در بخش اصلی یکسری بررسی ذاریم که فقط چک کنیم ادرس ها و فلذر های که میخواهیم در ان ذخیه کنیم وجود دارند یا نه و اگر وجود داشتن انهارو رو خالی کنیم و اگر نداشتن درست کنیم.

if (!System.IO.Directory.Exists(@"D:\university\AD entezari\ADproj\Create"))

System.IO.Directory.CreateDirectory(@"D:\university\AD entezari\ADproj\Create");

Console.WriteLine("Enter number of horizontal pixels that should be removed : ");

int horizontalremove = int.Parse(Console.ReadLine());

Console.WriteLine("Enter number of vertical pixels that should be removed : ");

int verticalremove = int.Parse(Console.ReadLine());

Console.WriteLine("Enter your image name : ");

string name = Console.ReadLine();

if (System.IO.Directory.Exists(@"D:\university\AD entezari\ADproj\Create\" + name))

System.IO.Directory.Delete(@"D:\university\AD entezari\ADproj\Create\" + name, true);

System.IO.Directory.CreateDirectory(@"D:\university\AD entezari\ADproj\Create\" + name);

در مرحله اول به وسیله کلاس bitmap و دادن ادرس به عنوان ورودی به این کلاس میتوانیم عکس داده شده را در ابجکتی از این کلاس ذخیره کنیم.

غیر از کلاس اصلی خود،2 کلاس دیگر به نام های horizontal و vertical داریم.

هر دو کلاس تقریبا کد های یکسانی دارند با این تفاوت که در یکی درز های عمودی را پیدا میکنیم و در دیگری درز های افقی.

کلاس horizontal به عنوان ورودی تعداد درز های افقی که میخواهیم حذف کنیم،ابجکتی از کلاس بیت مپ و اسم عکسی که داریم پردازش میکنیم رو میدهیم.کد base ای که زده شدی برای پیدا کردن درز ها در جعت عمودی هست برای همین وقتی میخواهیم درز هارو در جهت افقی پیدا کنیم،عکس رو 90 درجه rotate میدهیم.

کلاسی به اسم پیکسل داریم که دارای پراپرتی father,r,g,b,energy هست که هر پیکسل یک ابجکت از این کلاس را دارد که به صورت pixel[x,y] ذخیره میکنیم.

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

Color pixelColor = myBitmap.GetPixel(i, j);

pixel[i, j] = new Pixel

{

r = int.Parse(pixelColor.R.ToString()),

g = int.Parse(pixelColor.G.ToString()),

b = int.Parse(pixelColor.B.ToString())

};

}

}

در بخش بالا به وسیله GetPixel رنگ هر پیکسل رو در کلاس پیکسلی که درست کردیم نگه میداریم.

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height - 1; j++)

{

if ((i == 0 || i == width - 1) && (j == 0 || j == height - 1))

{

energy = 0;

}

else

{

if (i == 0 || i == width - 1)

{

energy = Math.Pow(pixel[i, j - 1].r - pixel[i, j + 1].r, 2) +

Math.Pow(pixel[i, j - 1].g - pixel[i, j + 1].g, 2) +

Math.Pow(pixel[i, j - 1].b - pixel[i, j + 1].b, 2);

}

else

{

if (j == 0 || j == height - 1)

{

energy = Math.Pow(pixel[i - 1, j].r - pixel[i + 1, j].r, 2) +

Math.Pow(pixel[i - 1, j].g - pixel[i + 1, j].g, 2) +

Math.Pow(pixel[i - 1, j].b - pixel[i + 1, j].b, 2);

}

else

{

energy = Math.Pow(pixel[i - 1, j].r - pixel[i + 1, j].r, 2) +

Math.Pow(pixel[i - 1, j].g - pixel[i + 1, j].g, 2) +

Math.Pow(pixel[i - 1, j].b - pixel[i + 1, j].b, 2);

energy += Math.Pow(pixel[i, j - 1].r - pixel[i, j + 1].r, 2) +

Math.Pow(pixel[i, j - 1].g - pixel[i, j + 1].g, 2) +

Math.Pow(pixel[i, j - 1].b - pixel[i, j + 1].b, 2);

}

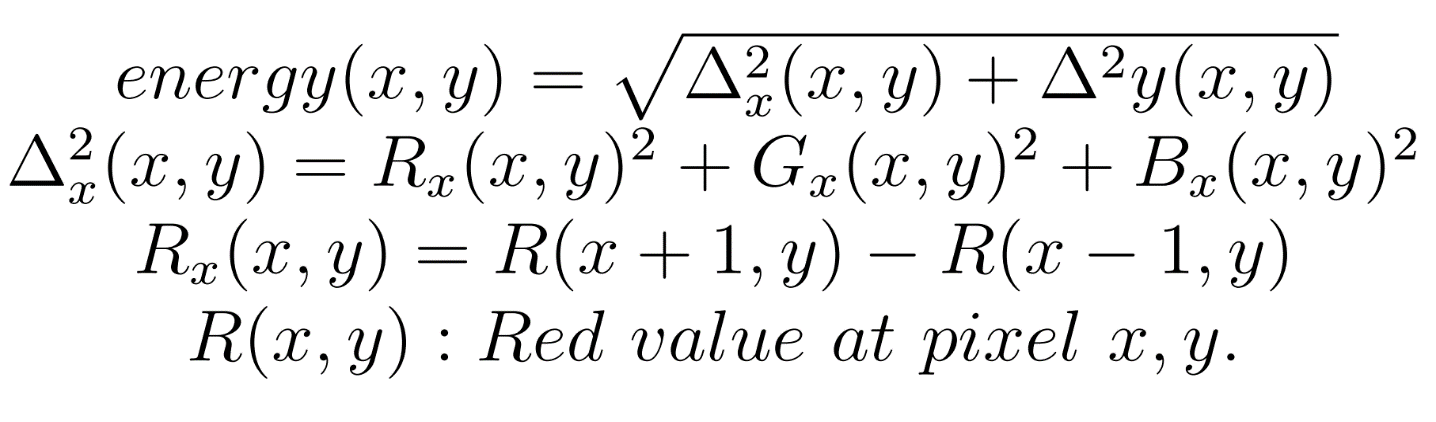
}

}

pixel[i, j].energy = Math.Sqrt(energy);

{

حال در بخش بالا به وسیله 2 تا حلقه for تمامی ارایه ای از کلاس پیکسلی که درست کردیم رو مرور میکنیم و انرژی هر پیکسل رو به صورت فرمولی که داده شده حساب میکنیم.



فقط در پیکسل هایی که دارای همسایه های بالا یا پایین یا چپ و راست نیستند،پیکسل رو به رو را به این پیکسل مپ میکنیم.

به طور مثال یک پیکسل،همسایه بالا ندارد،ما پیکسل پایین را روی پیکسل بالا مپ میکنیم که مطابق فرمول بالا بنوانیم انرژی را حساب بکنیم.

حال باید درز رو پیدا کنیم.برای این کار دوباره دو for توی هم میزنیم و همه اعضای ارایه پیکسل هارو مرور میکنیم.مسیر ما حاوی نود هایی است که جمع انرژی های ااین نود ها با هم،کمترین مقدار ممکن از بین مسیر های ماست.

حال برای اینکه این مسیر رو پیدا کنیم باید این روش رو پیش برویم.هر نود تنها از 3 نود بالا خود امکان دارد مسیری واردش بشود،پس انرژی این نود در حالت پیدا کردن مسیر برابر با مینیمم انرژی 3 نود بالایی خود است(نود بالاو،نود بالا راست و نود بالا چپ)

البته این رو باید رعایت کنیم که مقدار انرژی نود های سطر اول برابر با مقدار انرژی ای است که در کلاس پیکسل ان نود نگه داشتیم.

در هر مرحله وقتی که یکی از 3 نود بالای نود به عنوان مینیمم انتخاب شد،ان نود را به عنوان father ان نود در نظر میگیریم و در کلاس پیکس ان نود ذخیره میکنیم.

if (arr[i - 1, j - 1] < Math.Min(arr[i, j - 1], arr[i + 1, j - 1]))

{

pixel[i, j].father = Tuple.Create(i - 1, j - 1);

arr[i, j] = pixel[i, j].energy + arr[i - 1, j - 1];

}

else

{

if (arr[i, j - 1] < Math.Min(arr[i - 1, j - 1], arr[i + 1, j - 1]))

{

pixel[i, j].father = Tuple.Create(i, j - 1);

arr[i, j] = pixel[i, j].energy + arr[i, j - 1];

}

else

{

pixel[i, j].father = Tuple.Create(i + 1, j - 1);

arr[i, j] = pixel[i, j].energy + arr[i + 1, j - 1];

}

{

کد بالا برای حالتی است که نود دارای 3 نود بالای خود است و father این نود به صورت جفت نگه داری میشود

در انتها وقتی این دو for تمام شد،در پایین ترین سطر ازمجموع انرژی هایی که درست کردیم دنبال کوچیکترین مقدار ممکن میگردیم.

double[] holdarr = new double[width];

for (int i = 0; i < width; i++)

{

holdarr[i] = arr[i, height - 1];

}

Array.Sort(holdarr);

double max = 0;

hold = holdarr[0];

for (int i = 0; i < width; i++)

{

if (hold == arr[i, height - 1])

holdindx = i;

}

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

if (pixel[i, j].energy > max)

max = pixel[i, j].energy;

}

{

به طور مثال این نقطه دارای (x,y) = (I,j) است.حال با شروع از این نطقه و ادد کردن این نقطه به به لیست،پدر این نقطه را نیز به لیست اضافه میکنیم.در هر مرحله پدر اخرین نودی که اضافه کردیم را به لیست ذخیره میکنیم و این کار را تا زمانی ادامه میدهیم که به بالاترین نقطه ممکن در عکسمان برسیم و به این ترتیب درز خود را پیدا کردیم.

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

color = Color.FromArgb((int)(pixel[i, j].energy \* 255.9 / max), (int)(pixel[i, j].energy \* 255.9 / max), (int)(pixel[i, j].energy \* 255.9 / max));

saveBitmap.SetPixel(i, j, color);

}

{

حال این نقاط را به وسیله دستور SetPixel به رنگ قرمز در میاریم و باقی نقاط رو به وسیله همین دستور به نسب انرژی که دارند رنگ میکنیم.برای این کار بزرگترین انرژی ممکن در پیکسل هارد به عنوان max در نظر میگیریم و مقدار rgb برای ست کردن این پیکسل را برابر با pixel[i, j].energy \* 255.9 / max قرار میدهیم که با توجه به نسبت انرژی انها یک مقدار بین 1 تا 255.9 باشد.

saveBitmap.RotateFlip(RotateFlipType.Rotate270FlipNone);

saveBitmap.Save(@"D:\university\AD entezari\ADproj\Create\" + name +@"\horizontalWithRed" + counter + ".jpg");

saveBitmap.RotateFlip(RotateFlipType.Rotate90FlipNone);

حال بیت مپ را 270 درجه دوران میدهیم و سیو میکنیم و دوباره 90 درجه دوران میدهیم و کار خود را ادامه میدهیم.

در مرحله بعد باید این درز را حذف کنیم.تمامی پیکسل های راست این پیکسل ها را یکی به سمت چپ انتقال میدهیم و به این صورت پیکسل های درز حذف میشوند.

نحوه شیفت هم به این صورت است که مشخصات کلاس پیکسلی که برای ان نود درست کردیم را بهنود سمت چپ مانند زیر انتقال میدهیم.

for (int i = 0; i < height; i++)

{

holdx = tuples[i].Item1;

holdy = tuples[i].Item2;

for (int j = holdx; j < width - 1; j++)

{

pixel[j, holdy].b = pixel[j + 1, holdy].b;

pixel[j, holdy].r = pixel[j + 1, holdy].r;

pixel[j, holdy].g = pixel[j + 1, holdy].g;

pixel[j, holdy].father = pixel[j + 1, holdy].father;

pixel[j, holdy].energy = pixel[j + 1, holdy].energy;

color = Color.FromArgb(pixel[j, holdy].r, pixel[j, holdy].g, pixel[j, holdy].b);

myBitmap.SetPixel(j, holdy, color);

}

}

حال تعداد سطر هارو یکی کم میکنیم و کار بالا رو از اول به تعداد سطر هایی که میخواهیم حذف کنیم ادامه میدهیم.

حال در انتها باید حواسمان باشد که در مراحل بالا در بخشی که پیکسل هارو یکی به چپ شیفت میکردیم سایز عکس تغییر نمیکند و سمت چپ ترین پیکسل ها در اخر سر شبیه به هم میشوند.پس حال باید سایز عکس را نیز کوچیک کنیم.یک Bitmap جدید تعریف میکنیم و سایز ان را به صورت زیر تعرف میکنیم.اگربه طورمث ما 100 ردیف حذف کرده باشیم،سایز ان مثل عکس ورودی است با این تفاوت که ارتفاع ان 100 تا کمتر است.

Bitmap finalBitmap = new Bitmap(width, height);

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

finalBitmap.SetPixel(i, j, myBitmap.GetPixel(i, j));

}

Console.WriteLine("horizontalfinal Done");

finalBitmap.RotateFlip(RotateFlipType.Rotate270FlipNone);

با دو for این بیت مپ جدید را پر میکنیم . کار ما در این بخش تمام میشود و در جایی این عکس جدید را به عنوان horizontalfinalImage ذخیره میکنیم و ذر اخر سر هم 270 درجه عکس را دوران میدهیم.

کار در کلاس vertical نیز دقیق مثل بالاست ولی با این تفاوت که دیگر rotate نداریم و ورودی عکس در این کلاس،عکس خروجی در کلاس horizontal است.در اخر سر در همان مسیری که عکس خروجی horizontal ذخیره شد،عکس خروجی نهایی را ذخیره میکنیم.

در تابع main هم تنها کاری که میکنیم درست کرد فلدر های ضروری و کال کردن 2 کلاس بالاست.

horizontal horizontal = new horizontal();

horizontal.horizontalRemove(horizontalremove, myBitmap, name);

vertical vertical = new vertical();

vertical.VerticalRemove(verticalremove, name);

orderه زمانی این کد در هر کلاس به اندازه width \* height \* (horizontalremove + verticalremove) است.

بزرگترین for های ما به اندازه عرض ضربدر ارتفاع است.

پس حالت عادی برای پیدا کردن یه مسیر ما چندین حقه داریم که ماکسیمم order انها width \* height است.

ولی چون میخواهیم چندی مسیر پیدا کنیم باید در horizontalremove + verticalremove نیز ضرب کنیم.

البته دلیل کندی سوال اینه که در مرحله اول باید از هارد به روی رم داد های عکس روی کلاس بیت مپ ذخیره بشه.

به عنوان مثل برای عکس زیر خروجی باحذف 100 سطر و 100ستون به عکس زیر تبدیل میشود.



که عک سمت راست به عکس سمت چپ تبدیل شد.به طور مثال یکی از مسیر های تولیذ شده مثل زیر است.

عکس زیر مربوط به مرحله 11 و حذف ستونی است.

