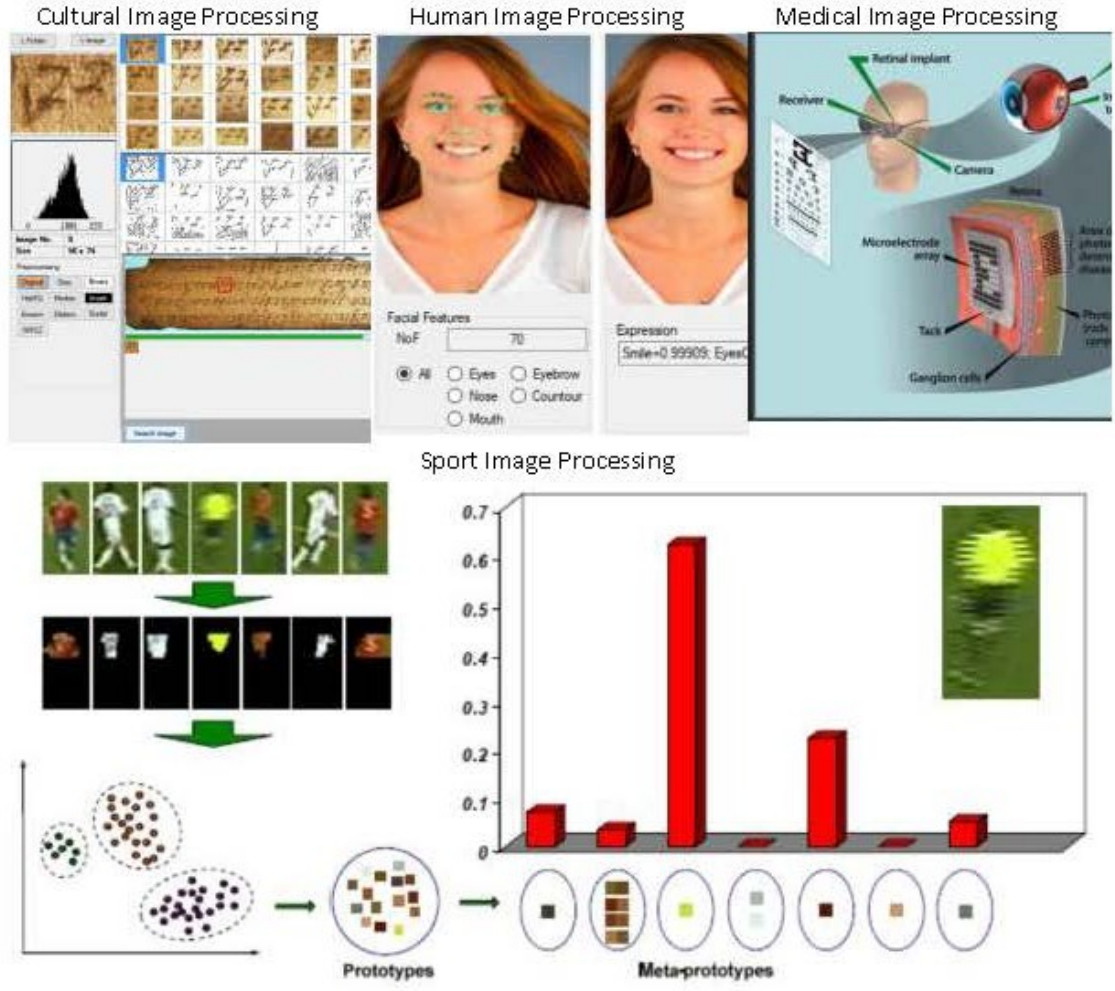
# MODUL PRAKTIKUM
# PENGOLAHAN DAN ANALISIS CITRA 2019

Oleh:
Dr. Setiawan Hadi, M.Sc.CS.




PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN
FEBRUARI 2019

# Kata Pengantar

Mulai Semester Genap 2018-2019, mata kuliah yang berkaitan dengan Pengolahan Citra dan Visi Komputer diintegrasikan dan namanya dirubah menjadi Pengolahan dan Analisis Citra Digital. hal ini dilakukan dengan merujuk kepada perkembangan teknologi Citra Digital yang semakin berkembang dengan munculnya aplikasi-aplikasi cerdas yang relevan dengan pemahaman citra (*image understanding*). Dengagn demikian diharapkan mahasiswa tidak hanya memahami proses pengolahan citra saja (*image-to-image*), namun juga memahami proses yang lebih lanjut yang berkaitan dan analisis dan pemahaman citra (*image-to-text*).

Praktikum ini disusun dalam 2 bagian besar yaitu Bagian Pertama yaitu Pengolahan Citra dan Bagian Kedua yaitu Analisis Citra. Buku ini menjadi pegangan dan bahan praktikum bagi peserta mata kuliah Pengolahan dan Analisis Citra Digital di laboratorium. Materi yang disajikan diusahakan sejalan dengan materi teori yang diberikan di dalam kelas. Dengan demikian penjelasan yang diberikan pada modul praktikum ini bersifat sebagai material pendukung yang akan meningkatkan pemahaman secara teoritis melalui praktek nyata di laboratorium.

Pertama-tama para mahasiswa akan diperkenalkan dengan teknik pemrograman citra digital menggunakan bahasa pemrograman C#. Selanjutnya bagian utama pertama yaitu Pengolahan Citra dikelompokkan menjadi lima bagian yaitu Pengolahan Citra Berbasis Titik, Pengolahan Citra Berbasis Daerah/Area/Region, Pengolahan Citra Berbasis Geometri, dan Pengolahan Citra Berbasis *Frame*. Beberapa pengetahuan pemrograman lanjut misalnya yang berkaitan dengan pengaksesan piksel via *pointer*, pengaksesan *video*/*webcam* serta penggunaan *software library* yang mendukung dan relevan, diberikan pada lampiran. Untuk tahun 2019 ini Bagian Kedua yaitu Analisis Citra masih belum dapat dilaksanakan praktikum, tetapi akan diajarkan di dalam kelas.

Setiap praktikum bisa dilaksanakan lebih dari satu pertemuan tergantung kepada luas dan dalamnya materi praktikum yang harus dilakukan. Struktur setiap praktikum diawali dengan penjelasan awal berisi deskripsi dan tujuan (target) yang menjadi indikator keberhasilan praktikum. Kemudian diberian secara singkat teori dan latihan yang kemudian diikuti dengan tugas praktikum yang berisi pertanyaan teori dan tugas pemrograman. Mudah-mudahan praktikum ini dapat membekali mahasiswa dengan kemampuan kepada para mahasiswa untuk memahami teknik pemrograman grafis/citra sekaligus menganalisis citra sehingga diharapkan muncul inspirasi dan inovasi untuk membangun dan mengembangkan aplikasi-aplikasi cerdas yang relevan dan dapat membantu pekerjaan manusia.

Jatinangor, Februari 2019
SH.

# Daftar Isi

Praktikum 1

# Pemrograman Citra Dengan C#

Pada praktikum ini peserta akan mempraktekkan pembuatan program komputer dengan bahasa C# yang berkaitan dengan masalah *input-output* citra dijital dan *property*nya. Setelah melaksanakan praktikum ini peserta akan :

1. mengetahui elemen-elemen inti dalam *Integrated Development Environment* (IDE) Visual Studio khusus untuk bahasa pemrograman C#.
2. memahami cara membuat program untuk membaca (*Load*) dan menampilkan gambar dijital (*Image*)
3. memahami elemen `PictureBox` secara mendalam termasuk pemanfaatannya dalam pemrograman
4. mampu membuat *user interface* untuk sebuah program aplikasi yang interaktif dan ramah (*user friendly*) khususnya untuk proses pengolahan citra dijital

## 1.1 Teori dan Latihan

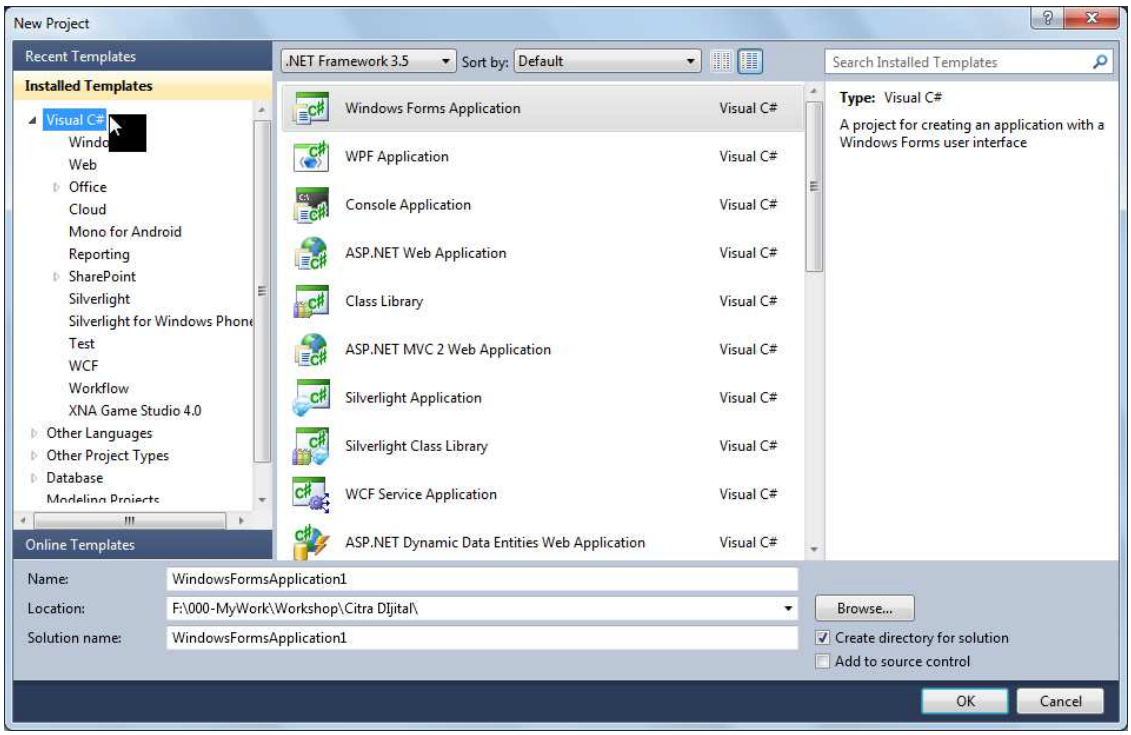
Pemrograman citra pada dasarnya mengacu kepada pemrograman biasa, hanya fokus pemrograman diarahkan pada pemrosesan citra dijital. Pemrograman citra bisa mempergunakan berbagai bahasa pemrograman mulai dari C, C++ Delphi, Phyton dan C#. Pada praktikum ini digunakan bahasa pemrograman C# dengan lingkungan pemrograman Visual Studio .NET 2012.

Citra dijital menjadi objek input yang kemudian diproses sedemikian rupa untuk mendapatkan informasi yang terkandung di dalamnya. Terdapat dua jenis informasi yang dapat diambil dari sebuah citra dijital. Pertama adalah informasi sintaksis dan kedua adalah informasi semantik.

Informasi sintaksis berkaitan dengan karakteristik fisik dari citra dijital antara lain resolusi, kuantisasi, nilai intensitas, tingkat *sampling* dan sebagainya. Informasi semantik berkaitan dengan eksplorasi kandungan makna citra dijital yang informasi dikelompokan ke dalam informasi tekstual, figural, fasial, ekspesional, aksional dan situasional/kondisional. Proses ini disebut juga analisis citra dijital yang kelompok ilmunya disebut dengan visi komputer (*computer vision*).

Visual Studio merupakan produk Microsoft untuk pengembangan aplikasi dalam lingkungan pemrograman yang terintegrasi (IDE). Salah satu bahasa yang didukung dalam Visual Studio adalah C#. Pada saat anda menginstalasi Visual Studio.NET maka ada opsi untuk

menginstal semua bahasa yang didukung atau hanya bahasa yang diperlukan saja. Contoh tampilan awal Visual Studio.NET setelah memilih proyek baru adalah sebagai berikut:




Gambar 1.1 Tampilan VS.NET Setelah Membuat Proyek Baru

Langkah selanjutnya yang dilakukan adalah:

1. Pilih `Windows Form Application Visual C#`
2. Isi `Name` dengan nama proyek yang akan dikerjakan
3. Tentukan lokasi foldernya pada isian `Solution Name`
4. Click OK

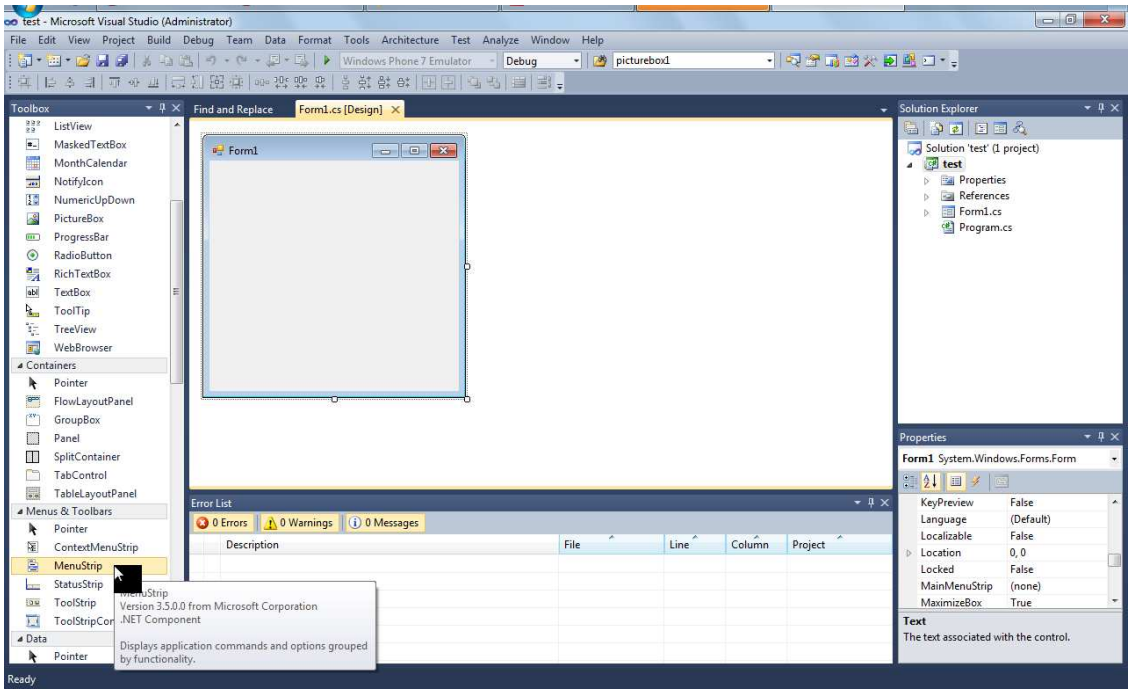
Layar berikutnya yang muncul adalah lembar kerja pemrograman seperti digambarkan pada gambar 1.2.

### 1.1.1 Elemen `PictureBox`

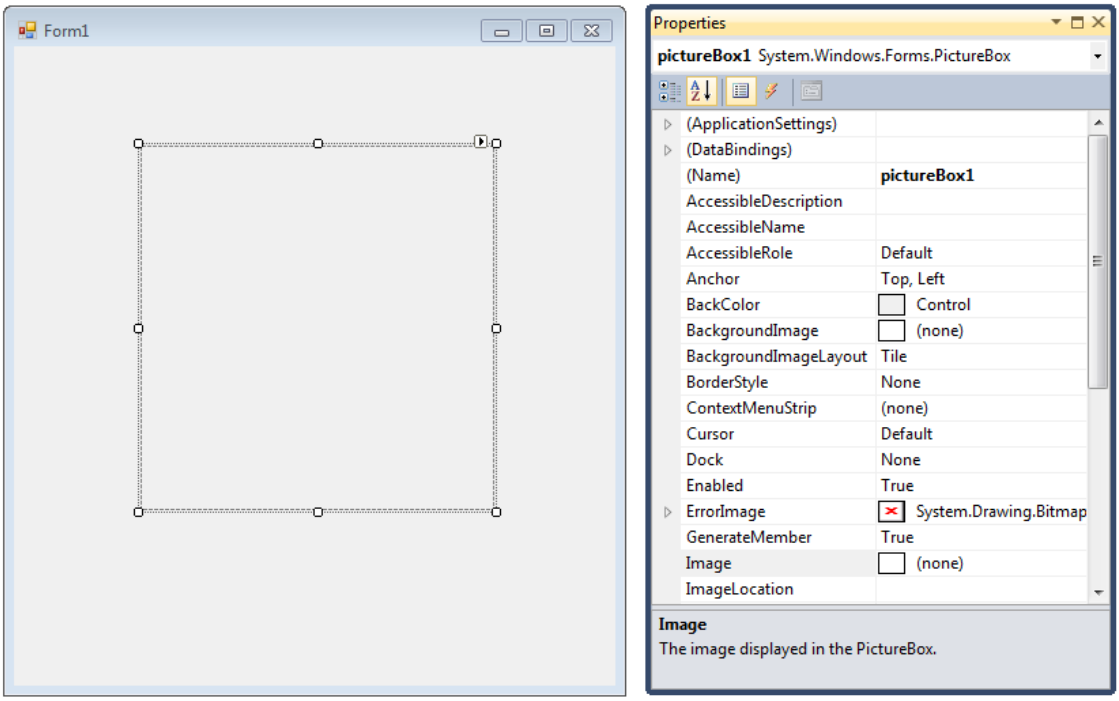
Elemen adalah bagian dari *Toolbox* yang ada dalam Visual Studio dan dapat terdiri dari *Common Controls*, *Containers*, *Menu & Toolbars*, *Data*, *Components*, *Printing*, *Dialogs* dan sebagainya. Seluruh elemen dalam *Toolbox* dapat ditampilkan dengan memilih menu *View* lalu *Toolbox*, atau dengan shortcut `Ctrl+W,X`. Elemen menjadi bagian penting dalam pemrograman visual. Tampilan elemen `picturebox` dan *property*nya dapat dilihat pada Gambar 1.3.

Berikut ini adalah latihan untuk dikerjakan. Sebagai ilustrasi lihat Gambar 1.4.

**Latihan 101**




Gambar 1.2 Lembar Kerja Pemrograman pada VS.NET

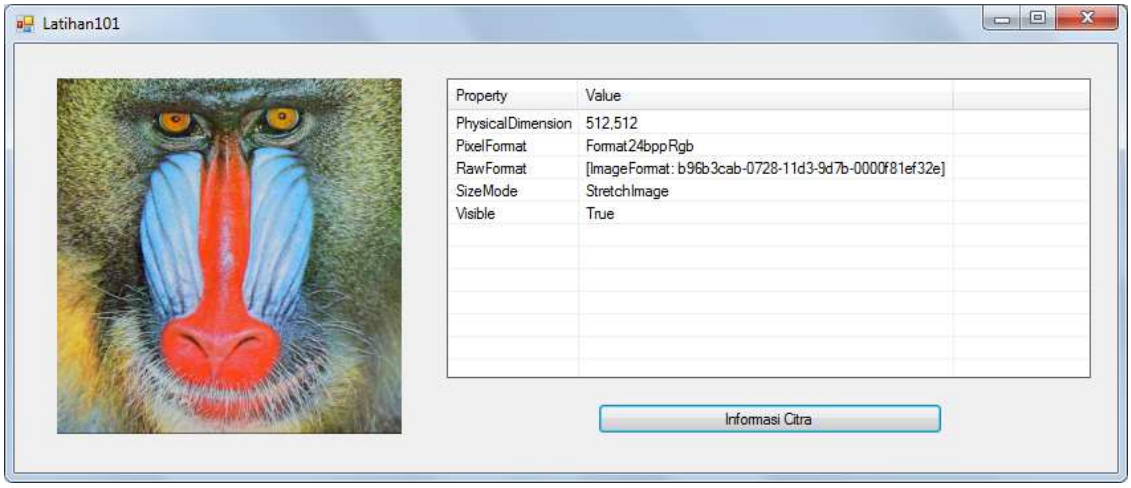



Gambar 1.3 Elemen `PictureBox` dan *`Property`*`nya`

1. Jalankan Visual Studio, lalu buatlah proyek baru dengan nama `Latihan101` .
2. Pilih dan *drag* elemen `PictureBox` ke area Form
3. Pilih dan isi elemen `Image` pada property `PictureBox` dengan gambar yang anda pilih sendiri

4. Jalankan (*compile*) program anda dengan mengklik tombol *toolbar* ▶ atau fungsi `F5`

5. Eksplorasi semua *property* di `PictureBox` dan cobalah dengan program komputer untuk mengetahui semua fungsinya




Gambar 1.4 Contoh Program **Latihan101**

### 1.1.2 Memuat dan Menampilkan Gambar

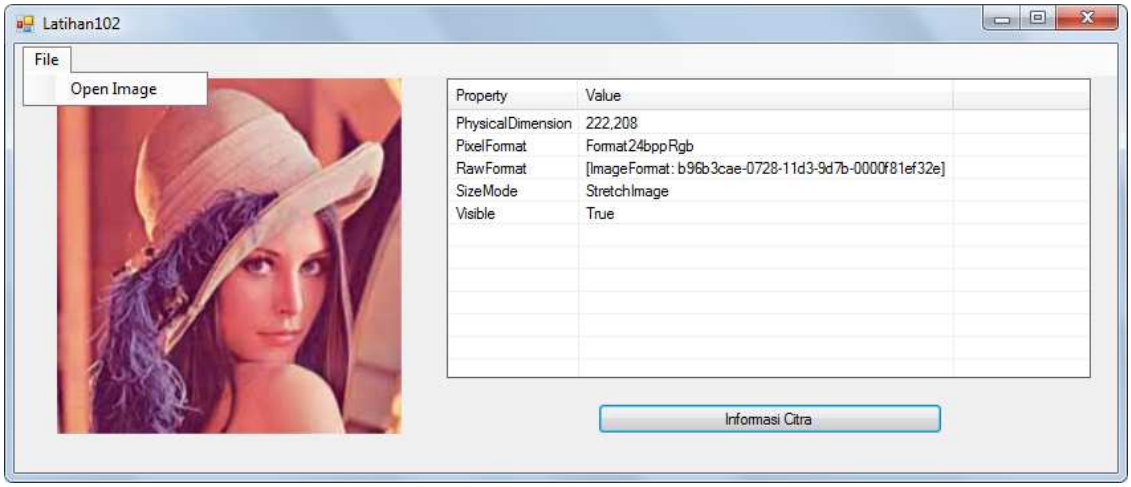
Agar program dapat bekerja secara fleksibel maka gambar yang diinginkan harus bisa diambil (*loading*) dari tempat penyimpanan. Selanjutnya gambar ditampilkan dan diproses seperti biasa. Dalam C#, untuk mengambil gambar dapat dilakukan dengan berbagai cara. Hal yang umum dilakukan adalah menggunakan elemen `OpenFileDialog` dengan cara seperti yang ditunjukkan pada potongan program berikut:

```
1 OpenFileDialog open = new OpenFileDialog();
2 open.Filter = "Image Files(*.jpg; *.bmp)|*.jpg; *.bmp";
3 if (open.ShowDialog()==DialogResult.OK)
4 {
5     pictureBox1.Image = new Bitmap(open.FileName);
6 }
```

Berikut ini adalah latihan untuk dikerjakan. Sebagai ilustrasi lihat Gambar 1.5.

**Latihan 102**

1. Copykan folder `latihan101` ke `latihan102`
2. Jalankan Visual Studio, lalu ganti semua elemen `latihan101` ke `Latihan102`.
3. Buat menu menggunakan elemen `MenuStrip`, gunakan penamaan yang standar yaitu `File` $->$ `Open Image`
4. Isikan *action* `Open Image` dengan potongan kode di atas
5. Jalankan (*compile*) program anda




Gambar 1.5 Contoh Tampilan Menggunakan Menu (**Latihan 102**)

## 1.2 Tugas Praktikum

1. Sebutkan elemen *Toolbox* dalam pemrograman citra dengan C# yang berfungsi untuk menampung gambar?
2. Apa kegunaan elemen `OpenFileDialog` dalam konteks pemrosesan citra ?
3. Sebutkan nilai-nilai dari property `SizeMode` dan jelaskan !
4. Jelaskan perbedaan antara file citra berformat `jpg` dan berformat `bmp`!
5. Buatlah program **Tugas101** yang lengkap, interaktif, ramah dan menarik untuk mengambil dan menampilkan gambar dijital serta memberikan informasi sintaksis tentang citra dijital tersebut.

   Hint: Gunakan elemen-elemen *Toolbox* yang relevan untuk mendukung fungsionalisasi program. Kembangkan kreativitas dan sifat eksplotratif. Berani mencoba dan siap menghadapi kesalahan. Jangan malu bertanya dan mencari materi pengetahuan pendukung dari berbagai sumber.

Praktikum 2

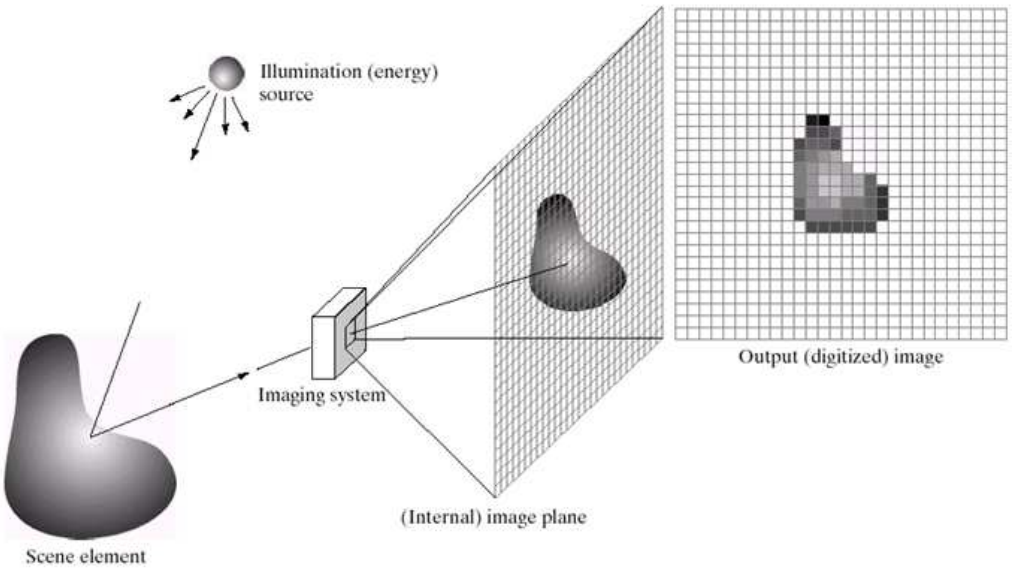
# Pemrosesan Citra Berbasis Titik

Pada praktikum ini peserta akan mempraktekkan algoritma pemrosesan citra didasarkan pada operasi antara citra dan nilai konstanta tunggal. Setelah melaksanakan praktikum ini peserta akan :

1. Memahami konsep dasar citra dijital serta representasi dan pengolahannya baik secara matematis maupun praktis pada komputer
2. Mengetahui cara membaca nilai intensitas *bitmap* dengan fungsi/method standar `GetPixel()`
3. Membuat program untuk mengakses nilai intensitas piksel citra dijital
4. Mengetahui cara menulis nilai pada sebuah *bitmap* dengan fungsi standar `SetPixel()`
5. Membuat program untuk mengimplementasi algoritma-algoritma dalam pengolahan citra berbasis titik sebagai berikut
   a. Algoritma pengatur kecerahan (*brightness*)
   b. Algoritma inversi citra
   c. Algoritma transformasi citra warna ke citra keabuan (*graylevel*)
   d. Algoritma *thresholding* sederhana
   e. Algoritma ekualisasi histogram
6. Memahami konsep pengaksesan piksel menggunakan *pointer* dan mengimplementasikannya pada pemrograman pengolahan citra berbasis titik untuk algoritma-algoritma yang telah dibuat sebelumnya

## 2.1 Teori dan Latihan

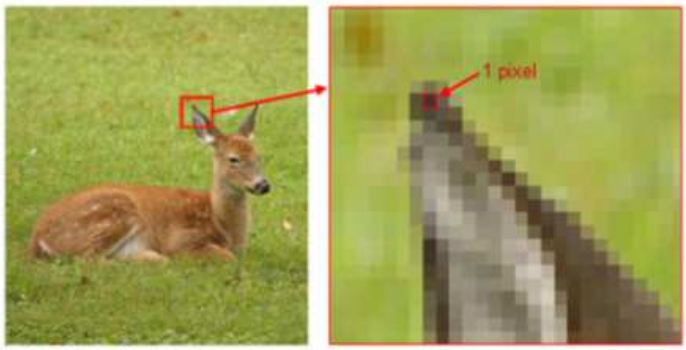
Citra dijital dapat didefinisikan sebagai kumpulan titik-titik diskrit dalam bidang dua dimensi (*array*), dimana setiap titik memiliki nilai kecerahan atau nilai intensitas. Titik-titik tersebut dinamakan *picture elements* dan dikenal dengan sebutan




Gambar 2.1 Konsep Citra Dijital

*pixel* atau piksel dalam Bahasa Indonesia. Gambar 2.1 mengilustrasikan hal tersebut.

Nilai intensitas piksel berupa bilangan bulat pada interval tertentu bergantung pada tingkat kuantisasinya. Proses ptransformasi dari gambar analog ke gambar dijital disebut dengan dijitisasi dimana energi warna pada gambar analog dikuantifikasikan dengan nilai berupa angka pada interval tertentu. Gambar 2.2 mengilustrasikan piksel dalam sebuah citra dijital. Tabel yang berisi angka-angka menun-




| | | | | |
|---|---|---|---|---|
| 166,172,170 | 154,150,187 | 127,110,80 | 143,128,42 | 171,180,98 |
| 165,170,73 | 85,81,46 | 59,52,39 | 160,155,108 | 170,184,89 |
| 170,175,85 | 88,89,47 | 107,93,82 | 158,155,110 | 170,185,90 |
| 169,175,69 | 88,89,47 | 75,72,53 | 106,92,55 | 169,186,82 |
| 185,189,103 | 143,146,92 | 102,99,69 | 84,77,61 | 84,68,45 |

Gambar 2.2 Citra Dijital dan Nilai Piksel

jukkan nilai intensitas pada region yang berpusat pada piksel dengan ukuran region

$5 \times 5$. Elemen tabel terdiri dari tiga nilai yang menunjukkan gabungan antara nilai Merah (*Red*), Hijau (*Green*) dan Biru (*Blue*) atau dikenal dengan nilai RGB. Fokus proses pengolahan citra diarahkan pada proses manipulasi nilai-nilai angka tersebut untuk menghasilkan nilai baru yang tampilan visualnya sesuai dengan yang diinginkan.

Secara umum jenis citra berdasarkan kompoisisi pikselnya dapat dikelompokkan menjadi tiga jaitu citra warna, citra graylevel dan citra biner. Citra warna dibangun menggunakan tiga *channel* warna yaitu *Red*, *Green* dan *Blue* dan secara umum tiap *channel* warna memiliki nilai interval sebesar 8 bit yaitu antara 0 sampai dengan 255 (256 nilai intensitas). Citra graylevel dapat dianggap sebagai citra dengan satu *channel* warna dengan interval yang sama yaitu 0 sampai dengan 255. Sedangkan citra biner adalah citra dengan dua nilai intensitas, misalnya 0 dan 255. Tabel gambar 2.1 dan 2.1 menggambarkan visualisasi jenis citra dijital dan citra pada msing-masing *channel* warna

Tabel 2.1 Jenis-jenis Citra Dijital

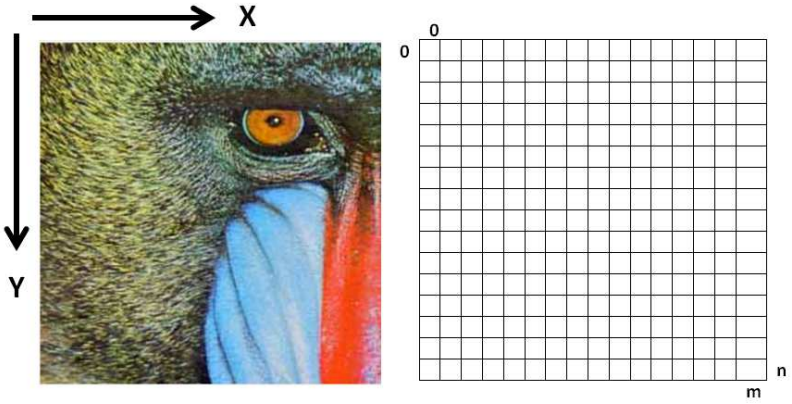
| Citra R G B | Citra Gray | Citra Biner |
|---|---|---|
|  |  |  |
| $I_R(x,y)$ $I_G(x,y)$ $I_B(x,y)$ | $I(x,y) \in [0 \ldots 255]$ | $I(x,y) \in [0,1]$ |

Tabel 2.2 *Channel* Warna dalam Citra Dijital

| Channel Red | Channel Green | Channel Blue |
|---|---|---|
|  |  |  |

### 2.1.1 Pengaksesan Piksel Menggunakan Fungsi `GetPixel()`

Nilai-nilai RGB dari sebuah citra dijital dapat diakses dengan pemrograman C# menggunakan fungsi `GetPixel()`. Hanya perlu diperhatikan bahwa koordinat origin citra dijital $(0,0)$ adalah pada elemen kiri atas dari citra dijital. Secara umum proses pengaksesan piksel dilakukan ke arah kanan dan apabila sudah sampai dibatas kanan, proses pengaksesan dilanjutkan ke baris berikutnya diawali dari kolom kiri. Ilustrasinya ditunjukkan pada Gambar 2.3.




Gambar 2.3 Pengaksesan Piksel Pada Citra Dijital

Potongan program di bawah ini memberikan gambaran bagaimana mengakses piksel menggunakan fungsi `GetPixel()`.

```
1         Bitmap bmp;
2         private void buttonProses_Click(object sender, EventArgs e)
3         {
4               int x = Convert.ToInt16(textBoxX.Text);
5               int y = Convert.ToInt16(textBoxY.Text);
6               bmp = (Bitmap)pictureBox1.Image;
7               int r = bmp.GetPixel(x, y).R;
8               int g = bmp.GetPixel(x, y).G;
9               int b = bmp.GetPixel(x, y).B;
10              textBoxR.Text = r.ToString();
11              textBoxG.Text = g.ToString();
12              textBoxB.Text = b.ToString();
13          }
```

**Baris 1:** Definisi variabel `bmp` bertipe `Bitmap` secara global

**Baris 2:** Definisi fungsi elemen `Button` yang aktif pada saat diklik. Nama variabelnya `buttonProses`

**Baris 4,5:** deklarasi variabel integer `x` dan `y` yang diisi nilai dari variabel `textBoxX` dan `textBoxY`

**Baris 6:** Variabel `bmp` diisi elemen `pictureBox1` yang dikonversikan ke tipe `Bitmap`

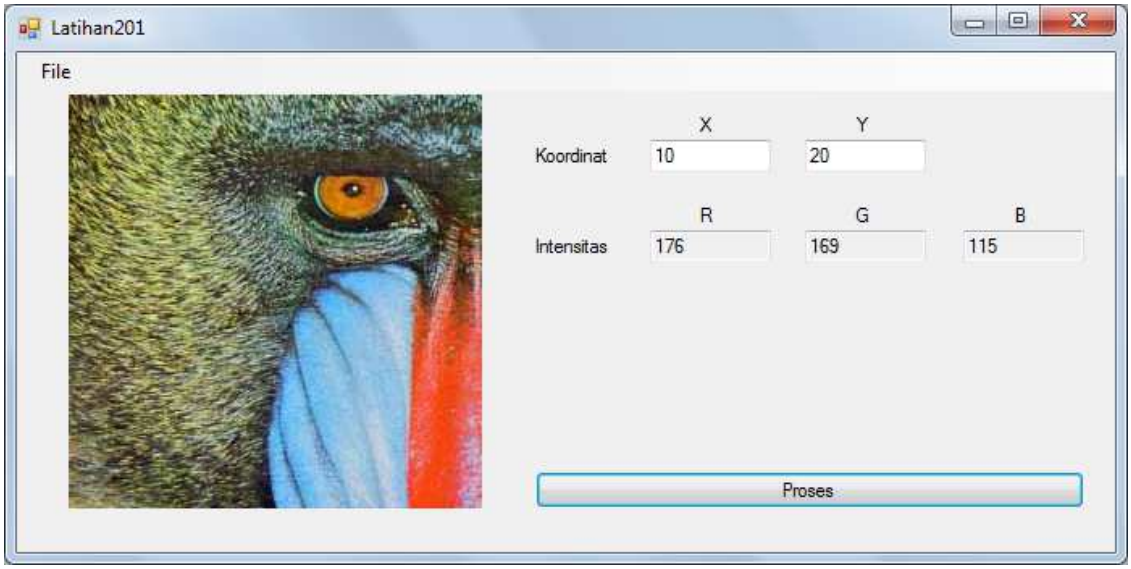
**Baris 7,8,9:** Mengambil nilai intensitas `R`, `G`, `B` dengan fungsi `GetPixel` pada koordinat `x` dan `y` dan dimasukkan ke variabel integer `r`, `g` dan `b`

**Baris 10,11,12:** Memasukkan nilai `r`, `g` dan `b` dan menampilkannya dalam `textBoxR`, `textBoxG`, dan `textBoxB`

Berikut ini adalah Latihan 201 untuk dikerjakan. Sebagai ilustrasi lihat Gambar 2.4.

**Latihan 201**

1. Kopikan folder `latihan102` menjadi `latihan201`, baca `latihan102` pada folder `latihan201`, ubah semua *identifier* `latihan102` menjadi `latihan201`.
2. Set *property* `SizeMode` `pictureBox1` ke `Normal`
3. Buat variabel koordinat nilai piksel, beri nama `textBoxX` dan `textBoxY`
4. Buat variabel nilai intensitas piksel, beri nama `textBoxR`, `textBoxG` dan `textBoxB`. Set properti `ReadOnly` ketiganya menjadi bernilai `True`
5. Buat/ubah `Button` dan beri nama variabelnya sebagai `buttonProses` dan isinya tulisan `Proses`
6. Isikan fungsi `buttonProses_Click` dengan baris seperti pada *listing* di atas.
7. Jalankan (*compile*) program anda




Gambar 2.4 Mengakses Nilai Intensitas Piksel (**Latihan201**)

Untuk mengetes program anda, muatlah sebuah citra. Lalu isikan koordinat `X` dan `Y` dengan nilai integer tertentu sesuai dengan ukuran elemen `pictureBox`. Hati-hati jika ukuran citra lebih besar maka gambar akan terpotong. Jika diklik *button* **Proses** maka akan ditampilkan nilai intensitas *Red*, *Green* dan *Blue* pada lokasi piksel $X, Y$.

Untuk membuat program anda lebih interaktif dan *user-friendly*, anda dapat menggunakan *Event Property* `MouseEnter`, `MouseLeave` dan `MouseMove` untuk memeriksa kejadian apakah *pointer mouse* memasuki wilayah `pictureBox`. Jika memasuki wilayah `pictureBox` maka posisi piksel dapat diketahui melalui argumen yang ada dalam even tersebut sehingga nilai intensitas pada posisi tersebut dapat langsung ditampilkan.

### 2.1.2 Mengakses Piksel Citra Dijital Secara Global

Sebagaimana dijelaskan sebelumnya, pada dasarnya representasi internal citra dijital semata-mata berupa kumpulan angka-angka dalam tiga kelompok (*layer*, *channel*) yang secara matematis dapat disebut sebagai matriks citra dua dimensi. Dengan demikian cara pengaksesan secara pemrograman menggunakan fasilitas *looping* yaitu `for` yang bergerak sepanjang baris dan kolom. Algoritma berikut mengilustrasikan pengaksesan nilai intensitas piksel pada citra dijital.

```
1        for (i=0;i<baris;i++)
2          for (j = 0; j < kolom; j++)
3          {
4              nilaiR = bmp.GetPixel(i, j).R;
5              nilaiG = bmp.GetPixel(i, j).G;
6              nilaiB = bmp.GetPixel(i, j).B;
7          }
```

Jika kita perhatikan potongan program di atas merupakan proses perulangan biasa yang melibatkan dua buah *statement* `for` yang mengakses posisi nilai intensitas $(i, j)$ mulai dari 0 hingga $baris - 1$ dan $kolom - 1$. Nilai baris dan kolom adalah banyaknya piksel dalam baris dan banyaknya piksel dalam kolom atau dikenal dengan apa yang disebut **resolusi**. `nilaiR, nilaiG dan nilaiB` merupakan variabel penampung nilai intensitas piksel $R$, $G$, $B$ pada piosisi $(i, j)$. Pada dasarnya proses pengolahan citra adalah proses mengubah nilai-nilai intensitas R,G,B ini sesuai dengan keperluan.

### 2.1.3 Menggunakan fungsi `SetPixel()`

Proses operasi citra dijital dilakukan dengan mengubah nilai intensitas pada posisi tertentu. Namun perubahan yang dilakukan tidak serta merta langsung mengubah tampilan pada layar. Untuk itu diperlukan sebuah fungsi lain untuk menampilkan perubahan nilai intensitas citra, yaitu fungsi `SetPixel()`. Fungsi ini akan mengambil nilai intensitas baru kemudian menuliskannya pada elemen `Bitmap` pada `PictureBox`.

Berikut ini disajikan **Latihan 202** yang menunjukkan proses peningkatan kecerahan citra dengan cara menambahkan nilai sebesar $k$ untuk setiap nilai intensitas citra dengan formulasi umum $F'(x, y) = F(x, y) + k$. Nilai $k$ berupa bilangan bulat sembarang. Perlu diperhatikan bahwa proses penambahan tidak boleh melebihi nilai maksimum intensitas secara umum yaitu 255 atau 0 (untuk pengurangan). Hasilnya ditampilkan pada elemen `pictureBox` yang lain seperti diilustrasikan pada Gambar 2.5.

**Latihan 202**

1. Kopikan folder `latihan201` menjadi `latihan202`, baca `latihan201` pada folder `latihan202`, ubah semua *identifier* `latihan201` menjadi `latihan202`.

2. Buat elemen `pictureBox2` dan `textBox1`.

3. Berikan nama `pictureBox1` sebagai `pictureBoxAsli`, `pictureBox2` sebagai

pictureBoxHasil, textBox1 dengan nama textBoxBrightness, button1 dengan nama buttonProses.

4. Jadi apabila dimasukkan nilai pada textBoxBrightness kemudian ditekan buttonProses, maka setiap piksel yang ada pada pictureBoxAsli akan ditambahkan dengan nilai pada textBoxBrightness dan hasilnya ditampilkan pada pictureBoxHasil. Tentunya gambar harus dimuat dulu sesuai dengan prosedur latihan sebelumnya.

5. Lihat listing program berikut dan cobalah

```
1         Bitmap bmpAsli, bmpHasil;
2         private void buttonProses_Click(object sender, EventArgs e)
3         {
4             int k = Convert.ToInt16(textBoxBrightness.Text);
5             int i, j;
6             int nilaiR, nilaiG, nilaiB;
7             bmpAsli = (Bitmap)pictureBoxAsli.Image;
8             int baris=bmpAsli.Width;
9             int kolom=bmpAsli.Height;
10            bmpHasil =new Bitmap(baris, kolom);
11            Cursor = Cursors.WaitCursor;
12            for (i=0;i<baris;i++)
13                for (j = 0; j < kolom; j++)
14                {
15                    nilaiR = bmpAsli.GetPixel(i, j).R + k;
16                    nilaiG = bmpAsli.GetPixel(i, j).G + k;
17                    nilaiB = bmpAsli.GetPixel(i, j).B + k;
18                    if (nilaiR > 255) nilaiR = 255;
19                    if (nilaiG > 255) nilaiG = 255;
20                    if (nilaiB > 255) nilaiB = 255;
21                    if (nilaiR < 0) nilaiR = 0;
22                    if (nilaiG < 0) nilaiG = 0;
23                    if (nilaiB < 0) nilaiB = 0;
24                    bmpHasil.SetPixel(i, j, Color.FromArgb(nilaiR,
                          nilaiG, nilaiB));
25                }
26            pictureBoxHasil.Image = bmpHasil;
27            Cursor = Cursors.Default;
28        }
```

**Baris 1:** Deklarasi variabel bmpAsli dan bertipe bmpHasil Bitmap secara global. Variabel ini berkaitan dengan elemen pictureBoxAsli dan pictureBoxHasil

**Baris 2:** Method untuk buttonProses

**Baris 4:** Deklarasi variabel *integer* k, inisiasi oleh nilai konversi *text* ke *integer* dari textBoxBrightness

**Baris 5:** Deklarasi variabel *integer* i dan j yang akan digunakan sebagai indeks ke matris intensitas citra

**Baris 6:** Deklarasi variabel *integer* nilaiR, nilaiG, nilaiB untuk menampung nilai baru hasil perubahan *brightness*

**Baris 7:** Pengambilan nilai intensitas gambar dan disimpan dalam matriks intensitas `bmpAsli`

**Baris 8,9:** Deklarasi variabel *integer* `baris` dan `kolom` yang berisi ukuran citra

**Baris 10:** Deklarasi variabel baru `bmpHasil` bertipe `Bitmap` yang berukuran `baris` dan `kolom`

**Baris 11,27:** Fungsi pengubah *cursor*

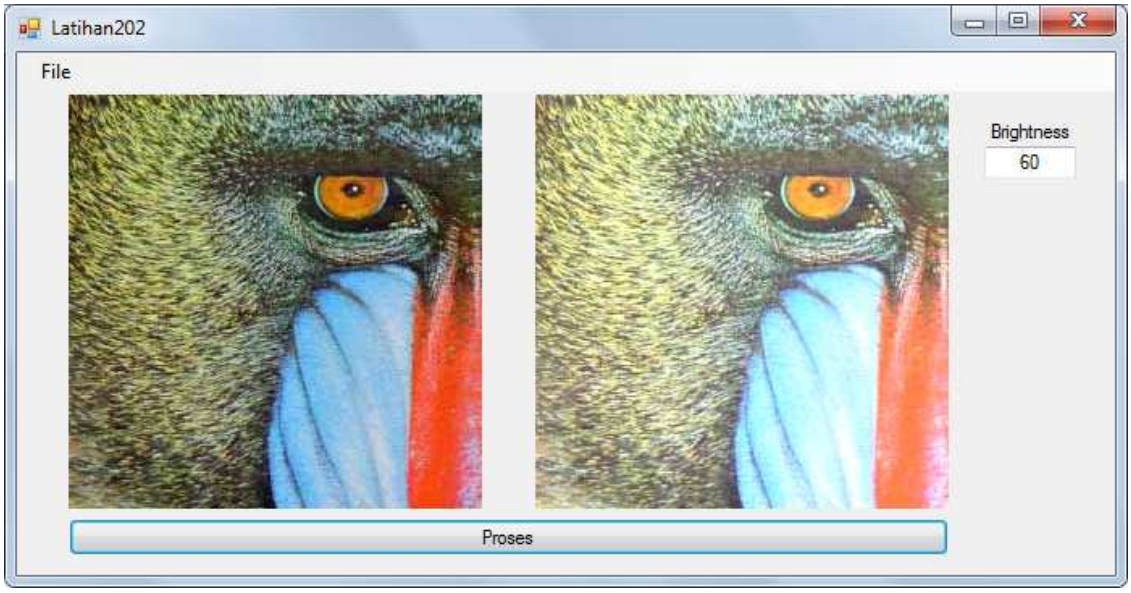
**Baris 12,13:** Perintah iterasi sepanjang kolom dan baris

**Baris 15-17:** Nilai intensitas dari citra asli pada posisi `(i,j)` ditambahkan dengan nilai `k`, hasilnya disimpan masing-masing di variabel `nilaiR`, `nilaiG` dan `nilaiB`.

**Baris 18-23** Proses *clamping* untuk mencegah *overflow* dan *underflow*

**Baris 24:** Proses memindahkan `nilaiR`, `nilaiG` dan `nilaiB` ke posisi `(i,j)` pada `bmpHasil`

**Baris 26:** `pictureBoxHasil` diisi dengan `bmpHasil` dan ditampilkan di layar




Gambar 2.5 Tampilan Program Latihan 202

*Catatan*: Kinerja proses menggunakan fungsi `SetPixel` dan `GetPixel` walaupun fleksibel dan mudah dimengerti namun kinerjanya kurang optimal. Untuk mengoptimalkan proses dan pengaksesan piksel dalam citra biasanya digunakan teknik yang lebih canggih dengan kinerja yang tinggi yaitu *pointer* yang contohnya dapat dilihat di lampiran.

## 2.2 Tugas Praktikum

1. Jelaskan fungsi `GetPixel()` dan `SetPixel` dalam pemrograman C#
2. Sebutkan proses untuk mencegah terjadinya *overflow* dan *underflow*, lalu jelaskan
3. Buatlah program terintegrasi **Tugas201** untuk melakukan fungsi-fungsi sebagai berikut:
   a. Operasi Brightness
   $$f'(i,j) = f(i,j) + k$$
   b. Operasi Inversi Citra
   $$f'(i,j) = 255 - f(i,j)$$
   c. Operasi Konversi Citra Warna ke Citra *grey* Menggunakan teknik perataan
   $$f'(i,j) = \frac{f_R(i,j) + f_G(i,j) + f_B(i,j)}{3}$$
   atau menggunakan teknik pembobotan
   $$f'(i,j) = 0.299 f_R(i,j) + 0.587 f_G(i,j) + 0.114 f_B(i,j)$$
   d. Operasi *Thresholding* Tunggal (Konversi ke Citra Biner)
   $$f'(i,j) = \begin{cases} a_1 & if \quad f(i,j) > T \\ a_2 & if \quad f(i,j) <= T \end{cases}$$
   *Untuk citra biner gunakan* $a_1 = 255$ *dan* $a_2 = 0$
   e. Operasi Ekualisasi Histogram (Eksplorasi via internet dan *attach* sebagai **Tugas201**)
4. Perbaiki kinerja program **Tugas201** dengan memanfaatkan pointer sebagai *tool* untuk mengakses piksel pada citra dijital

Praktikum 3

# Pemrosesan Citra Berbasis Area

Pada praktikum ini peserta akan mempraktekkan algoritma pemrosesan citra didasarkan pada operasi antara citra dan matriks berukuran kecil yang disebut dengan operator. Nama lain matriks ini adalah *kernel* atau *mask* atau *subwindow* atau *filter*. Setelah melaksanakan praktikum ini peserta akan :

1. Memahami apa yang disebut dengan *pixel neigborhood*
2. Memahami konsep dan perhitungan konvolusi citra dan penanggulangan *boundary problems*
3. Membuat program untuk mengimplementasi algoritma-algoritma dalam pengolahan citra berbasis area sebagai berikut
   a. Algoritma *Filtering* Dengan *Mask*
      i. Algoritma *Lowpass Filter* (*blurring*)
      ii. Algoritma *High Filter* (*sharpening*)
      iii. Algoritma *Embossing*
      iv. Algoritma Pendeteksi Tepi Citra (*edge detection*) dengan teknik Sobel, Prewitt, Laplacian dan sebagainya
   b. Algoritma *Filtering* Dengan Tanpa *Mask*
      i. Algoritma *Median Filtering*
      ii. Algoritma *Maximum/Minimum Filtering*
4. Algoritma *Morphological Filtering*

## 3.1 Teori dan Latihan

Pemrosesan citra berbasis area pada dasarnya mirip dengan pemrosesan citra berbasis piksel, namun dalam proses komputasinya, nilai-nilai piksel disekitar piksel utama (*pixel neighborhood*) diperhatikan dan menjadi elemen penentu piksel yang akan dihitung hasilnya.

Tingkat kompleksitas proses citra ini cukup lumayan mengingat skala pemrosesan tidak tunggal, namun skalanya bervariasi tergantung pada ukuran operator

yang digunakan. Umumnya ukuran operator bernilai ganjil, mulai dari $3 \times 3, 5 \times 5$ dan seterusnya. Semakin besar ukuran operator maka proses komputasi semakin besar kebutuhan sumberdaya komputasinya (prosesor dan memori).

### 3.1.1 Konvolusi menggunakan C#

Konvolusi merupakan operasi matematika sederhana namun penting untuk operasi citra berbasis area. Proses konvolusi melibatkan sebuah matriks citra dijital dan sebuah matris berukuran kecil yang disebut operator. Operasionalya diilustrasikan sebagai berikut. Diketahui matriks citra $I$ dan matriks operator $K$ seperti ditunjukkan pada Gambar 3.1. Misalkan matriks hasil operasi konvolusi disebut $O$. Maka

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $I_{11}$ | $I_{12}$ | $I_{13}$ | $I_{14}$ | $I_{15}$ | $I_{16}$ | $I_{17}$ | $I_{18}$ | $I_{19}$ |
| $I_{21}$ | $I_{22}$ | $I_{23}$ | $I_{24}$ | $I_{25}$ | $I_{26}$ | $I_{27}$ | $I_{28}$ | $I_{29}$ |
| $I_{31}$ | $I_{32}$ | $I_{33}$ | $I_{34}$ | $I_{35}$ | $I_{36}$ | $I_{37}$ | $I_{38}$ | $I_{39}$ |
| $I_{41}$ | $I_{42}$ | $I_{43}$ | $I_{44}$ | $I_{45}$ | $I_{46}$ | $I_{47}$ | $I_{48}$ | $I_{49}$ |
| $I_{51}$ | $I_{52}$ | $I_{53}$ | $I_{54}$ | $I_{55}$ | $I_{56}$ | $I_{57}$ | $I_{58}$ | $I_{59}$ |
| $I_{61}$ | $I_{62}$ | $I_{63}$ | $I_{64}$ | $I_{65}$ | $I_{66}$ | $I_{67}$ | $I_{68}$ | $I_{69}$ |

| | | |
|---|---|---|
| $K_{11}$ | $K_{12}$ | $K_{13}$ |
| $K_{21}$ | $K_{22}$ | $K_{23}$ |

Gambar 3.1 Matriks Citra dan Operator

operasi konvolusi untuk menghitung nilai piksel baru pada lokasi $(5, 7)$ dapat dihitung dengan cara

$$O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23}$$

Operasi konvolusi dilakukan pada sluruh piksel pada citra dijital seolah-olah matriks operator ditumpukkan pada matriks citra (*superimpose*) kemudian dilakukan operasi konvolusi dan seterusnya bergerak satu piksel ke kanan hingga mencapai piksel terahir di kanan bawah. Perhatikan kondisi yang disebut dengan *boundary problems* (lihat materi kuliah).

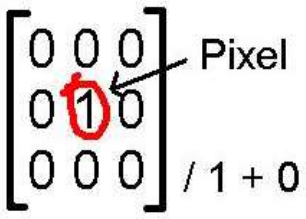
Secara matematis formulasi konvolusi dapat dituliskan sebagai:

$$O_{ij} = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i + k - 1, j + l - 1)K(k, l)$$

untuk nilai $i$ dari 1 ke $M - m + 1$ dan $j$ dari 1 ke $N - n + 1$. $O$ adalah matriks citra output, $I$ adalah matriks citra asli/original dan $K$ adalah matriks kernel atau operator.

### 3.1.2 Anatomi Matriks Konvolusi Dalam Pemrograman

Pada dasarnya matriks konvolusi atau *filter* terdiri dari elemen matriks itu sendiri, misalnya untuk *filter* 3×3, maka terdapat sembilan elemen matriks, yang terdiri dari elemen-elemen tetangga sebanyak delapan elemen dan elemen fokus piksel yang terdapat di tengah-tengah. Selain itu ada elemen eksternal yaitu faktor dan *offset*. Faktor adalah sebuah nilai konstanta yang menjadi pembagi dari jumlah seluruh elemen matriks, sedangkan *offset* adalah nilai yang ditambahkan setelah tersebut. Gambar 3.2 mengilustrasikan sebuah matriks identitas untuk sebuah proses konvolusi.




Gambar 3.2 Anatomi Filter

**Latihan 301**

Pada latihan ini peserta akan membuat program pengolahan citra berbasis area dengan menggunakan *pointer*. Teknik ini digunakan karena kinerjanya JAUH lebih baik dibandingkan dengan cara konvensional yaitu menggunakan perintah GDI+ `SetPixel` dan `GetPixel`. Ilustrasi tampilannya dapat dilihat pada Gambar 3.3.

1. Salinlah folder **latihan202** ke **latihan301**, lakukan perubahan seperlunya seperti yang anda lakukan sebelumnya

2. Buatlah elemen `textBox` 3 × 3 dan beri nama `textBoxK11` sampai dengan `textBoxK33`. Jangan lupa untuk memberi nilai awal dari setiap elemen.

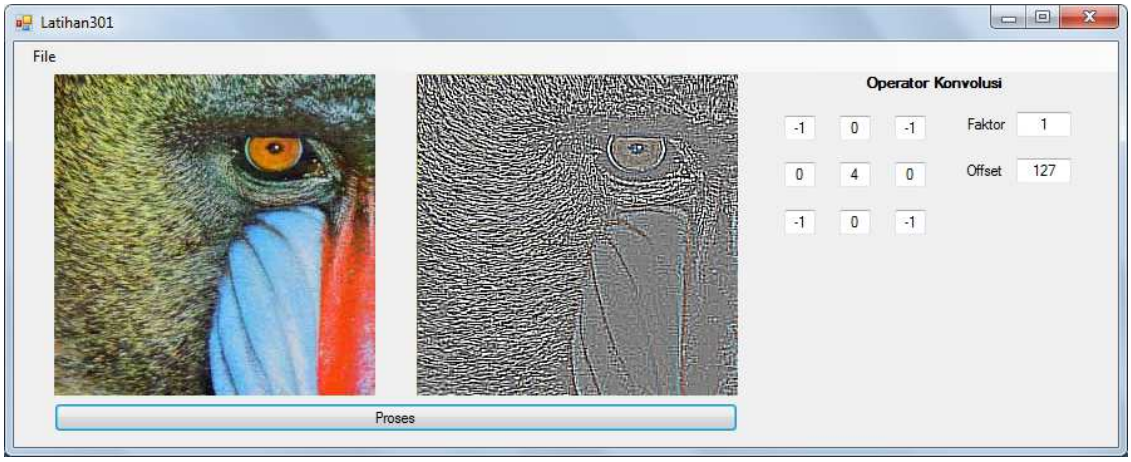
3. Buatlah elemen `textBox` untuk faktor dan *offset*, beri nama `textBoxFaktor` dan `textBoxOffset`

4. Ketikkan potongan program berikut: (1) Kelas `Operator`, untuk mendefinisikan nilai matriks Kernel, (2) fungsi `Konvolusi` untuk proses menggunakan pointer yang mengembalikan nilai (*return*) bitmap, dan (3) *private method* yang dijalankan pada saat tombol **Proses** di-klik.

```
1  public class Operator
2  {
3      public int TopLeft = 0, TopMid = 0, TopRight = 0;
4      public int MidLeft = 0, Pixel = 1, MidRight = 0;
5      public int BottomLeft = 0, BottomMid = 0, BottomRight = 0;
6      public int Factor = 1;
7      public int Offset = 0;
8      public void SetAll(int nVal)
9      {
10         TopLeft = TopMid = TopRight = MidLeft = Pixel = MidRight =
               BottomLeft = BottomMid = BottomRight = nVal;
11     }
12 }
13 public static Bitmap Konvolusi(Bitmap b, Operator m)
14 {
15     Bitmap bSrc = (Bitmap)b.Clone();
16     if (m.Factor == 0) return b;
17     BitmapData bmData = b.LockBits(new Rectangle(0, 0, b.Width, b.
           Height), ImageLockMode.ReadWrite, PixelFormat.
           Format24bppRgb);
```

```
18      BitmapData bmSrc = bSrc.LockBits(new Rectangle(0, 0, bSrc.Width
            , bSrc.Height), ImageLockMode.ReadWrite, PixelFormat.
            Format24bppRgb);
19      int stride = bmData.Stride;
20      int stride2 = stride * 2;
21      System.IntPtr Scan0 = bmData.Scan0;
22      System.IntPtr SrcScan0 = bmSrc.Scan0;
23      unsafe
24      {
25          byte* p = (byte*)(void*)Scan0;
26          byte* pSrc = (byte*)(void*)SrcScan0;
27          int nOffset = stride + 6 - b.Width * 3;
28          int nWidth = b.Width - 2;
29          int nHeight = b.Height - 2;
30          int nPixel;
31          for (int y = 0; y < nHeight; ++y)
32          {
33              for (int x = 0; x < nWidth; ++x)
34              {
35                  nPixel = ((((pSrc[2] * m.TopLeft) + (pSrc[5] * m.
                        TopMid) + (pSrc[8] * m.TopRight) +
36                                  (pSrc[2 + stride] * m.MidLeft) + (pSrc[5
                                     + stride] * m.Pixel) + (pSrc[8 +
                                     stride] * m.MidRight) +
37                                  (pSrc[2 + stride2] * m.BottomLeft) + (
                                     pSrc[5 + stride2] * m.BottomMid) + (
                                     pSrc[8 + stride2] * m.BottomRight)) /
                                      m.Factor) + m.Offset);
38                  if (nPixel < 0) nPixel = 0;
39                  if (nPixel > 255) nPixel = 255;
40                  p[5 + stride] = (byte)nPixel;
41                  nPixel = ((((pSrc[1] * m.TopLeft) + (pSrc[4] * m.
                        TopMid) + (pSrc[7] * m.TopRight) +
42                                  (pSrc[1 + stride] * m.MidLeft) + (pSrc[4
                                     + stride] * m.Pixel) + (pSrc[7 +
                                     stride] * m.MidRight) +
43                                  (pSrc[1 + stride2] * m.BottomLeft) + (
                                     pSrc[4 + stride2] * m.BottomMid) + (
                                     pSrc[7 + stride2] * m.BottomRight)) /
                                      m.Factor) + m.Offset);
44                  if (nPixel < 0) nPixel = 0;
45                  if (nPixel > 255) nPixel = 255;
46                  p[4 + stride] = (byte)nPixel;
47                  nPixel = ((((pSrc[0] * m.TopLeft) + (pSrc[3] * m.
                        TopMid) + (pSrc[6] * m.TopRight) +
48                                  (pSrc[0 + stride] * m.MidLeft) + (pSrc[3
                                     + stride] * m.Pixel) + (pSrc[6 +
                                     stride] * m.MidRight) +
49                                  (pSrc[0 + stride2] * m.BottomLeft) + (
                                     pSrc[3 + stride2] * m.BottomMid) + (
                                     pSrc[6 + stride2] * m.BottomRight)) /
                                      m.Factor) + m.Offset);
50                  if (nPixel < 0) nPixel = 0;
51                  if (nPixel > 255) nPixel = 255;
52                  p[3 + stride] = (byte)nPixel;
53                  p += 3;
```

```
54                  pSrc += 3;
55              }
56              p += nOffset;
57              pSrc += nOffset;
58          }
59      }
60      b.UnlockBits(bmData);
61      bSrc.UnlockBits(bmSrc);
62      return b;
63  }
64  private void buttonProses_Click(object sender, EventArgs e)
65  {
66      Operator Filter = new Operator();
67      Filter.TopLeft = Convert.ToInt16(textBoxK11.Text);
68      Filter.TopMid = Convert.ToInt16(textBoxK12.Text);
69      Filter.TopRight = Convert.ToInt16(textBoxK13.Text);
70      Filter.MidLeft = Convert.ToInt16(textBoxK21.Text);
71      Filter.Pixel = Convert.ToInt16(textBoxK22.Text);
72      Filter.MidRight = Convert.ToInt16(textBoxK23.Text);
73      Filter.BottomLeft = Convert.ToInt16(textBoxK31.Text);
74      Filter.BottomMid = Convert.ToInt16(textBoxK32.Text);
75      Filter.BottomRight = Convert.ToInt16(textBoxK33.Text);
76      Filter.Factor = Convert.ToInt16(textBoxFaktor.Text);
77      Filter.Offset = Convert.ToInt16(textBoxOffset.Text);
78      pictureBoxHasil.Image = (Bitmap)Konvolusi((Bitmap)
            pictureBoxAsli.Image, Filter);
79  }
```




Gambar 3.3 Ilustrasi Latihan 301

### 3.1.3 Nilai-nilai *Mask* Untuk Beberapa Operasi Citra

Berikut ini adalah Tabel 3.1 yang menunjukkan nilai-nilai *mask* untuk operasi *blurring*, *sharpening*, *embossing* dan *edge detection*.

Tabel 3.1 Nilai Mask Untuk Beberapa Operasi Citra

| Blurring | | | Sharpening | | | Embossing | | | Edging | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 0 | -2 | 0 | -1 | 0 | -1 | -1 | -1 | -1 |
| 2 | 4 | 2 | -2 | 11 | -2 | 0 | 4 | 0 | 0 | 0 | 0 |
| 1 | 2 | 1 | 0 | 2 | 0 | -1 | 0 | -1 | 1 | 1 | 1 |
| *faktor*=16 | | | *faktor*=3 | | | *faktor*=1 | | | *faktor*=1 | | |
| *offset*=0 | | | *offset*=0 | | | *offset*=127 | | | *offset*=127 | | |

### 3.1.4 Operasi Citra Berbasis Area Tanpa *Mask*

Operasi citra tanpa menggunakan *mask* merupakan hal yang umum dilakukan untuk menghilangkan gangguan atau *noise* pada citra dijital. Operasi dasar yang berkaitan dengan ini adalah *Median Filtering* dan *Maximum/Minimum Filtering*. Pada dasarnya operasi dasarnya mirip dengan konvolusi hanya dalam hal ini tidak digunakan matriks operator, namun tetap menerapkan konsep pemrosesan area dengan memperhatikan piksel-piksel tetangga.

**Latihan 302**
Modifikasi **Latihan 301** menjadi **Latihan 302** yang menerapkan algoritma *Median Filtering* dan *Maximum/Minimum Filtering* sesuai dengan teori yang diajarkan di kelas..

### 3.1.5 *Morphological Filtering*

*Morphological filtering* atau filter morfologis adalah operasi pengolahan citra yang banyak digunakan terutama untuk keperluan pembersihan (*cleaning*) citra dijital dan juga untuk analisis citra dijital. Pada dasarnya, proses filter morfologis sama seperti konvolusi dimana *mask* dalam filter morfologis disebut dengan *structuring element*. Secara konseptual filter morfologis diturunkan berdasarkan pada operasi himpunan matematika.

Bentuk *structuring element* bisa berupa *cube*, *box*, *circle*, *plus*, dan *cross*. Masing-masing akan memberikan efek yang berbeda walaupun belum tentu signifikan. Pada umumnya ukuran *structuring element* adalah $3 \times 3$. Semakin besar ukurannya kinerja filteringnya relatif semakin baik namun semakin berat beban komputasinya.

Pada umumnya operasi filter morfologis dilakukan pada domain citra biner Ada dua operasi dasar filter morfologis yaitu erosi dan dilasi. Proses erosi yang proses mengkikis citra sedemikian rupa sehingga citra menjadi tipis. Proses dilasi adalah proses membanjiri citra sehingga citra menjadi tebal. Turunan kedua proses itu adalah proses *opening* dan proses *closing*. Proses *opening* adalah proses filter morfologis yang merupakan urutan pelaksanaan operasi dasar erosi dilanjutkan dengan

dilasi. Sebaliknya operasi *closing* adalah urutan operasi dilasi kemudian erosi.

**Latihan 303**
Berdasarkan potongan program yang telah diberikan, buatkan program untuk melakukan proses Filter Morfologis. Perhatikan bahwa sebaiknya citrta dijital dikonversikan dulu ke biner.

## 3.2 Tugas Praktikum

1. Apakah yang dimaksud dengan *pixel neighborhood*?
2. Jelaskan proses konvolusi secara empiris !
3. Jika sebuah citra berukuran $m \times n$ dikonvolusikan dengan kernel berukuran $p \times q$, tentukan banyaknya operasi komputasi yang dilakukan !
4. Sebutkan efek-efek lain yang bisa dibuat melalui operasi pengolahan citra berbasis area
5. Buatlah program terintegrasi **Tugas301** yang mengimplementasikan selengkap-lengkapnya operasi pengolahan citra berbasis area yang sudah dijelaskan di atas. Gunakan kreativitas anda ...

Praktikum 4

# Pemrosesan Citra Berbasis Geometri

Pada praktikum ini peserta akan mempraktekkan algoritma pemrosesan citra yang berkaitan dnegan operasi geometri citra. Setelah melaksanakan praktikum ini peserta akan :

1. Memahami konsep transformasi dasar geometri citra: Rotasi, Translasi, Penskalaan (*stretch* atau *scaling*) dan Refleksi (*flipping*)
2. Membuat program dalam bahasa pemrograman C# untuk melakukan operasi transformasi geometri citra dijital tersebut

## 4.1 Teori dan Latihan

Transformasi geometri citra adalah proses perubahan lokasi piksel berdasarkan kriteria tertentu. Transformasi rotasi adalah proses perubahan lokasi piksel dimana lokasi piksel berputar berdasarkan pada sudut dan posisi titik pusat rotasi. Transformasi translasi adalah pergeseran citra berdasarkan sumbu horizontal, vertikal, diagonal, maupun secara bebas. Transformasi geometri penskalaan berupa perbesaran (*zoom-in*) atau pengecilan (*zoom-out*) citra. Transformasi refleksi atau *flipping* adalah proses pencerminan objek pada sebuah sumbu tertentu. Hasil proses transformasi geometris citra dijital dipengaruhi oleh teknik interpolasi yang diterapkan. Interpolasi ini secara umum akan menghaluskan hasil tranformasi sehingga hasilnya sesuai dengan yang diharapkan dan lebih realistis dan bersih dari *noise*.

### 4.1.1 Konsep Dasar Transformasi 2D dalam .NET

Transformasi citra 2D dalam .NET dilakukan dengan menggunakan sebuah kelas `Matrix` yang terdapat dalam *namespace* `System.Drawing.Drawing2D`. Kelas `Matrix` berisi 6 elemen yang diatur dalam 3 baris 2 kolom. Sebagai contoh diketahui sebuah matriks baku yang dibentuk olek konstruktor baku yang berisi nilai $(1, 0, 0, 1, 0, 0)$. Dalam representasi matriksnya, nilai-nilai tersebut dituliskan sebagai $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$ yang

merupakan penyederhanaan dari $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ dimana kolom terakhir (ketiga) selalu bernilai $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. Sebuah operasi transformasi translasi dengan besaran dan arah 3 pada sumbu $x$ dan 2 pada sumbu $y$ direpresentasikan menjadi : $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 3 & 2 \end{bmatrix}$ tetapi secara internal (pada pemrograman komputer) dituliskan sebagai $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}$

Untuk operasi transformasi dengan matriks citra maka operasinya adalah matriks citra dikalikan dengan matriks transformasi. Contoh, apabila diketahui matriks citra terdiri dari 4 titik yaitu $(1,1)(2,3)(5,0)(6,7)$, maka representasinya dalam bentuk matriks adalah matriks 4 baris 2 kolom yaitu $\begin{bmatrix} 1 & 1 \\ 2 & 3 \\ 5 & 0 \\ 6 & 7 \end{bmatrix}$ dimana secara internal direpresentasikan sebagai $\begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 1 \\ 5 & 0 & 1 \\ 6 & 7 & 1 \end{bmatrix}$. Apabila matriks citra tersebut ditransformasikan dengan sebuah matriks transformasi, maka operasinya adalah sebagai berikut:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 1 \\ 5 & 0 & 1 \\ 6 & 7 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 1 \\ 5 & 5 & 1 \\ 8 & 2 & 1 \\ 9 & 9 & 1 \end{bmatrix}$$

Dengan mengabaikan kolom terakhir maka hasil dari transformasi tersebut adalah titik-titik $(4,3)(5,5)(8,2)$ dan $(9,9)$.

Transformasi komposisi dibangun melalui perkalian dua atau lebih matriks. Contoh sebuah matriks transformasi penskalaan sebesar 2 pada sumbu $x$ dan 3 pada sumbu $y$ adalah $\begin{bmatrix} 2 & 0 \\ 0 & 3 \\ 0 & 0 \end{bmatrix}$, yang secara internal direpresentasikan sebagai $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Apabila dilakukan operasi translasi diikuti dengan penskalaan, maka kedua matriks tersebut dikomposisikan sebagai berikut:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Perlu dicatat bahwa perkalian matriks tidak komutatif.

### 4.1.2 Transformasi Citra 2D Menggunakan C#

Kelas `Matrix` mengimplementasikan method-method `RotateTransform`, `TranslateTransform`, `ScaleTransform` dan `MultiplyTransform`. Berikut ini diberikan contoh kelas untuk merotasikan sebuah citra.

```
1  private static Bitmap RotateImage(Image image, PointF offset, float
       angle)
2  {
3      var rotatedBmp = new Bitmap(image.Width, image.Height);
4      rotatedBmp.SetResolution(image.HorizontalResolution, image.
           VerticalResolution);
5      var g = Graphics.FromImage(rotatedBmp);
6      g.TranslateTransform(offset.X, offset.Y);
7      g.RotateTransform(angle);
8      g.TranslateTransform(-offset.X, -offset.Y);
9      g.DrawImage(image, new PointF(0, 0));
10     return rotatedBmp;
11 }
```

**Baris 1** Kelas `RotateImage` berjenis `Bitmap` (akan mengembalikan nilai `Bitmap`) dengan argumen `image` berjenis `Image`, `offset` berjenis `PointF` (struktur titik dengan koordinat $X$ dan $Y$ berjenis bilangan desimal dan `angle` berjenis `float`.

**Baris 3** Variabel `rotateBMP` merupakan diturunkan (*inherited*) dari kelas `Bitmap` baru berukuran `image`

**Baris 4** Resolusi hasil yang disimpan pada `rotateBMP` disesuai dengan resolusi layar komputer (horizontal dan vertikal)

**Baris 5** Deklarasi variabel `g`, berkelas `Graphics` yang berasal dari `rotateBMP`

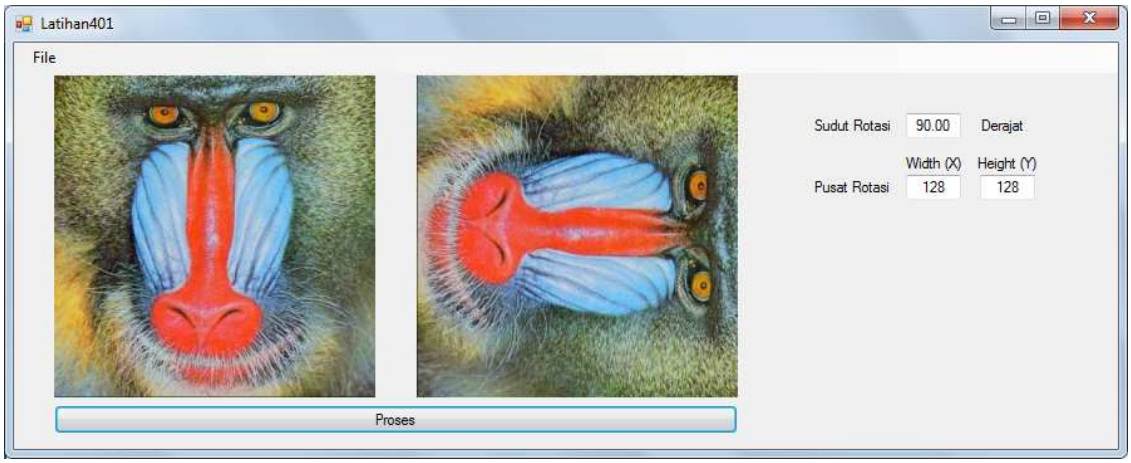
**Baris 6-8** Translasikan pusat rotasi ke origin, lakukan proses rotasi, kembalikan origin ke semula (retranslasi)

**Baris 9-10** *Redraw* `image` sesuai dengan hasil transformasi dan kembalikan hasil

Pada **Latihan 401** berikut ini akan dibuat program lengkap untuk proses rotasi citra. Dua hal penting yang berperan adalah pusat dan sudut rotasi dalam derajat. Pusat rotasi dihitung sebagai titik tengah citra (bukan titik tengah `pictureBox`). Sebagai ilustrasi dapat dilihat pada Gambar 4.1.

**Latihan 401**

1. Ambil program dari proyek sebelumnya, modifikasi dan beri nama **Latihan401**
2. Buat elemen `textBoxAngle` untuk nilai sudut rotasi, elemen `textBoxPusatX` dan `textBoxPusatY` untuk pusat rotasi pada sumbu $X$ dan $Y$.
3. Mengingat nilai Pusat Rotasi harus ditentukan berdasarkan citra yang ditampilkan atau pada saat program dijalankan, maka tambahkan kode pada *event* saat program di *load* dan pada saat file citra dibuka melalui menu.




Gambar 4.1 Ilustrasi Latihan 401

```
1  private void openImageToolStripMenuItem_Click(object sender,
       EventArgs e)
2  {
3      OpenFileDialog open = new OpenFileDialog();
4      open.Filter = "Image Files(*.jpg; *.bmp)|*.jpg; *.bmp";
5      if (open.ShowDialog() == DialogResult.OK)
6      {
7          pictureBoxAsli.Image = new Bitmap(open.FileName);
8          Bitmap bmp = (Bitmap)pictureBoxAsli.Image;
9          textBoxPusatX.Text = (bmp.Width / 2).ToString();
10         textBoxPusatY.Text = (bmp.Height / 2).ToString();
11     }
12 }
13 private void Form1_Load(object sender, EventArgs e)
14 {
15     Bitmap bmp = (Bitmap)pictureBoxAsli.Image;
16     textBoxPusatX.Text=(bmp.Width/2).ToString();
17     textBoxPusatY.Text=(bmp.Height/2).ToString();
18 }
```

4. Kemudian tambahkan kelas `RotateImage` yang telah dijelaskan sebelumnya

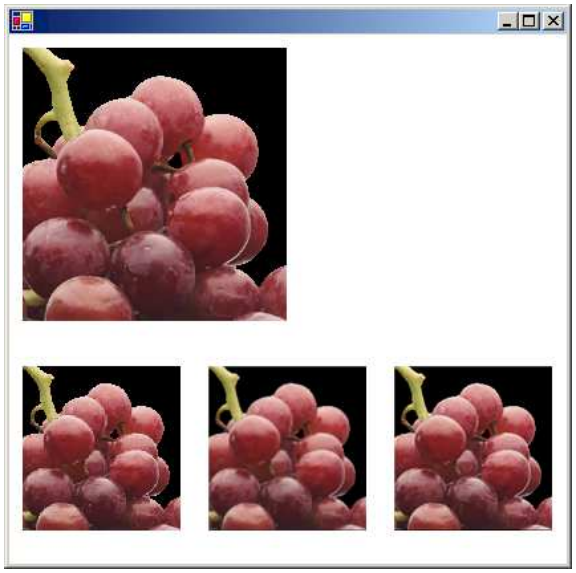
5. Proses yang dilakukan adalah memanggil kelas `RotateImage` dengan cara sebagai berikut:

```
1  private void buttonProses_Click(object sender, EventArgs e)
2  {
3      float Sudut = (float)Convert.ToSingle(textBoxAngle.Text);
4      PointF PusatRotasi=new PointF((float)Convert.ToSingle(
           textBoxPusatX.Text),(float)Convert.ToSingle(textBoxPusatY.
           Text));
5      pictureBoxHasil.Image = (Bitmap)RotateImage((Bitmap)
           pictureBoxAsli.Image, PusatRotasi, Sudut);
6  }
```

### 4.1.3 Interpolasi Citra dalam .NET

Interpolasi adalah elemen penting dalam mengontrol hasil operasi transformasi geometri. Mode interpolasi yang disediakan adalah *NearestNeighbor*, *Bilinear*, *HighQualityBilinear*, *Bicubic*, *HighQualityBicubic*. Secara teoritis, operasi geometrik citra adalah memetakan (*mapping*) lokasi piksel-piksel pada citra asli ke lokasi citra hasil atau sebaliknya. Tentunya proses pemetaan ini dalam beberapa hal akan menghasilkan kualitas yang bervariasi bergantung pada teknik interpolasi yang digunakan. Semakin canggih teknik interpolasi yang digunakan maka hasil akan semakin baik namun membutuhkan waktu pemrosesan yang lebih lama. *NearestNeighbor* adalah mode paling rendah kualitasnya dan *HighQualityBicubic* adalah mode yang paling baik kualitasnya. Berikut ini potongan program yang menunjukkan cara dan hasil proses interpolasi (Gambar 4.2).

```
1  Image image = new Bitmap("GrapeBunch.bmp");
2  int width = image.Width;
3  int height = image.Height;
4  e.Graphics.DrawImage(image, new Rectangle(10, 10, width, height), 0,
       0, width, height, GraphicsUnit.Pixel, null);
5
6  e.Graphics.InterpolationMode = InterpolationMode.NearestNeighbor;
7  e.Graphics.DrawImage(image, new Rectangle(10, 250, (int)(0.6 * width),
       (int)(0.6 * height)), 0, 0, width, height, GraphicsUnit.Pixel);
8
9  e.Graphics.InterpolationMode = InterpolationMode.HighQualityBilinear;
10 e.Graphics.DrawImage(image,new Rectangle(150, 250, (int)(0.6 * width),
       (int)(0.6 * height)), 0, 0, width, height,      GraphicsUnit.Pixel);
11
12 e.Graphics.InterpolationMode = InterpolationMode.HighQualityBicubic;
13 e.Graphics.DrawImage(image, new Rectangle(290, 250, (int)(0.6 * width),
       (int)(0.6 * height)),0, 0,width, height, GraphicsUnit.Pixel);
```



Gambar 4.2 Contoh Kualitas Citra Hasil Interpolasi

## 4.2 Tugas Praktikum

Buatlah program lengkap **Tugas401** untuk proses transformasi geometrik citra dijital mencakup operasi-operasi berikut:

1. **Translasi**. Gunakan method `TranslateTransform`.
2. **Rotasi**. Sudah dilakukan pada **Latihan 401**
3. **Penskalaan**. Gunakan method `ScaleTransform`.
4. **Refleksi**. Fungsi ini tidak disediakan dalam kelas `Matrix` .NET. Untuk itu bisa digunakan method `MultiplyTransfom` dimana untuk refleksi pada sumbu $x$ gunakan matriks transformasi $(1, 0, 0, -1, 0, 0)$ dan refleksi pada sumbu $y$ gunakan matriks transformasi $(-1, 0, 0, 1, 0, 0)$. Contohnya sebagai berikut:

```
g.MultiplyTransform(new Matrix(1, 0, 0, -1, 0, 40));
```

5. **Interpolasi**. Lengkapi program anda dengan fasilitas interpolasi yang disediakan dalam pemrograman .NET.

Praktikum 5

# Pemrosesan Citra Berbasis Frame

Pada praktikum ini peserta akan mempraktekkan algoritma pemrosesan citra didasarkan pada operasi yang dilakukan antara dua atau lebih citra dijital. Pada dasarnya operasi yang dilakukan mirip dengan operasi berbasis titik, namun elemen kedua dalam operasi ini bukanlah konstanta tunggal namun berupa matriks citra. Setelah melaksanakan praktikum ini peserta akan :

1. Memahami konsep operasi dua citra menggunakan operator aritmatika, logika dan operator lainnya.
2. Mampu membuat program dalam bahasa C# untuk operasi *frame process* : *Add*, *Subtract*, *Difference*, *Multiply*, *Average*, *Cross Fading*, *Min* dan *Max*, *Amplitude*, AND, OR dan XOR.

## 5.1 Teori dan Latihan

Operasi pengolahan citra *frame process*, dikenal juga sebagai aritmatika citra (*image arithmetic*), adalah sebuah operasi yang melibatkan dua buah citra dimana setiap piksel dari kedua citra tersebut diproses dengan operator tertentu untuk menghasilkan citra ketiga.

### 5.1.1 Formulasi Operasi Aritmatika Citra

Secara sederhana operasi aritmatika citra dapat dituliskan sebagai berikut:

$$R(i,j) = P(i,j) \odot Q(i,j)$$

dimana $\odot$ adalah operator aritmatika citra yang bisa berupa a) Penjumlahan (*Add*), b)Kurang (*Subtract*), c) Selisih (*Difference*), d) Kali(*Multiply*), e) Rata-rata (*Average*), f) Pembobotan (*Cross Fading*, *Alpha Channel*), g) *Min* dan *Max*, h) *Amplitude*, i) *Logical* AND, j) *Logical* OR atau k) *Logical* XOR. Perlu diperhatikan bahwa operasi beberapa operator akan menghasilkan kondisi *overflow*, *underflow* dan desimal yang membutuhkan penanganan khusus seperti yang sudah dilakukan pada praktikum sebelumnya.

Sebagai ilustrasi, operator "kurang" yang diterapkan pada dua buah citra akan akan mengurangi nilai piksel pada setiap lokasi di citra pertama dengan nilai piksel pada setiap lokasi di citra kedua. Lihat contoh berikut:

$$\begin{bmatrix} 7 & 7 & 6 & 6 \\ 5 & 4 & 3 & 2 \\ 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 1 & 3 & 0 \\ 3 & 7 & 1 & 2 \\ 4 & 4 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 7 & 6 & 4 & 3 \\ 3 & 3 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 5 & 4 \end{bmatrix}$$

### 5.1.2 Jenis-jenis Aritmatika Citra

1. **Add**

```
result[x][y].r = min(img2[x][y].r + img3[x][y].r, 255);
result[x][y].g = min(img2[x][y].g + img3[x][y].g, 255);
result[x][y].b = min(img2[x][y].b + img3[x][y].b, 255);
```

2. **Subtract**

```
result[x][y].r = max(img2[x][y].r - img1[x][y].r, 0);
result[x][y].g = max(img2[x][y].g - img1[x][y].g, 0);
result[x][y].b = max(img2[x][y].b - img1[x][y].b, 0);
```

3. **Difference**

```
result[x][y].r = abs(img1[x][y].r - img2[x][y].r);
result[x][y].g = abs(img1[x][y].g - img2[x][y].g);
result[x][y].b = abs(img1[x][y].b - img2[x][y].b);
```

4. **Multiply**

```
result[x][y].r=int(255*(img2[x][y].r/255.0*img1[x][y].r/255.0));
result[x][y].g=int(255*(img2[x][y].g/255.0*img1[x][y].g/255.0));
result[x][y].b=int(255*(img2[x][y].b/255.0*img1[x][y].b/255.0));
```

5. **Average**

```
result[x][y].r = (img1[x][y].r + img2[x][y].r) / 2;
result[x][y].g = (img1[x][y].g + img2[x][y].g) / 2;
result[x][y].b = (img1[x][y].b + img2[x][y].b) / 2;
```

6. **Cross Fading**

```
result[x][y].r = int(img1[x][y].r*weight+img2[x][y].r*(1-weight));
result[x][y].g = int(img1[x][y].g*weight+img2[x][y].g*(1-weight));
result[x][y].b = int(img1[x][y].b*weight+img2[x][y].b*(1-weight));
```

7. **Min dan Max**

```
result[x][y].r = min(img1[x][y].r, img2[x][y].r);
result[x][y].g = min(img1[x][y].g, img2[x][y].g);
result[x][y].b = min(img1[x][y].b, img2[x][y].b);
```

8. **Amplitude**

```
result[x][y].r = int(sqrt(double(img1[x][y].r * img1[x][y].r +
    img2[x][y].r * img2[x][y].r)) / sqrt(2.0));
result[x][y].g = int(sqrt(double(img1[x][y].g * img1[x][y].g +
    img2[x][y].g * img2[x][y].g)) / sqrt(2.0));
result[x][y].b = int(sqrt(double(img1[x][y].b * img1[x][y].b +
    img2[x][y].b * img2[x][y].b)) / sqrt(2.0));
```

   Amplitude dihitung menggunakan formula amplitudo $\sqrt{x^2 + y^2}$ antara kanal warna. Karena operasi ini akan menghasilkan nilai 1.41 kali lebih besar dari 255, maka hasil operasinya dibagi dengan 1.41 atau $\sqrt{2}$. Perhatikan proses konversi yang perlu dilakukan jangan sampai terjadi kesalahan.

9. **Logical AND**

```
result[x][y].r = img1[x][y].r & img2[x][y].r;
result[x][y].g = img1[x][y].g & img2[x][y].g;
result[x][y].b = img1[x][y].b & img2[x][y].b;
```

10. **Logical OR**

```
result[x][y].r = img1[x][y].r | img2[x][y].r;
result[x][y].g = img1[x][y].g | img2[x][y].g;
result[x][y].b = img1[x][y].b | img2[x][y].b;
```

11. **Logical XOR**

```
result[x][y].r = img1[x][y].r ^ img2[x][y].r;
result[x][y].g = img1[x][y].g ^ img2[x][y].g;
result[x][y].b = img1[x][y].b ^ img2[x][y].b;
```

Berikut ini **Latihan 501** yaitu proses subtraksi dua citra menggunakan *pointer* dan GDI+. Ilustrasinya dapat dilihat pada Gambar 5.1.

**Latihan 501**

1. Modifikasi program dari proyek sebelumnya, modifikasi dan beri nama **Latihan501**

2. Buatlah 3 elemen `pictureBox`, beri nama `pictureBoxImg1`, `pictureBoxImg2`, `pictureBoxImgResult`

3. Tambahkan menu untuk me*load* masing-masing image yaitu **Open Image1** dan **Open Image2**

4. Tambahkan elemen `Button` untuk `Proses`, untuk proses menggunakan *pointer*, dan `ProsesGDI`, untuk proses menggunakan GDI+

5. Tambahkan kode untuk method `Subtraction` yang menggunakan *pointer* berikut ini:

```
Bitmap bmp1;
public Bitmap Subtraction(Bitmap bmp2)
{
  BitmapData bmpData1 = bmp1.LockBits(new Rectangle(0, 0, bmp1.
      Width, bmp1.Height), ImageLockMode.ReadWrite, PixelFormat.
      Format24bppRgb);
  BitmapData bmpData2 = bmp2.LockBits(new Rectangle(0, 0, bmp2.
      Width, bmp2.Height), ImageLockMode.ReadWrite, PixelFormat.
      Format24bppRgb);
  int width = bmpData1.Width;
  int height = bmpData1.Height;
  Bitmap bmpresult = new Bitmap(width, height);
  BitmapData bmpData3 = bmpresult.LockBits(new Rectangle(0, 0,
      bmpresult.Width, bmpresult.Height), ImageLockMode.ReadWrite,
       PixelFormat.Format24bppRgb);
  unsafe
  {
    int remain1 = bmpData1.Stride - bmpData1.Width * 3;
    int remain2 = bmpData2.Stride - bmpData2.Width * 3;
    int remain3 = bmpData3.Stride - bmpData3.Width * 3;
    byte* ptr1 = (byte*)bmpData1.Scan0;
    byte* ptr2 = (byte*)bmpData2.Scan0;
    byte* ptr3 = (byte*)bmpData3.Scan0;
    for (int i = 0; i < height; i++)
    {
       for (int j = 0; j < width*3 ; j++)
       {
          ptr3[0] = (byte)Math.Max((ptr1[0] - ptr2[0]),0);
          ++ptr1;
          ++ptr2;
          ++ptr3;
       }
       ptr1 += remain1;
       ptr2 += remain2;
       ptr3 += remain3;
     }
   }
   bmp1.UnlockBits(bmpData1);
   bmp2.UnlockBits(bmpData2);
   bmpresult.UnlockBits(bmpData3);
   return bmpresult;
}
```

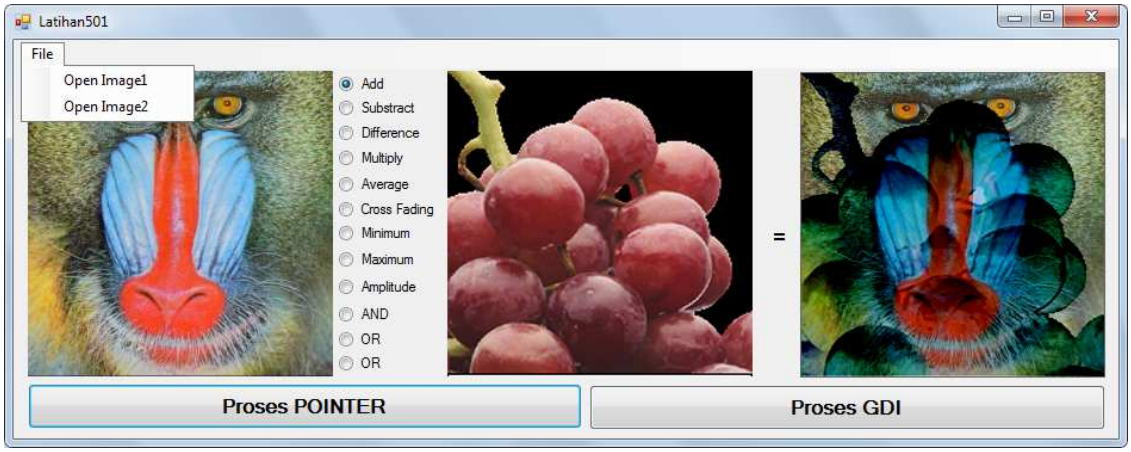
6. Tambahkan kode untuk method `SubtractionGDI` yang menggunakan GDI+ berikut ini:

```
public Bitmap SubtractionGDI(Bitmap bmp2)
{
  int width = bmp1.Width;
  int height = bmp1.Height;
  int r,g,b;
  Bitmap bmpresult = new Bitmap(width, height);
  for (int i = 0; i < height; i++)
  {
     for (int j = 0; j < width; j++)
```

```
10          {
11              r=Math.Max(bmp1.GetPixel(i,j).R-bmp2.GetPixel(i,j).R,0);
12              g=Math.Max(bmp1.GetPixel(i,j).G-bmp2.GetPixel(i,j).G,0);
13              b=Math.Max(bmp1.GetPixel(i,j).B-bmp2.GetPixel(i,j).B,0);
14              bmpresult.SetPixel(i,j,Color.FromArgb(r,g,b));
15          }
16      }
17      return bmpresult;
18  }
```

7. Isikan kode untuk masing-masing `button Proses` sebagai berikut:

```
1  private void buttonProses_Click(object sender, EventArgs e)
2  {
3      bmp1 = (Bitmap)pictureBoxImg1.Image;
4      Bitmap bmp2=(Bitmap)pictureBoxImg2.Image;
5      pictureBoxResult.Image = Subtraction(bmp2);
6  }
7  private void buttonProsesGDI_Click(object sender, EventArgs e)
8  {
9      bmp1 = (Bitmap)pictureBoxImg1.Image;
10     Bitmap bmp2 = (Bitmap)pictureBoxImg2.Image;
11     pictureBoxResult.Image = SubtractionGDI(bmp2);
12 }
```




Gambar 5.1 Ilustrasi Latihan 501

## 5.2 Tugas Praktikum

Buatlah program **Tugas501** yang mirip dengan **Latihan 501** namun dilengkapi dengan fungsi-fungsi operasi transformasi citra berbasis *frame* lainnya yaitu *Add*, *Difference*, *Multiply*, *Average*, *Cross Fading*, *Min* dan *Max*, *Amplitude*, AND, OR dan XOR.

Praktikum 6

# Analisis Citra Digital

Analisis citra digital adalah penyelidikan terhadap suatu citra digital melalui proses-proses tertentu untuk mengetahui ciri-ciri tertentu dari suatu citra atau dengan kata lain disebut dengan pemisahan fitur. Beberapa teknik diantaranya adalah :

**Morphology:** Dilation,Erotion, Opening, Closing, Hit and Miss Transform, Thinning, Thickening

**Edge Detection:** Edge, Line, dan Point Detection, Edge Detection berdasarkan turunanpertama, Operator Robert, Operator Sobel, Operator Prewitt, Operator Krisch, Deteksi Tepi Isotropik, Operator Canny, Edge Detection berdasarkan turunan kedua, Laplacian of Gaussian (LOG)

**Line Detection:**

**Image Segmentation:** Thresholding, Histogram, Local Adaptive Threshold, Otsu, Connected Component Labeling, Clustering, Hough Transformation, Template Matching

**Shape Analysis:** Chain Code, Topology Attribute, Moments, Fourier Descriptor, Medial Axis Transform

Setelah melalui pemisahan citra dan ekstraksi fitur dari suatu citra, biasanya proses dilanjutkan untuk proses pengenalan pola dimana proses tersebut adalah proses yang memiliki output deskripsi suatu citra, namun tidak menutup kemungkinan untuk menghentikan proses sampai pemisahan fitur sebagai output citra digital saja. Dalam proses pengenalan pola, ekstraksi fitur dapat disebut sebagai *preprocessing* dan pemisahan fitur disebut *low level image processing*.

Proses pengenalan pola sangat membutuhkan proses *feature extraction* ini karena kita tidak mungkin menggunakan metode-metode pengenalan pola dengan satu citra digital yang utuh dan mentah, walaupun memungkinkan hal tersebut dapat memakan memori yang sangat banyak dan sulit mengenali pola dari suatu citra, maka dari itulah *feature extraction* diperlukan.

*Feature extraction* memiliki beberapa metode atau teknik diantaranya Amplitudo, Histogram, Matriks Co-ocurrence, Gradient, Deteksi Tepi, Spektrum Fourier, Wavelet, Color based, Overlapping dan Nonoverlapping Block, Tapis Gabor, Fraktal

# Penutup

Modul praktikum ini belum sempurna. Dalam melaksanakan latihan dan tugas praktikum tidak mustahil para peserta mengalami kesulitan terutama dalam memahami apa yang diinginkan penulis dalam modul praktikum ini. Untuk itu perlu dilakukan revisi terus menerus baik dari sisi bahasa maupun struktur penulisan serta kejelasan dan ke-*uptodate*-an materi yang diinginkan. Walaupun demikian mudah-mudahan buku ini bermanfaat dan dapat dijadikan pegangan bagi yang berminat dalam bidang Pengolahan dan Analisis Citra Dijital.

Perlu ditekankan disini bahwa pengetahuan dan kemampuan praktis pemrograman komputer merupakan kompetensi dan *skill* yang wajib dimiliki oleh para mahasiswa maupun peminat dari modul praktikum ini. Mengingat juga perkembangan teknologi, maka pengetahuan pemrograman dalam konteks pemrograman *mobile* dan IoT merupakan hal yang cukup penting dalam konteks implementasi dari algoritma-algoritma pengolahan citra yang disajikan.

Bagi yang ingin memperdalam lebih jauh lagi tentunya dapat dibaca buku-buku bacaan yang ditampilkan pada daftar bacaan modul praktikum ini. Penyempurnaan akan terus dilakukan pada waktu yang akan datang; kritik dan saran penyempurnaan, penulis terima dengan senang hati untuk dipertimbangkan.

*Terimakasih sudah menggunakan modul praktikum ini.*

# Bahan Bacaan


(1) Rafael C Gonzalez dan Richard E Woods, *Digital Image Processing, 4th ed.* , Addison-Wesley, 2018

(2) Al Bovik, *The Essential Guide to Image Processing*, Elsevier, 2009

(3) William K Pratt, *Digital Image Processing*, Wiley, 2007

(4) John C Russ, *The Image Processing Handbook*, CRC Press, 2011

(5) Bernd Jähne, *Practical Handbook on Image Processing for Scientific and Technical Applications*, CRC Press, 2004

(6) Randy Crane, *A Simplified Approach to Image Processing*, Prentice Hall, 1997

(7) John Sharp, *Microsoft Visual Studio 2012 Step By Step*, Microsoft Press, 2012

(8) Rob Miles, *Learn The Kinect API, Microsoft Press*, 2012

(9) Md. Atiqur Rachman Ahad, *Computer Vision and Action Recognition*, Atlantic Press, 2012

(10) Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, *Digital Image Processing Using MATLAB*, McGraw Hill, 2013

(11) Bernd Girod, David Chen, Matt Yu, *Digital Image Processing using Android*, Stanford University, 2013.

# Lampiran

## A. Pengakses Piksel via Pointer

**Proses Perubahan Kecerahan (Brigtness) via Pointer**

```
1  private void Brightness_PTR_Click(object sender, EventArgs e){
2      BitmapData data = bitmap.LockBits(new Rectangle(0,0,bitmap.Width,
           bitmap.Height),ImageLockMode.ReadWrite,PixelFormat.
           Format24bppRgb);
3      int nOffset = data.Stride - data.Width * 3, nVal, nBrightness = 50;
4      int nWidth = data.Width * 3;
5      unsafe{
6          byte* ptr = (byte*)(data.Scan0);
7          for (int y = 0; y < data.Height; ++y){
8              for (int x = 0; x < nWidth; ++x){
9                  Val = (int)(ptr[0] + nBrightness);
10                 if (nVal < 0) nVal = 0;
11                 if (nVal > 255) nVal = 255;
12                 ptr[0] = (byte)nVal;
13                 ++ptr;}
14             ptr += nOffset;}}
15     bitmap.UnlockBits(data);
16     pictureBox1.Image = bitmap;}
```

**Konversi Citra RGB Ke Citra Grey via Pointer**

```
1  private void Konversi2GreyViaPointer(Bitmap bmp){
2      BitmapData bmData = bmp.LockBits(new Rectangle(0,0,bmp.Width, bmp.
           Height),ImageLockMode.ReadWrite,PixelFormat.Format24bppRgb);
3      unsafe{
4          byte* p = (byte*)(void*)bmData.Scan0.ToPointer();
5          int stopAddress = (int)p + bmData.Stride * bmData.Height;
6          while ((int)p != stopAddress){
7              p[0] = (byte)(.299 * p[2] + .587 * p[1] + .114 * p[0]);
8              p[1] = p[0];
9              p[2] = p[0];
10             p += 3;}}
11     bmp.UnlockBits(bmData);}
```

Jangan lupa mengaktifkan *unsafe processing mode* pada *project properties*

# B. Pemrosesan Video Secara *Realtime*

1. **Install AForge.NET**

   a. Jalankan *VS 2010*, buat *Project* baru

   b. Fokus ke `Solution Explorer`:

   c. Pilih `References`, tekan tombol kanan, `Add Reference`

   d. *Browse* ke tempat dimana `AForge` diinstall, pilih folder `Release`

   e. Pilih DLL : `AForge.Video.DLL`, lalu `AForge.Video.DirectShow.DLL`

   f. Pada `Solution Explorer` akan tampil kedua DLL tersebut

   g. Fokus ke `View Code` untuk *Form* yang kita buat

   h. Tambahkan `using Aforge.Video;` dan `Aforge.Video.DirectShow;` pada *header* program

2. **Persiapan Komponen dan Cek Kamera**

   a. Fokus ke Form Designer

   - Tambahkan komponen PictureBox
   - Tambahkan variabel-variabel berikut (posisikan diatas `public Form1()`,

     ```
     FilterInfoCollection webCams;
     VideoCaptureDevice kamera;
     Bitmap frame;
     ```

   b. Mengecek keberadaan kamera

   - Tambahkan Button ke Form, isi dengan kode berikut:

     ```
     webCams = new  FilterInfoCollection(FilterCategory.VideoInputDevice);
     foreach (FilterInfo camera in webCams)
     MessageBox.Show(camera.Name);
     ```

     *Jika kamera tersedia akan muncul kotak pesan dan nama kameranya*

3. **Koneksi Ke Kamera**

   a. Buat Button koneksi ke kamera dan isi dengan kode berikut:

   ```
   kamera = new VideoCaptureDevice(webCams[4].MonikerString);
   kamera.NewFrame += new NewFrameEventHandler(kamera_ProsesFrame);
   kamera.Start();
   ```

   [4] *menunjukkan nomor urut pada saat mengecek kamera. Ganti/sesuaikan dengan hasil pada komputer anda*

   b. Buat fungsi Pemrosesan Frame `kamera_ProsesFrame`

   ```
   void kamera_ProsesFrame(object sender, NewFrameEventArgs eventArgs)
   {
     frame= (Bitmap)eventArgs.Frame.Clone();
     pictureBox1.Image = frame;
   }
   ```

c. Run program anda

4. Perhatikan hal-hal berikut:

   a. Set *project properties* ke AnyCPU

   b. Set `SizeMode` untuk `pictureBox` ke `StretchImage`

   c. Kelas `kamera_ProsesFrame` berisi informasi setiap *frame*, sehingga kita bisa memproses *frame* tersebut sesuai dengan kebutuhan kita.

   d. Agar pemrosesan *frame* berjalan dengan cepat, maka harus melakukan setup sbb:

      - `allow unsafe code` pada *project properties*
      - Tambahkan `using System.Drawing.Imaging;` pada *header* program

5. Contoh Pemrosesan *Frame*

```
void kamera_ProsesFrame(object sender, NewFrameEventArgs eventArgs)
{
   frame=(Bitmap)eventArgs.Frame.Clone();
   int r, g, b;
   BitmapData data = frame.LockBits(new Rectangle(0, 0, frame.Width,
      frame.Height), ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);
   unsafe
   {
      byte* ptr = (byte*)(data.Scan0);
      for (int i = 0; i < frame.Height; i++)
      {
        for (int j = 0; j < frame.Width; j++)
        {
           b = ptr[0]; g = ptr[1]; r = ptr[2];
           ptr[0] = ptr[1] = ptr[2] =
                (byte)(.299 * r + .587 * g + .114 * b);
           if (ptr[0] >= 128) ptr[0] = ptr[1] = ptr[2] = 255;
           else ptr[0] = ptr[1] = ptr[2] = 0;
           ptr += 3;
         }
         ptr += data.Stride - data.Width * 3;
      }
}
  frame.UnlockBits(data);
pictureBox1.Image = frame;
}
```

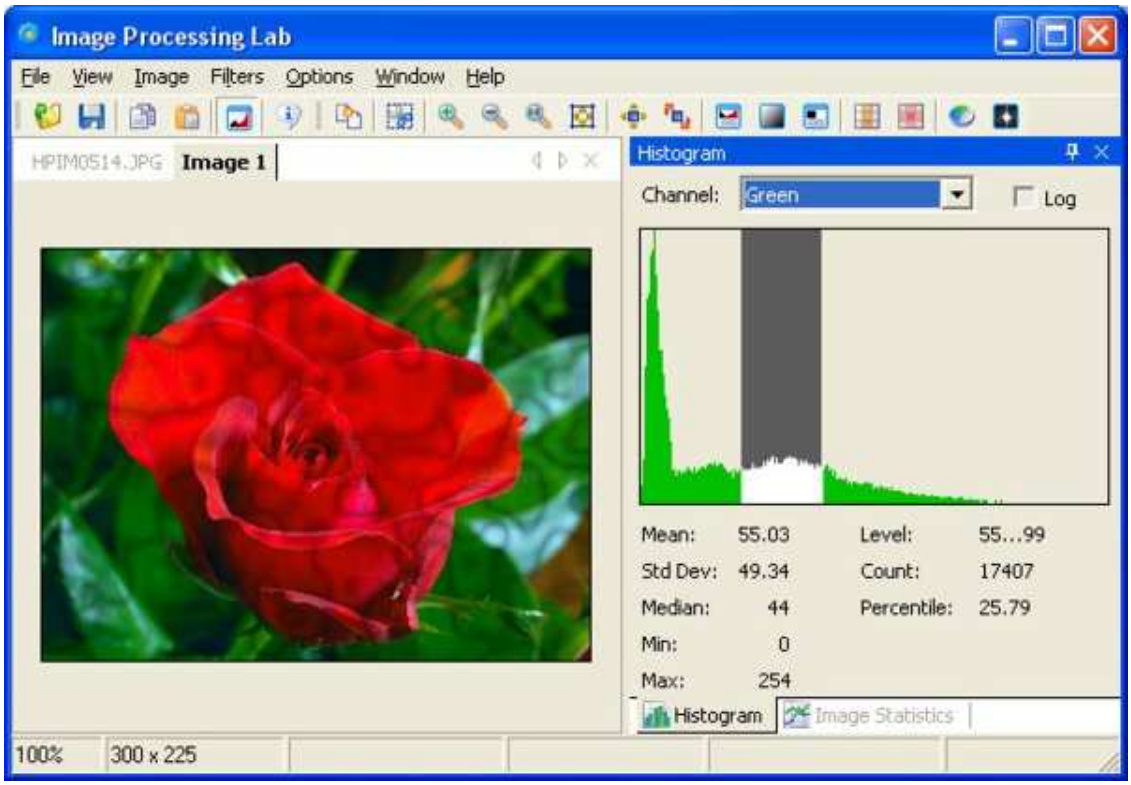
Program lengkap dapat diunduh di `http://setiawanhadi.unpad.ac.id` kemudian pilihkan menu `Download`

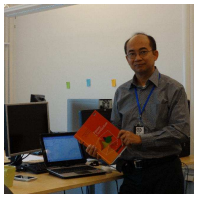
# C. Pengenalan *Image Processing Library*

Program-program untuk mengolah citra dijital banyak dijumpai di pasaran, misalnya yang terkenal adalah Adobe Photoshop. Selain perangkat lunak yang sudah jadi EXECUTABLE tersebut, dewasa ini dijumpai juga *library* khusus untuk melakukan proses pembuatan program yang berkaitan dengan pengolahan citra dijital. *Library* tersebut ada yang berupa DLL, adapula yang diberikan dalam bentuk *Source Program*.

Keberadaan *library image processing* ini sangatlah mempermudah pembuatan program. Disatu sisi hal ini sangatlah membantu karena kita tinggal menerapkannya pada program komputer kita dan tidak perlu memikirkan hal-hal yang mendasar. Disisi lain, khususnya bagi mahasiswa, konsep dasar citra dijital menjadi kurang sehingga perlu usaha keras untuk memahami teori dasarnya.

Program program yang berupa DLL dan gratis antara lain adalah OpenCV, EmguCV, AForge.NET dan sebagainya. Sedangkan program-program yang diberikan *source*nya dapat diperoleh melalui akses internet. Salah satu yang lengkap adalah Image Processing Lab in C# yang dibuat oleh Andrew Kirilov.





Setiawan Hadi adalah dosen tetap Departemen Ilmu Komputer dan mengajar pada program studi sarjana Teknik Informatika (sebelumnya di jurusan Matematika) Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran sejak tahun 1993 dengan pangkat/golongan Lektor Kepala (2007)/IV.a (2009).

Ia menyelesaikan studi doktoral informatika (S3) di STEI ITB pada Juli 2008, memiliki gelar Master of Science in Computer Science (M.Sc.CS.) dari FCS UNB Canada yang diperoleh pada Oktober 1996 dan menyelesaikan program Sarjana Matematika di FMIPA UNPAD pada Agustus 1991. Selain itu dia menyandang ijazah kediplomaan, D-1 Pemrogram Komputer (September 1981) dan D-3 Sistem Analis (November 1983), yang diperolehnya dari Program PAT-JPK ITB , yang diikutinya sejak selesai SMA Trinitas Bandung tahun 1980.

Sebagai pengajar, Mata kuliah yang diampu adalah Pengolahan Citra, Grafika Komputer, Visi Komputer, dan Metode Penelitian Informatika. Selain itu dia memiliki pengalaman mengajar mata kuliah Interaksi Manusia dan Komputer, Pemrograman Non Prosedural (Prolog), Pemrograman Java, Sistem Tersebar, Sistem Operasi, Algoritma & Pemrograman, Mobile Computing, dan Kecerdasan Buatan, pada tingkatan Diploma III, Sarjana (S1) dan Pascasarjana (S3) di Universitas Padjadjaran. Sebagai pembimbing dan penguji tugas akhir, SH telah membimbing dan menguji banyak mahasiswa program Diploma (MI, TI, TK) dan Sarjana (Matematika dan Informatika). Pada program pascasarjana, Setiawan telah membimbing 1 mahasiswa dari Unpad dan menguji 4 orang mahasiswa doktoral dari ITB, Unpad, dan Gunadarma.

Sebagai peneliti pada RAID Laboratory (Divisi *Computer Vision Laboratory*), topik riset yang diminati dan digeluti sejak menyelesaikan studi S2 adalah grafika komputer, pengolahan citra dan visi komputer. Jenis hibah penelitian yang pernah diperoleh adalah Penelitian Hibah Pasca ITB tahun 2005 dan 2006 serta penelitian Hibah Kompetisi (PHK) A2 tahun 2005. Pada tahun 2011 dan 2012 mendapatkan penelitian Fundamental DIKTI. Pada tahun 2013 mendapat hibah Penelitian Unggulan Perguruan Tinggi (RUPT, a.k.a Hibah Bersaing DIKTI). Sejak September 2013 sampai dengan November 2013 telah berada di University of Skövde Swedia dalam rangka program SAME Dikti Kemdikbud. Tahun 2014 terpilih menerima beasiswa NANUM dan berpartisipasi dalam ICM 2014 di Korea. Tahun 2015 terlibat dalam penelitian PUPT (sebagai ketua), PIC dalam Penelitian Kolaborasi Internasional STIC ASIE dengan Topik AMADI, Anggota Tim ALG FMIPA dan ALG FPSI, serta Ketua Pelaksana HUPS TIF Universitas Padjadjaran. Tahun 2016, Setiawan menjadi ketua peneliti Riset Kompetitif Nasional KLN, menjadi anggota penelitian ALG dengan Prof. A.K. Supriatna, dan pernah menjadi anggota penelitian PUPT Psikologi dengan Prof. Wilis Srisayekti. Bulan Februari 2016 Setiawan ditugaskan menjadi Kepala Departemen Ilmu Komputer FMIPA Universitas Padjadjaran. Pada tahun 2016 dan 2018, Setiawan melakukan *research visit* ke University of La Rochelle dalam rangka Proyek AMADI dan Program SAME.

Setiawan telah mengikuti berbagai seminar nasional dan internasional, serta telah mempublikasikan lebih dari 50 publikasi dan karya ilmiah. Setiawan adalah anggota dari himpunan profesi IndoMS, IEEE Computer, Aptikom, IPKIN, IAEng, dan ACM, pemegang Sertifikasi MCE, dan anggota Laboratorium RAID. Selain itu, Setiawan juga menjadi asesor BAN-PT, LPDP, Serdos, Usulan Prodi Baru, dan reviewer beberapa Jurnal Nasional dan Internasional.