

Baik  berikut penjelasan **baris demi baris** untuk program `ImageDataProcessor.cs`, lengkap dengan fungsi, logika, dan maknanya.

Kelas ini merupakan bagian dari logika **MiniPhotoshop**, dan digunakan untuk **mengonversi gambar menjadi data numerik (array 3D)** dan sebaliknya — yaitu **membangun kembali gambar dari data array**.

Bagian 1: Import Namespace

```
using System.Drawing.Drawing2D;
```

- Menyediakan kelas-kelas untuk manipulasi gambar 2D seperti transformasi dan penskalaan.
 - Walau tidak digunakan langsung dalam fungsi ini, tetap disertakan karena file ini berhubungan dengan pemrosesan gambar 2D.
-

Bagian 2: Deklarasi Namespace dan Kelas

```
namespace MiniPhotoshop.Logic.Helpers
{
    public static class ImageDataProcessor
```

- namespace `MiniPhotoshop.Logic.Helpers` → Menunjukkan bahwa kelas ini berfungsi sebagai *helper logic* untuk memproses data gambar di proyek `MiniPhotoshop`.
 - `public static class ImageDataProcessor` → Kelas statis, artinya:
 - Tidak perlu dibuat objek untuk memanggilnya.
 - Semua metode di dalamnya bersifat statis.
 - Berfungsi seperti “alat bantu” (utility) untuk mengubah format data gambar.
-

Bagian 3: Fungsi Pertama — `LoadTo3DArray()`

Tujuan:

Mengubah objek `Bitmap` menjadi array 3 dimensi (`int[, ,]`) yang berisi nilai **RGB** dan **Grayscale** dari setiap piksel.

Kode:

```
public static int[, ,] LoadTo3DArray(Bitmap bmp)
```

- Parameter:

- o `bmp` → gambar bertipe `Bitmap` yang akan dikonversi ke bentuk numerik.
 - Return:
 - o `int[, ,]` → array tiga dimensi [`x`, `y`, `channel`].
-

Mendapatkan Ukuran Gambar

```
int width = bmp.Width;
int height = bmp.Height;
int[, , ] imageData3D = new int[width, height, 4];
```

- `width` dan `height` → lebar dan tinggi gambar.
 - Membuat array 3D dengan dimensi [`lebar`, `tinggi`, `4`], di mana:
 - o Indeks ke-0 = nilai **Red**
 - o Indeks ke-1 = nilai **Green**
 - o Indeks ke-2 = nilai **Blue**
 - o Indeks ke-3 = nilai **Grayscale**
-

Looping Piksel Gambar

```
for (int y = 0; y < height; y++)
{
    for (int x = 0; x < width; x++)
```

- Melakukan iterasi pada setiap baris (`y`) dan kolom (`x`) piksel dalam gambar.
-

Mengambil Warna Piksel

```
Color pixelColor = bmp.GetPixel(x, y);
int r = pixelColor.R;
int g = pixelColor.G;
int b = pixelColor.B;
```

- `GetPixel(x, y)` mengambil warna pada posisi (`x`, `y`).
 - `pixelColor.R`, `.G`, dan `.B` masing-masing mengembalikan nilai 0–255 untuk kanal **merah, hijau, dan biru**.
-

Menghitung Nilai Grayscale

```
int gray = (int)(0.3 * r + 0.59 * g + 0.11 * b);
```

- Rumus konversi warna ke grayscale berdasarkan **persepsi manusia terhadap intensitas warna**:
 - 30% merah + 59% hijau + 11% biru.
 - Dikonversi ke `int` agar sesuai dengan tipe data array.
-

Menyimpan Nilai ke Dalam Array

```
imageData3D[x, y, 0] = r;  
imageData3D[x, y, 1] = g;  
imageData3D[x, y, 2] = b;  
imageData3D[x, y, 3] = gray;
```

- Setiap piksel disimpan ke empat kanal array 3D sesuai urutan warna dan grayscale.
-

Mengembalikan Hasil

```
return imageData3D;
```

- Mengembalikan array 3D hasil konversi dari gambar `Bitmap`.
-

Bagian 4: Fungsi Kedua — `CreateBitmapFrom3DArray()`

Tujuan:

Mengubah kembali data array 3D menjadi gambar `Bitmap`, dengan opsi menampilkan kanal tertentu saja.

Kode:

```
public static Bitmap CreateBitmapFrom3DArray(int[,] imageData3D, int channel  
= -1)
```

- Parameter:
 - `imageData3D` → data hasil `LoadTo3DArray()`.
 - `channel` (opsional, default = -1):
 - 0 = hanya Red
 - 1 = hanya Green
 - 2 = hanya Blue

- 3 = Grayscale
 - -1 = Full color (RGB lengkap)
-

Validasi Awal

```
if (imageData3D == null)
{
    return null;
}
```

- Mengecek apakah array kosong (belum diisi). Jika ya, fungsi keluar dengan nilai `null`.
-



Menentukan Ukuran Bitmap

```
int width = imageData3D.GetLength(0);
int height = imageData3D.GetLength(1);
Bitmap bmp = new Bitmap(width, height);
```

- `GetLength(0)` dan `GetLength(1)` mengambil ukuran array pada sumbu X dan Y.
 - Membuat objek `Bitmap` baru sesuai ukurannya.
-



Loop untuk Mengisi Piksel Gambar

```
for (int y = 0; y < height; y++)
{
    for (int x = 0; x < width; x++)
```

- Iterasi setiap piksel di dalam array 3D.
-



Mengambil Nilai RGB & Grayscale

```
int r = imageData3D[x, y, 0];
int g = imageData3D[x, y, 1];
int b = imageData3D[x, y, 2];
int gray = imageData3D[x, y, 3];
```

- Mengambil kembali nilai warna dari tiap kanal di array 3D.
-

Menentukan Warna Berdasarkan Kanal

```
Color color;
if (channel == 0)
{ // hanya red
    color = Color.FromArgb(r, 0, 0);
}
else if (channel == 1)
{ // hanya green
    color = Color.FromArgb(0, g, 0);
}
else if (channel == 2)
{ // hanya blue
    color = Color.FromArgb(0, 0, b);
}
else if (channel == 3)
{ // grayscale
    color = Color.FromArgb(gray, gray, gray);
}
else
{ // tampilkan full color
    color = Color.FromArgb(r, g, b);
}
```

- Berdasarkan nilai `channel`, fungsi membuat objek `Color`:
 - Hanya menampilkan satu kanal warna tertentu, atau
 - Menggabungkan semua kanal RGB untuk gambar penuh.
-

Menetapkan Warna ke Bitmap

```
bmp.SetPixel(x, y, color);
```

- Menetapkan warna hasil ke piksel `(x, y)` pada gambar `Bitmap` baru.
-

Mengembalikan Gambar Akhir

```
return bmp;
```

- Mengembalikan `Bitmap` hasil konversi dari array numerik kembali menjadi gambar visual.
-

Kesimpulan Umum

Kelas `ImageDataProcessor` digunakan untuk **konversi dua arah** antara gambar (`Bitmap`) dan data numerik (`int[, ,]`), yang penting untuk operasi pengolahan citra digital.

Fungsi	Arah Konversi	Penjelasan
<code>LoadTo3DArray(Bitmap bmp)</code>	<code>Gambar → Array 3D</code>	Mengurai setiap piksel menjadi nilai RGB + Grayscale.
<code>CreateBitmapFrom3DArray(int[,] data, int channel)</code>	<code>Array 3D → Gambar</code>	Membangun ulang gambar berdasarkan kanal warna tertentu atau seluruhnya.

Dengan dua fungsi ini, MiniPhotoshop bisa melakukan operasi seperti:

- Filter citra (blur, edge detection, dll),
 - Analisis warna,
 - Pemrosesan grayscale,
karena gambar diubah menjadi data numerik yang mudah dimanipulasi dengan algoritma.
-