else //circles
{
    CircleF[] circle=edit.HoughCircles(new Gray(100),new Gray(80),20,400,50,0)[0];

    foreach (CircleF c in circle)
    {
        editWithColor.Draw(c, new Bgr(Color.Blue), 5);
    }
}

editedImage.Image = editWithColor.ToBitmap();
}
}
```

Chapter 05



5.1. Pattern Recognition and Classification

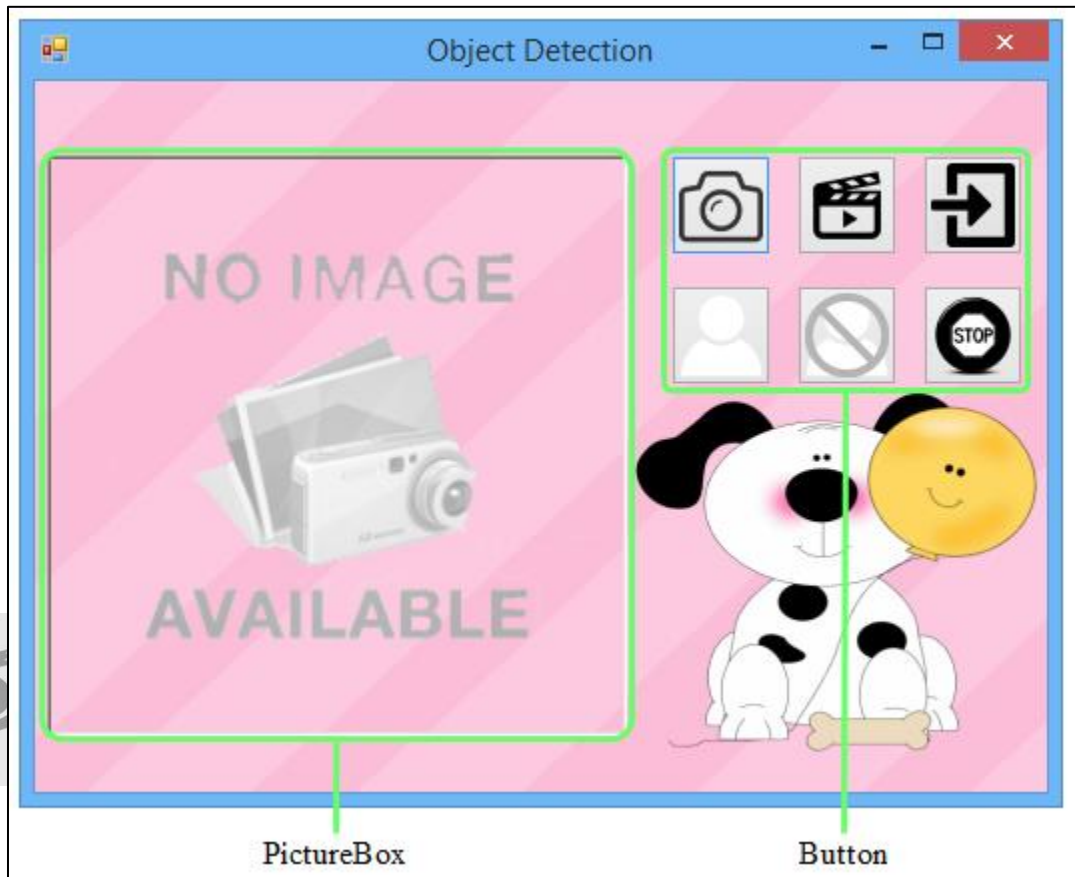
Pattern Recognition adalah penugasan semacam nilai *output* (atau label) untuk nilai *input* yang diberikan (atau misalnya), menurut beberapa algoritma tertentu. Sebuah contoh dari **Pattern Recognition** adalah **Classification**, yang mencoba untuk memberikan setiap nilai *input* ke salah satu dari satu *hardware* (misalnya, menentukan apakah *email* yang diberikan adalah "*spam*" atau "*non-spam*").

Namun, **Pattern Recognition** adalah masalah yang lebih umum yang meliputi jenis lain *output* juga. Contoh lain adalah regresi, yang memberikan *output real*-nilai untuk setiap *input*, pelabelan urutan, yang memberikan kelas untuk setiap anggota urutan nilai (misalnya, bagian dari penandaan pidato, yang memberikan bagian dari pidato untuk setiap kata di sebuah kalimat *input*), dan *parsing*, yang memberikan pohon *parse* ke input kalimat, menggambarkan struktur sintaksis kalimat.

Di dalam *computer vision* sendiri, pengenalan pola sangat sering digunakan untuk melakukan beberapa deteksi bentuk gambar. Antara lain adalah pengenalan bentuk wajah, bentuk badan, dan lainnya. Hal ini menggunakan metode pengenalan pola (**Pattern Recognition**) yang telah diberikan pengenalan sebelumnya ke dalam suatu jaringan *neural*.

5.2. Exercise

Buatlah sebuah program untuk melakukan operasi *face detection*, seperti gambar berikut ini:



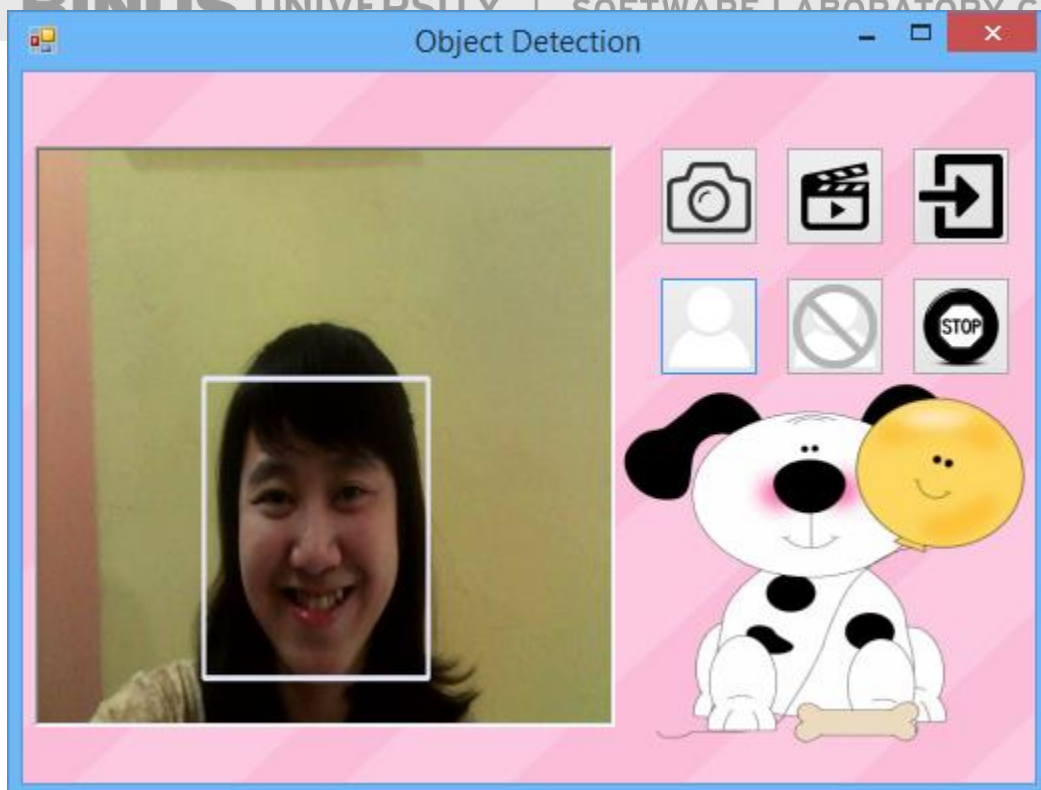
Keterangan:

	cameraLoadBtn button Load dari webcam (real time)		faceDetectBtn button Melakukan <i>face detection</i>
	videoLoadBtn button Load dari video		noFaceDetectBtn button Berhenti melakukan <i>face detection</i>
	exitBtn button Keluar dari aplikasi		stopBtn button Berhenti me-load dari webcam atau video

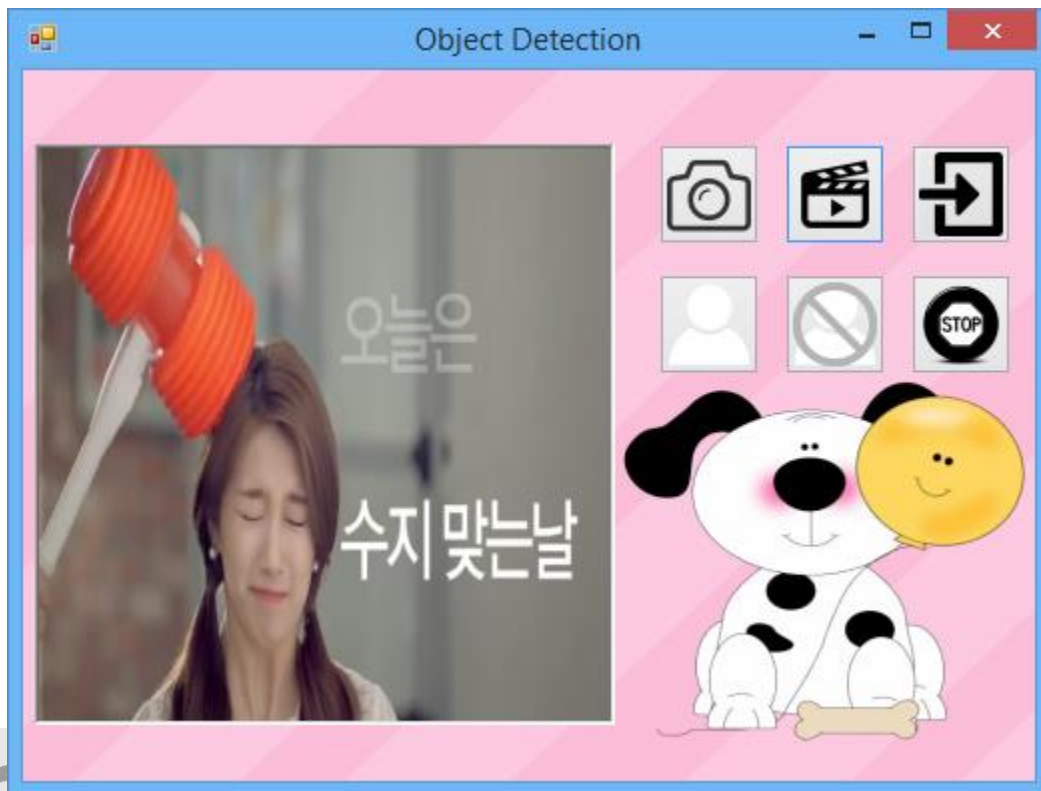
Setelah *load* dari webcam:



Hasil dari **Face Detection** (*load* dari webcam):



Setelah *load* dari **video**:



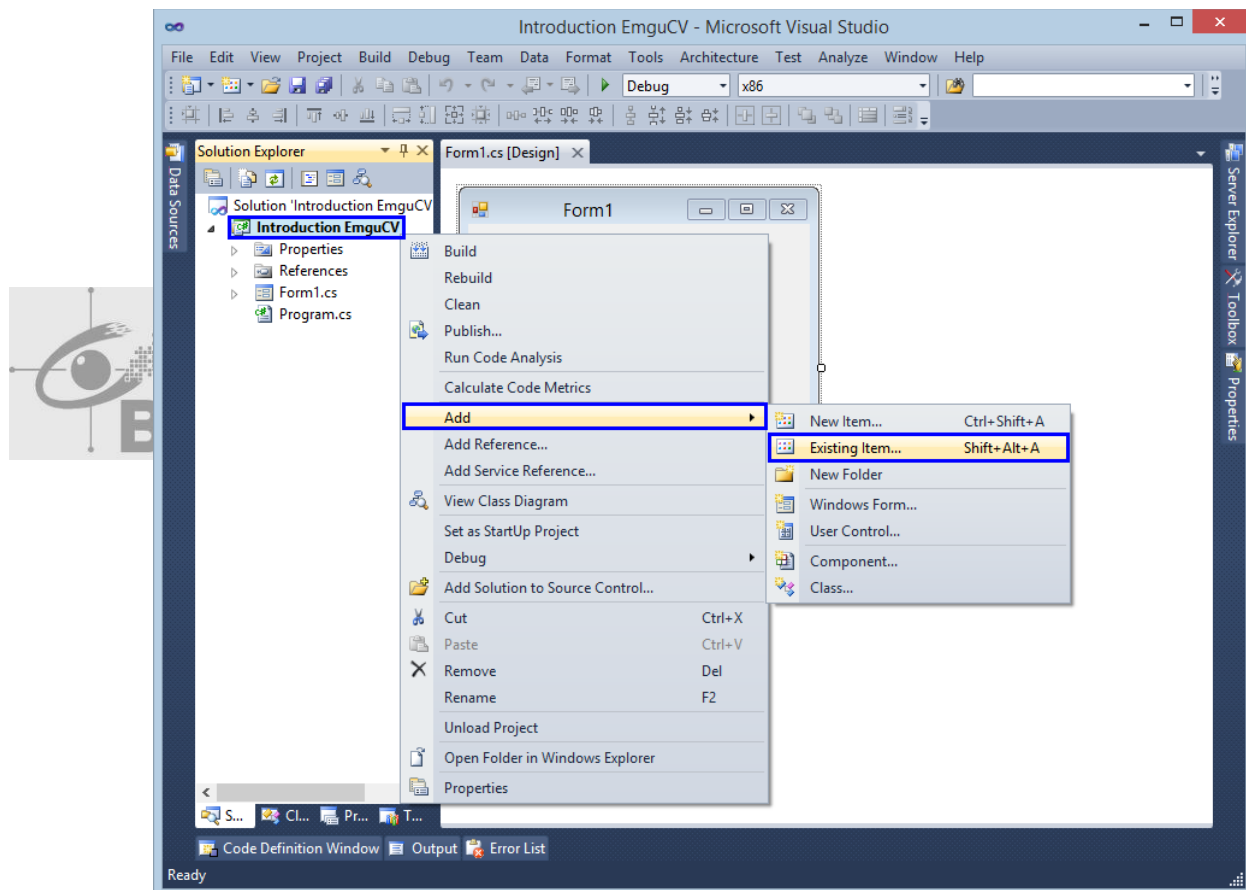
Hasil dari **Face Detection** (*load* dari **video**):



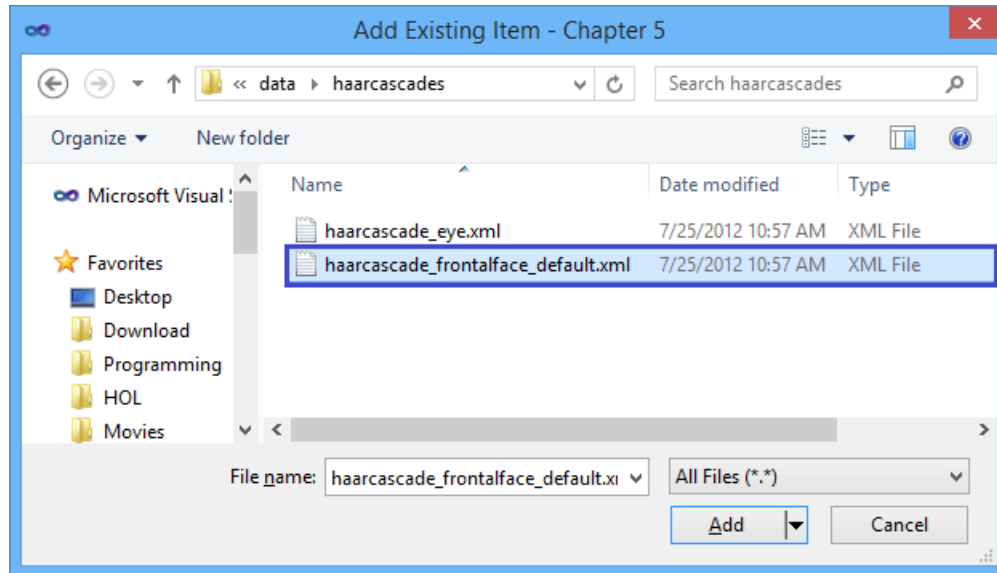
Jawaban:

Berikut merupakan langkah-langkah pengerjaan:

1. Buat *project* baru dari **Microsoft Visual Studio 2010 C#**. Cara membuat *project* baru dan mengkonfigurasi **EmguCV** masih sama seperti yang telah kita pelajari pada **Chapter I**.
2. Tambahkan *file xml* yang akan digunakan dalam melakukan *face detection* dengan cara: pada **Solution Explorer**, klik kanan di **Project**, klik **Add** dan klik **Existing Item**.



3. Pilih hasil *extract libemgucv-windows-x86-2.3.0.1416* yang di folder **libemgucv-libemgucv-windows-x86-2.3.0.1416 > opencv > data > haarcascades**, kemudian *select file xml* yang akan digunakan (dalam pembelajaran ini menggunakan *file xml* “**haarcascade_frontalface_default.xml**”, dikarenakan objek yang akan dideteksi adalah wajah). Lalu, klik tombol **Add**.

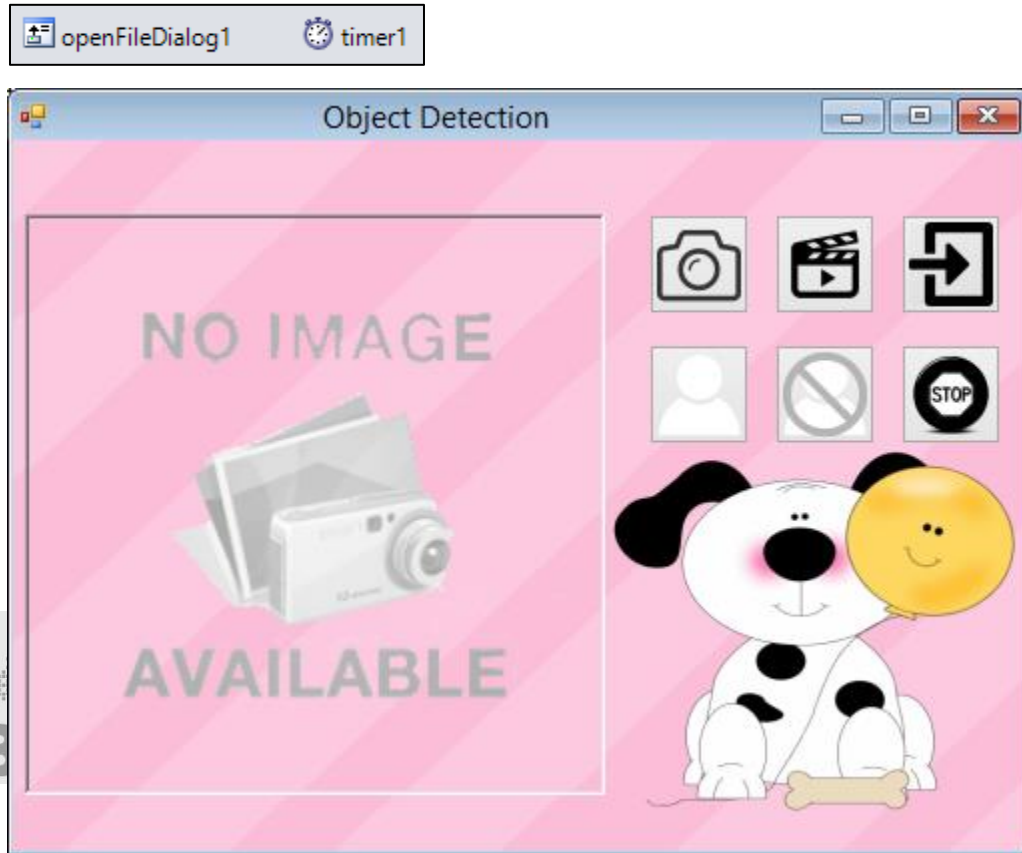


Catatan: pada jawaban ini, setelah “**haarcascade_frontalface_default.xml**” berhasil ditambahkan, maka *file* tersebut akan di-*rename* menjadi “**face_haarcascade.xml**”.

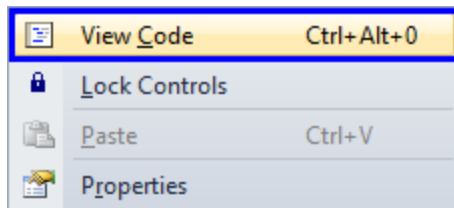
4. Lalu, ubah **Size form** menjadi *width* = 512 dan *height* = 384, ganti **Text** menjadi “*Object Detection*”, ubah **BackgroundImage** dengan *image* yang diperlukan, dan ubah **BackgroundImageLayout** menjadi “*Stretch*”.



5. Masukkan semua komponen yang dibutuhkan, seperti: **OpenFileDialog**, **Timer**, **PictureBox**, dan **Button**. Lalu, atur *properties* yang diperlukan untuk setiap komponen.



6. Klik kanan pada *form* dan pilih **View Code**.



7. *Import namespace* yang dibutuhkan untuk **EmguCV**.

```
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
```

Penjelasan:

Untuk menggunakan fungsi-fungsi **EmguCV**, diperlukan *namespace* berikut:

- *Namespace* **Emgu.CV**, berguna sebagai *wrapper* dari OpenCV *image processing function*.
- *Namespace* **Emgu.CV.Structure**, berguna sebagai *wrapper* dari OpenCV *structure*.
- *Namespace* **Emgu.CV.CvEnum**, berguna sebagai *wrapper* untuk OpenCV *enumeration*.

8. Buat *variable global* yang akan digunakan.

```
Capture cap = null;
bool isInProgress = false,
    isFaceDetection=false;
int cameraDevice = 0,
    fps = 30;
Image<Bgr, Byte> frame;
Image<Gray, Byte> edit;
HaarCascade cascade;
```

Penjelasan:

Capture adalah suatu *class* untuk menangkap *images* dari kamera atau *file* video.

Image adalah suatu *class* untuk **IplImage** dan mendefinisikan **TColor** sebagai tipe warna untuk **IplImage** dan **TDepth** sebagai *depth* dari **IplImage**.

Bgr adalah suatu *class* yang mendefinisikan warna *blue green red*.

Gray adalah suatu *class* yang mendefinisikan warna *gray*.

Byte adalah suatu *class* yang mendefinisikan jenis *depth* dalam ukuran *byte*.

HaarCascade adalah suatu *class* untuk *cascade* yang akan digunakan dalam mendeteksi objek.

9. Tambahkan *create object* dari class **HaarCascade** dan *set properties* **Timer** pada **Constructor Form**.

```
public Form1()
{
    InitializeComponent();
    cascade = new HaarCascade("../..\\face_haarcascade.xml");
    timer1.Interval = 1000 / fps;
}
```

Penjelasan:

Code di atas digunakan untuk *create object* dari class **HaarCascade** sehingga *cascade* telah dapat digunakan ketika mendeteksi wajah. Selain itu, digunakan juga untuk melakukan *set* pada *properties* **Timer** sehingga dalam 1 detik akan menjalankan 30 *frames* (jumlah *frame* per detik telah di *set* sebelumnya pada variabel **fps**).

10. Kemudian, buatlah *method* **faceDetection**.



```
void faceDetection(Image<Bgr,byte> ori)
{
    edit = new Image<Gray, byte>(ori.Size.Width,ori.Size.Height);
    CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
    var faces = cascade.Detect(edit,1.8,2,
        HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,new Size(ori.Width,ori.Height));
    foreach (var face in faces)
    {
        ori.Draw(face.rect, new Bgr(Color.Lavender), 3);
    }
}
```

Penjelasan:

Method ini akan digunakan untuk mendeteksi wajah ketika *method* ini dipanggil.

```
edit = new Image<Gray, byte>
    (ori.Size.Width,ori.Size.Height);
```

Digunakan untuk membuat *object* dari **Image** dengan warna **Gray** yang ukurannya disesuaikan dengan gambar **ori** yang berasal dari parameter yang dikirimkan.

```
CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);
```

Digunakan untuk melakukan konversi warna gambar dari **Bgr** ke **Gray**.

Catatan: Dalam melakukan proses *face detection*, gambar harus dijadikan *grayscale* terlebih dahulu.

```
var faces = cascade.Detect(edit,1.8,2,
    HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,new Size(ori.Width,ori.Height));
```

Digunakan untuk melakukan *face detection* dan hasilnya akan ditampung ke **MCvAvgComp[]**. Fungsi **Detect ()** memiliki parameter sebagai berikut:

- 1) **Source:** *image* yang akan dilakukan *face detection* (dengan tipe data **Image <Gray, byte>**).
- 2) **double scaleFactor:** faktor yang menentukan *image* di-scale antara *scan* berikutnya. (misalnya: jika **scaleFactor** = 1.1 maka *image* akan di-scale sebesar 10%).
- 3) **int minNeighbors:** jumlah minimum *neighbor rectangles* yang membentuk objek (minimum nilai **minNeighbors** = -1). Jika kelompok *rectangles* jumlahnya kurang dari **minNeighbor-1**, maka akan di tolak sebagai sebuah objek.
- 4) **HAAR_DETECTION_TYPE flag:** mode dari operasi. Untuk sekarang flag yang baru bisa digunakan hanya satu jenis yaitu **DO_CANNY_PRUNNING**. Jika **DO_CANNY_PRUNNING** yang digunakan maka fungsi akan menjalankan *canny edge detector*.
- 5) **Size minSize:** nilai minimum ukuran *image* yang akan terdeteksi.

```
foreach (var face in faces)
```


Digunakan untuk melakukan perulangan sebanyak jumlah isi dari *faces* agar menampilkan *faces* yang terdeteksi dengan sesuai.

```
ori.Draw(face.rect, new Bgr(Color.Lavender), 3);
```

Digunakan untuk menggambarkan *face* yang terdeteksi dalam bentuk *rectangle* pada *image ori*. Fungsi **Draw()** memiliki parameter sebagai berikut:

- 1) **Rectangle[] rect**: *face* yang terdeteksi yang ingin digambarkan dalam bentuk *rectangle*.
- 2) **TColor color**: warna untuk gambar *rectangle*.
- 3) **int thickness**: ketebalan *rectangle* yang akan digambarkan.

11. Tambahkan *event tick* pada **timer1** timer.



```
private void timer1_Tick(object sender, EventArgs e)
{
    if (isInProgress)
    {
        frame = cap.QueryFrame();

        if (frame != null)
        {
            if (isFaceDetection) faceDetection(frame);

            frameImage.Image = frame.ToBitmap();
        }
        else
        {
            isInProgress = false;
            timer1.Stop();
        }
    }
}
```

Penjelasan:

Event ini akan dijalankan ketika **timer1** di *start*.

```
frame = cap.QueryFrame();
```

Digunakan untuk menangkap *frame image* dengan tipe **Bgr** dan menampungnya ke **Image <Bgr, byte>**.

12. Tambahkan *event click* pada **cameraLoadBtn** button.

```
private void cameraLoadBtn_Click(object sender, EventArgs e)
{
    if (!isInProgress && cap == null)
    {
        timer1.Start();
        cap = new Capture(cameraDevice);
        isInProgress = true;
    }
    else
    {
        MessageBox.Show("Please stop video mode first!",
            "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}
```

Penjelasan:

Digunakan untuk me-load image dari webcam atau kamera (bersifat *real time*).

```
cap = new Capture(cameraDevice);
```

Digunakan untuk meng-create objek dari class **Capture** dengan menentukan bahwa *image* yang akan ditangkap berasal dari **webcam**. Webcam yang digunakan akan ditentukan berdasarkan indeks dan indeks yang berlaku dimulai dari 0. Untuk jawaban ini menggunakan indeks ke-0 yang telah ditentukan sebelumnya pada variabel **cameraDevice**.

Catatan: jika memiliki webcam internal dan eksternal, maka webcam internal yang akan didahulukan indeksnya.

13. Tambahkan *event click* pada **videoLoadBtn** button.

```
private void videoLoadBtn_Click(object sender, EventArgs e)
{
    if (isInProgress && cap != null)
    {
        MessageBox.Show("Please stop camera mode first!",
            "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error);

        return;
    }

    DialogResult result = openFileDialog1.ShowDialog();

    if (result == System.Windows.Forms.DialogResult.OK)
    {
        if (!isInProgress && cap == null)
        {
            timer1.Start();
            cap = new Capture(openFileDialog1.FileName);
            isInProgress = true;
        }
    }
}
```

Penjelasan:

Digunakan untuk me-load image dari file video.

```
cap = new Capture(openFileDialog1.FileName);
```

Digunakan untuk meng-create objek dari class **Capture** dengan menentukan bahwa *image* yang akan ditangkap berasal dari file **video**.

14. Tambahkan *event click* pada **exitBtn** button.

```
private void exitBtn_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Penjelasan:

Digunakan untuk **keluar** dari aplikasi ini.

15. Tambahkan *event click* pada **stopBtn** button.

```
private void stopBtn_Click(object sender, EventArgs e)
{
    timer1.Stop();
    isInProgress = false;
    isFaceDetection = false;
    cap = null;
    frameImage.Image=Chapter_5.Properties.Resources.no_image;
}
```

Penjelasan:

Digunakan untuk memberhentikan *frame image* yang ditangkap baik dari *webcam* maupun *file video*. Selain itu, *method* ini juga mengembalikan aplikasi pada kondisi awal.

16. Tambahkan *event click* pada **faceDetectBtn** button.

```
private void faceDetectBtn_Click(object sender, EventArgs e)
{
    isFaceDetection = true;
}
```



Penjelasan:

Digunakan untuk meng-*set* variabel **isFaceDetection** menjadi **true** yang artinya akan dilakukan *face detection* pada *frame image* yang tertangkap. Variabel **isFaceDetection** tersebut akan digunakan sebagai *flag* yang menentukan apakah *frame image* yang tertangkap akan langsung dilakukan *face detection* atau tidak pada *event tick timer1*.

17. Tambahkan *event click* pada **noFaceDetectBtn** button.

```
private void noFaceDetectBtn_Click(object sender, EventArgs e)
{
    isFaceDetection = false;
}
```

Penjelasan:

Digunakan untuk meng-*set* variabel **isFaceDetection** menjadi **false** yang artinya proses *face detection* pada *frame image* yang tertangkap akan langsung diberhentikan seketika. Variabel **isFaceDetection** tersebut akan digunakan sebagai *flag* yang

menentukan apakah *frame image* yang tertangkap akan langsung dilakukan *face detection* atau tidak pada event tick **timer1**.

Berikut ini adalah keseluruhan *code* dari *code* di atas:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;

namespace Chapter_5
{
    public partial class Form1 : Form
    {
        Capture cap = null;
        bool isInProgress = false,
            isFaceDetection=false;
        int cameraDevice = 0,
            fps = 30;
        Image<Bgr, Byte> frame;
        Image<Gray, Byte> edit;
        HaarCascade cascade;

        public Form1()
        {
            InitializeComponent();
            cascade = new HaarCascade("../\\..\\face_haarcascade.xml");
            timer1.Interval = 1000 / fps;
        }

        void faceDetection(Image<Bgr,byte> ori)
        {
            edit = new Image<Gray, byte>(ori.Size.Width,ori.Size.Height);
            CvInvoke.cvCvtColor(ori, edit, COLOR_CONVERSION.CV_BGR2GRAY);

            var faces = cascade.Detect(edit,1.8,2,
                HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,new Size(ori.Width,ori.Height));
```

```
        foreach (var face in faces)
        {
            ori.Draw(face.rect, new Bgr(Color.Lavender), 3);
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        if (isInProgress)
        {
            frame = cap.QueryFrame();

            if (frame != null)
            {
                if (isFaceDetection) faceDetection(frame);

                frameImage.Image = frame.ToBitmap();
            }
            else
            {
                isInProgress = false;
                timer1.Stop();
            }
        }
    }

    private void cameraLoadBtn_Click(object sender, EventArgs e)
    {
        if (!isInProgress && cap == null)
        {
            timer1.Start();
            cap = new Capture(cameraDevice);
            isInProgress = true;
        }
        else
        {
            MessageBox.Show("Please stop video mode first!",
                            "Error", MessageBoxButtons.OK,
                            MessageBoxIcon.Error);
        }
    }
}
```

```
private void videoLoadBtn_Click(object sender, EventArgs e)
{
    if (isInProgress && cap != null)
    {
        MessageBox.Show("Please stop camera mode first!",
            "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error);

        return;
    }

    DialogResult result = openFileDialog1.ShowDialog();

    if (result == System.Windows.Forms.DialogResult.OK)
    {
        if (!isInProgress && cap == null)
        {
            timer1.Start();
            cap = new Capture(openFileDialog1.FileName);
            isInProgress = true;
        }
    }
}

private void exitBtn_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void stopBtn_Click(object sender, EventArgs e)
{
    timer1.Stop();
    isInProgress = false;
    isFaceDetection = false;
    cap = null;
    frameImage.Image=Chapter_5.Properties.Resources.no_image;
}

private void faceDetectBtn_Click(object sender, EventArgs e)
{
    isFaceDetection = true;
}
```

```
private void noFaceDetectBtn_Click(object sender, EventArgs e)
{
    isFaceDetection = false;
}
}
```

References:

<http://docs.opencv.org/doc/tutorials/tutorials.html>

<http://www.emgu.com/wiki/files/2.4.2/document/Index.html>

<http://www.scs.ryerson.ca/~jmisic/theses/Shahnoor.pdf>

<http://bengkel.info/templates/bengkelinfo/images/icon-no-image.png>

<http://photo-bugs.com/wp-content/uploads/2011/12/Cute-Dog-Jacket.jpg>

<http://pichost.me/1302940/>

http://s3.amazonaws.com/viz-test/products/bleach_s19.jpg

<https://cdn0.iconfinder.com/data/icons/feather/96/no-512.png>

http://www.comm-tech.co.uk/wp-content/uploads/2013/05/person_icon_white.png

<https://cdn3.iconfinder.com/data/icons/linecons-free-vector-icons-pack/32/camera-512.png>

<http://thehottestwallpapers.net/wp-content/uploads/2014/06/cute-wallpaper-background-for-desktop-2.jpg>

<http://cainclusion.org/camap/images/resources/video.png>

<http://etc-mysitemyway.s3.amazonaws.com/icons/legacy-previews/icons/glossy-black-3d-buttons-icons-signs/091790-glossy-black-3d-button-icon-signs-z-roadsign20.png>

<https://cdn3.iconfinder.com/data/icons/unicons-vector-icons-pack/32/exit-512.png>

<http://www.youtube.com/watch?v=gqdn2T197Sc>