VB图像处理之几个常用滤镜的实现 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/137/2021_2022_VB_E5_9B_BE_E5_83_8F_E5_A4_c97_137660.htm 前面讲到了二次线性插值的应用。这一篇来给大家讲一下关于锐化、柔化、扩散、雕刻这几个滤镜的实现。 一、锐化 锐化的算法很简单，就是比较相邻的几个像素，把当前像素加上和周围的像素的差就可以了。这里我给出一个示例： A B C D E F G H I J K L M N O P 假设有一个图片，4*4，共16个像素，分别用A－－L来代表。我们先观察这个图片，只有中间的F,G,J,K这四个像素的“ 邻居 ”是全的。 为了简便起见，我们只处理这4个像素，因为在实际的图片中由于图片的大小都很多像素组成，所以周围的一圈像素不做处理不会影响到最终的效果。 先计算差值: Delta= F - (A B C E G I J K) / 8 (A B C E G I J K) / 8就是F周围的像素的平均值， 将这个平均值乘以一个系数再加到F上，就得到了一个新的F值： F=F Delta * Alpha 这个系数Alpha就是锐化度，改变这个系数就能得到不同的锐化效果。不过一般都是取得比较小的，如：0.3 于是，我们只要使用两个循环来遍历整个图片的像素值（去除边界）就能得到一个锐化的效果了。 但是大家或许会发现在处理后面几个点的时候，前面的点的值已经不是原来的值了，比如处理G的时候，需要用到F的值，而F则已经被改变，并且F的改变又和G的值有关系，这样就会变成一种循环引用。为了避免整个问题，这里给出一个改良的方法： A B C D E F G H I J K L M N O P 我们从A点开始做，将差值计算方法改成: Delta= A - (B E F) / 3 F=F Delta * Alpha 按照从左到右，从上到下的顺序来扫描所有像素，这时

在计算中就不会遇到已经被处理过的像素了，并且因为减少了参与运算的像素，整个处理过程也得以加快。 按照我们在《VB图像处理之像素的获取和输出》中已经得到的像素数组。我们可以这样写： Public Sub Sharp(Optional ByVal SharpDgree As Single = 0.3) Dim X As Long Dim Y As Long Dim Ix As Long Dim Iy As Long Dim Diff As Long Dim Diff1 As Long Dim Div1 As Single Dim Div2 As Single Dim Max As Long On Error GoTo ErrLine Max = 255 Done = False TimeFilter = timeGetTime TemplateSize = 1 Sensitivity = Sensitivity * 9 Div1 = 1 SharpDgree Div2 = -SharpDgree / 3 For X = 0 To OutPutWid - 1 For Y = 0 To OutPutHei -1 RR = ColOut(0, X, Y) * Div1 GG = ColOut(1, X, Y) * Div1 BB = ColOut(2, X, Y) * Div1 Ix = X 1 Iy = Y 1 R = ColOut(0, Ix, Iy) R = R ColOut(0, X, Iy) ColOut(0, Ix, Y) G = ColOut(1, Ix, Iy) G = G ColOut(1, X, Iy) ColOut(1, Ix, Y) B = ColOut(2, Ix, Iy) B = B ColOut(2, X, Iy) ColOut(2, Ix, Y) R = R * Div2 G = G * Div2 B = B * Div2 RR = RR R GG = GG G BB = BB B If RR If RR > Max Then RR = Max If GG If GG > Max Then GG = Max If BB If BB > Max Then BB = Max ColOut(0, X, Y) = RR ColOut(1, X, Y) = GG ColOut(2, X, Y) = BB Next Next Done = True TimeFilter = timeGetTime - TimeFilter Exit Sub ErrLine: Done = True MsgBox Err.Description End Sub 因为在计算新的像素的过程中会出现新的值大于255或小于0的情况，因此必须在计算完成后判断。 所用到的全局变量： Public TimeFilter As Long 用于记录滤镜处理所花费的时间 Dim RR As Long 用于保存红色分量 Dim GG As Long 用于保存绿色分量 Dim BB As Long 用于保存蓝色分量 原图： 锐化效果： 二、柔化 柔化的算法和锐化相近似

，不过作用正好相反，就是把当前点用周围几个点的平均值来代替。 A B C D E F G H I J K L M N O P 计算方法： F=(A B C E F G I J K) / 9 G=(B C D F G H J K L) / 9 ... ... 具体的程序，我这里就不罗嗦了，大家只要把上面的程序小小改动一下就可以了。 原图： 柔化效果： 三、扩散 产生一种类似水彩画的效果。 算法很简单，就是将当前点用周围的随即的点来代替。 A B C D E F G H I J K L M N O P F点可以从它周围的A,B,C,E,G,I,J,K中任意选一点代替。 G点可以从它周围的B,C,D,F,H,J,K,L中任意选一点代替。 J点可以从它周围的E,F,G,I,K,M,N,O中任意选一点代替。 K点可以从它周围的F,G,H,J,L,N,O,P中任意选一点代替。 至于选哪一点，可以用一个随即数来选定。 原图： 扩散效果： 四、雕刻 将相邻的两个像素相减，得到的差加上127作为新的值 A B C D E F G H I J K L M N O P 如果我们按照从左向右的方向来“ 雕刻 ” A=B-A 127 B=C-B 127 C=D-C 127 ... 如果我们按照从上向下的方向来“ 雕刻 ” A=E-A 127 B=F-B 127 C=G-C 127 ... 当然我们还可以从更多的方向来“ 雕刻 ”比如：向左下、右上、左上、右下...等等，一共8个可以选择的方向。 另外这个127，就是“ 雕刻 ”效果后的亮度。我们可以把雕刻方向和亮度都作为参数写到过程中 Public Sub Emboss(Optional EmbossDirection As Integer , Optional Lighteness As Integer) ... 原图： 柔化效果： 这几个滤镜的算法都比较简单，很容易用VB来实现。