Object Detect MA Dökümanına Hoşgeldiniz.

Bilgiler aşağıda yer almaktadır.

|  |
| --- |
| Object Detect MA V1.0 |
| Resim İşleme Kütüphanesi |
|  |
| Görüntü işleme üzerine geliştiğim kütüphanemin ilk versiyonudur. |
|  |

Object Detect MA V1.0

Resim İşleme Kütüphanesi

MUHAMMED KANDEMİR

• • •

Öncelikle kütüphanemi indirdiğiniz için çok teşekkür ederim. Yazılım MUHAMMED KANDEMİR’e aittir. <https://www.facebook.com/muhammed.kandemir.54>” adresinden ulaşabilirsiniz. Youtube kanalım <https://www.youtube.com/channel/UClrBPXVSCiyPAZIV3hCtLNw>.

## Nedir?

Resim işleme konusunda kolaylıklar sunan bir kütüphanedir. Belli geliştirilmiş algoritmalar üzerinden size hızlı ve rahat bir kodlama sunmaktadır.

## ODColorMA

Bu renk sınıftır. İçerik olarak;

public class ODColorMA

{

public ODColorMA(byte r, byte g, byte b);

public byte B { get; set; }

public byte G { get; set; }

public byte R { get; set; }

public static implicit operator ODColorMA(Color color);

}

Şeklindedir. System.Drawing.Color sınıfına implicit(Çevirme) desteği sunulmaktadır. Bu sayede elinizdeki Color sınıfı değerindeki bir değişkeni rahatlıkla ODColorMA türüne taşıyabilirsiniz.

ODColorMA sınıfı bizim kütüphanemizde renk işlemleri için kullanılacaktır.

## ODColorDiffMA

Bu renklerin oranlarını saklayan sınıftır. İçerisinde RGB(Red – Kırmızı, Green – Yeşil, Blue – Mavi) değerlerinin oranlarını saklar. İçerik olarak;

public class ODColorDiffMA

{

public ODColorDiffMA(int red, int green, int blue);

public float GreenDivideBlue { get; set; }

public float RedDivideBlue { get; set; }

public float RedDivideGreen { get; set; }

public static void Calc(float red, float green, float blue, out float redDivideGreen, out float redDivideBlue, out float greenDivideBlue);

public bool Control(int red, int green, int blue, float diff);

public static implicit operator ODColorDiffMA(Color color);

public static implicit operator ODColorDiffMA(ODColorMA color);

}

Şeklindedir. System.Drawing.Color sınıfına ve ODColorMA sınıfına implicit(Çevirme) desteği bulunmaktadır. Bu sınıfın amacı bize döngü sırasında işlemlerin sürekli bir şekilde hesaplanması yerine anlık olarak hesaplamasıdır. Manuel olarak değişiklikler yapabilmekteyiz fakat tavsiye etmiyorum.

## ODRectangleMA

Rectangle(Dikdörtgen) sınıfıdır. “.NET” ten farklı olarak X2 ve Y2 değerleri bulundurur. Bunlarda ikinci koordinatlar olarak geçmektedir. İçerik olarak;

public class ODRectangleMA

{

public ODRectangleMA(float x, float y, float width, float height);

public int CountDetect { get; set; }

public float Height { get; set; }

public float Width { get; set; }

public float X { get; set; }

public float X2 { get; set; }

public float Y { get; set; }

public float Y2 { get; set; }

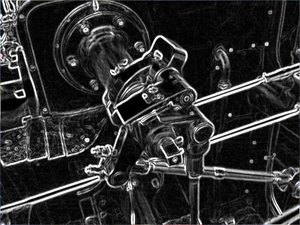
public static implicit operator RectangleF(ODRectangleMA value);

}

Sınıfın System.Drawing.RectangleF implicit özelliği bulunmaktadır. Bu sayede sınıfınızı RectangleF türüne rahatlıkla cast(Dönüştürmek) edebilirsiniz.

## ODSobelEdgeMA

Sobel algoritması ve renk filtreleme özelliği ile nesneleri algılama sınıfımız. Sobel bir kenar algılama algoritmasıdır. Resimlerde kenar geçişleri için kullanılır.



Üstteki resim örnektir. Renk filtrelemeyi ise ODColorDiffMA üzerinden yapmaktadır. İçerik olarak;

public abstract class ODSobelEdgeMA

{

protected ODSobelEdgeMA();

public static List<ODRectangleMA> Detect(Bitmap bmp, DetectEvent sobelEdgeDetectEvent, DetectEvent rectDetectEvent, DetectEvent otherPixelDetectEvent, FilterRect filterRectEvent, List<ODColorDiffMA> diffDatas, float sensivity = 0.1F, byte boxRangeWidthPercent = 4, byte boxRangeHeightPercent = 4, byte detectRangeWidthPercent = 10, byte detectRangeHeightPercent = 10);

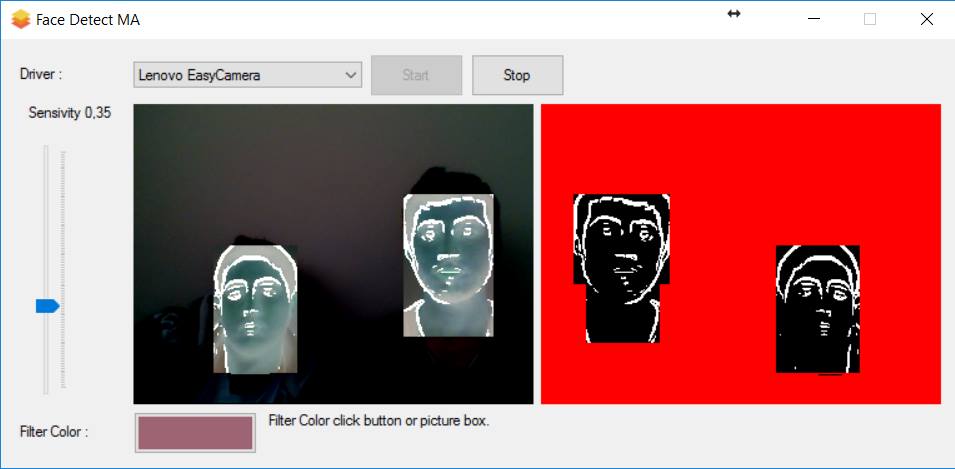
public delegate ODColorMA DetectEvent(int x, int y, ODColorMA color);

public delegate List<ODRectangleMA> FilterRect(List<ODRectangleMA> rects);

}

Biraz karışık gelmiş olabilir. Buradaki Detect fonksiyonu bize analiz işlemi yapmakta. Diğer delegate ler ise sadece event olmak üzere. Şimdi ise Detect fonksiyonunun parametrelerini inceleyelim.

Dönüş değeri “List<ODRectangleMA>” : Bu size resim sırasında algılanmış renk filtreleri üzerinden dikdörtgenleri verir. Örnek :



Örnekte görülen iki dikdörtgen yüz mevcut. Bizim sınıfımız bu değerleri döndürmektedir. Bu sayede resimde nesnelerin alanlarını bulmuş oluyorsunuz.

Parametre “Bitmap bmp” : Bu bizim işlenmesini istediğimiz resim. Yukarıdaki örnekler işlendikten sonraki halleri. Burda orijinal resim kamera resmi olmaktadır. Gönderilen değer üzerinde sonraki parametreler olan eventler üzerinden değişiklik yapabilirsiniz örnekteki gibi.

Parametre “DetectEvent sobelEdgeDetectEvent” : Buradaki event;

public delegate ODColorMA DetectEvent(int x, int y, ODColorMA color); şeklindedir.

Eğer resimde kenar algılanırsa size o koordinat ve renk değeri dönücektir. Burada sizden bir dönüş değeri beklemektedir. Bu değeri null döndürdüğünüz taktirde herhangi bir işlem yapmamaktadır. Herhangi bir değer döndürüldüğünde sobel kenarı bulduğu pixelde renk değiştirmenizi sağlar. Yukarıdaki örnekte beyaz döndürüldü.

Parametre “DetectEvent rectDetectEvent” : Buradaki event yukarıdaki ile aynıdır. Aynı kurallar çerçevesinde null döndürürseniz işlem yapmaz. Renk döndürürseniz ise nesnenin bulunduğu kare içerisinde ve sobel kenarları dışındaki pixellerde renk değişimi olanağı sunar. Yukarıdaki örnekte renk değerleri tersine çevirilmiştir.

Parametre “DetectEvent otherPixelDetectEvent” : Buradaki event yukarıdaki ile aynıdır. Aynı kurallar çerçevesinde null döndürürseniz işlem yapmaz. Renk döndürürseniz sobel ve bulunan nesneler dışındaki pixellerde renk değişimi olanağı sunar.

Parametre “FilterRect filterRectEvent” : Buradaki event;

public delegate List<ODRectangleMA> FilterRect(List<ODRectangleMA> rects); şeklindedir.

Size resim içerisinde algılanan dikdörtgenleri döndürür ve filtre yapma olanağı sunar. ODRectangleMA sınıfındaki koordinatlardan alan olarak veya CountDetect değişkenindeki algılanan pixel sayısından yola çıkarak ekrandaki boş kareleri temizleyebilirsiniz. Her görüntü işlemede hata payı olduğu gibi bu kütüphanede istemsizce size hatalı kareler döndürecektir. Bunun filtresini bu iki yoldan yapabilirsiniz. Sonucunda yeni listeyi dönerek filtreleme işlemini tamamlamış olucaksınız.

Parametre “List<ODColorDiffMA> diffDatas” : Bu değer sizden ekranda filtreleme yapılacak olan renklerin olduğu kısım. Buraya gönderdiğiniz değerler üzerinden filtreleme yapılacaktır. İnsan yüzü filtresi istiyorsanız örnek olarak 5 – 10 insan yüzündeki renk tonlarını buraya yazabilirsiniz.

Parametre “float sensivity = 0.1F” : Bu değer hassaslık değeridir. İşlemler sırasında ışığa göre değişebilecek olan renk tonlarını buradaki hassaslık ile dengeleyebilirsiniz.

Parametre “byte boxRangeWidthPercent = 4” : Bu değer size kutu içerisinde filtre yapma alanını sormaktadır. Burda yüzdelik olarak girdiğiniz değer yatay düzlemde filtreleme yapıcağınız alanın boş bırakılcak yüzdesidir.



Yukarıdaki örnekteki gibi gri alanlar taranmayacak olan kısımlardır.

Parametre “byte boxRangeHeightPercent = 4” : Bu değer size kutu içerisinde filtre yapma alanını sormaktadır. Burda yüzdelik olarak girdiğiniz değer dikey düzlemde filtreleme yapıcağınız alanın boş bırakılcak yüzdesidir.



Yukarıdaki örnekteki gibi gri alanlar taranmayacak olan kısımlardır.

Parametre “byte detectRangeWidthPercent = 10” : Bu değer size iki algılanan nesne karesi arasında maksimum uzaklığın yatay cinsinden yüzdelik olarak uzunluğunu istiyor. Burada yazılan değer ne kadar büyürse nesne sayısı o kadar azalır.

Bunu şöyle düşünün iki yüz arasında 1 karış olursa 10, 2 karış mesafe olursa 20 değerlerine sahip olur.

Parametre “byte detectRangeHeightPercent = 10” : Bu değer size iki algılanan nesne karesi arasında maksimum uzaklığın dikey cinsinden yüzdelik olarak uzunluğunu istiyor. Burada yazılan değer ne kadar büyürse nesne sayısı o kadar azalır.

Bunu şöyle düşünün iki yüz arasında 1 karış olursa 10, 2 karış mesafe olursa 20 değerlerine sahip olur.

# MUHAMMED KANDEMİR

## <https://www.facebook.com/groups/gameprogramerhacker/>

## https://github.com/mav1995MK