Bild abdunkeln wenn es zu hell ist

Ja, das geht ganz einfach mit reinem .NET-Framework und C#. Du brauchst im Wesentlichen zwei Schritte:

1. **Mittlere Helligkeit messen**
2. **Bild um einen errechneten Faktor abdunkeln**

---

1. Mittlere Helligkeit messen

Wir laden das JPG in ein `Bitmap`, iterieren über alle Pixel (oder eine Unter-Stichprobe, um es schneller zu machen) und berechnen die durchschnittliche Luminanz mit der üblichen Formel

Y=0,299 R+0,587 G+0,114 BY = 0{,}299\,R + 0{,}587\,G + 0{,}114\,B

```
private static float GetAverageBrightness(Bitmap bmp)
{
    long sum = 0;
    int width = bmp.Width;
    int height = bmp.Height;
    int total = width * height;

    // Optional: für sehr große Bilder nur jedes N-te Pixel prüfen
    int step = (total > 1_000_000) ? 10 : 1;

    for (int y = 0; y < height; y += step)
    for (int x = 0; x < width;  x += step)
    {
        Color c = bmp.GetPixel(x, y);
        // Luminanz-Berechnung
        int lum = (int)(0.299 * c.R + 0.587 * c.G + 0.114 * c.B);
        sum += lum;
    }
```

```
    // Durch Schritt-Vergrößerung wieder korrigieren
    int count = (width / step) * (height / step);
    return sum / (float)count / 255f;  // Normiert auf [0,1]
}
```

Ruft ihr `GetAverageBrightness(yourBitmap)` auf, erhaltet ihr eine Zahl zwischen etwa 0 (komplett schwarz) und 1 (komplett weiß). Werte über ~0.6–0.7 sind in der Regel zu hell für Passfotos.

---

2. Bild abdunkeln

Wenn der gemessene Wert bb zu hoch ist, könnt ihr das Bild um einen Faktor (target/b)(\mathrm{target} / b) abdunkeln. Am einfachsten geht das mit einem **ColorMatrix** und `Graphics.DrawImage`:

```
private static Bitmap AdjustBrightness(Bitmap original, float brightnessFactor)
{
    // ColorMatrix: Diagonale bleibt bei 1, und das Offset in der letzten Spalte skaliert RGB.
    var cm = new ColorMatrix(new float[][]
    {
        new float[] { brightnessFactor, 0,              0,              0, 0 },
        new float[] { 0,              brightnessFactor, 0,              0, 0 },
        new float[] { 0,              0,              brightnessFactor, 0, 0 },
        new float[] { 0,              0,              0,              1, 0 },
        new float[] { 0,              0,              0,              0, 1 }
    });

    var ret = new Bitmap(original.Width, original.Height);
    using (Graphics g = Graphics.FromImage(ret))
    using (var attrs = new ImageAttributes())
    {
        attrs.SetColorMatrix(cm);
        g.DrawImage(
            original,
            new Rectangle(0, 0, ret.Width, ret.Height),
            0, 0, original.Width, original.Height,
            GraphicsUnit.Pixel,
            attrs);
```

```
    }
    return ret;
}
```

---

3. Gesamtroutine

Kombinieren wir beides in einer Methode:

```
public static Bitmap AutoDarken(Bitmap bmp, float targetBrightness = 0.5f)
{
    // 1. aktuelle Helligkeit
    float current = GetAverageBrightness(bmp);

    // 2. gewünschten Zielwert nicht überschreiten
    if (current <= targetBrightness)
        return bmp;  // ist dunkel genug

    // 3. Faktor berechnen und anwenden
    float factor = targetBrightness / current;
    return AdjustBrightness(bmp, factor);
}
```

Beispiel-Aufruf

```
var img = (Bitmap)Image.FromFile("passfoto.jpg");
var darkened = AutoDarken(img, 0.5f);
darkened.Save("passfoto_dunkler.jpg", ImageFormat.Jpeg);
```

- `targetBrightness` ist hier auf 0.5 gesetzt (also mittleres Grau). Je kleiner der Wert, desto dunkler das Ergebnis.
- Siehst du, dass manche Bilder immer noch zu hell sind, kannst du den Default-Wert weiter absenken (z. B. 0.45).

Performance-Tipp

- Für große Bilder lohnt sich `Bitmap.LockBits` + Pointer-Zugriff statt `GetPixel`/`SetPixel`.

- Du könntest auch nur jeden 5. oder 10. Pixel zur Helligkeitsmessung heranziehen (wie im Beispiel), das reicht für eine grobe Schätzung meistens aus.

Mit dieser Methode kannst du automatisiert durch einen ganzen Ordner von Passfotos iterieren und alle zu hellen Bilder automatisch abdunkeln, bevor du sie auf die Plastikkarte druckst.

Revision #1
Created 20 June 2025 10:37:40 by Stefan Mechler
Updated 20 June 2025 10:38:15 by Stefan Mechler