图像处理笔记 —— 卷积 (/a/1190000004644078)

图像处理 (https://segmentfault.com/t/图像处理/blogs) c++ (https://segmentfault.com/t/c++/blogs) opencv (https://segmentfault.com/t/opencv/blogs)

Shihira (https://segmentfault.com/u/shihira) 2 天前发布

这篇文章是我以前在别的地方发的，最近发现Segmentfault把公式bug修好了，搬过来

网上有各种各样对卷积的理解，有搞EE的，有搞CS的，有搞数学的。我尝试从图像处理的角度加入自己的理解。

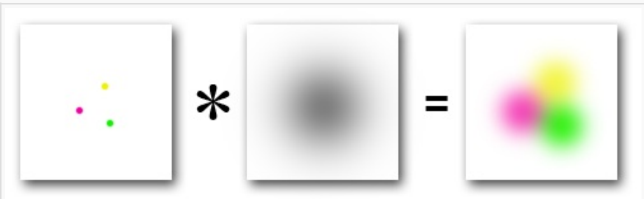
输入、响应和输出

在这里，输入是红绿黄三个点，对于每个点，它的响应是一个尖头向右下的水滴状，最右就是整个图像在系统响应后的输出。怎样理解**响应**呢？你可以把输入当作是纸面上一滴滴颜料，响应就是你用手指把它们在纸上抹开（先暂时这样理解）。现在我们化二维为一维，然后来定量分析一下：

先把输入、响应和输出分别记作 $f(x)$, $h(x)$, $g(x)$。在本例中，输入是一些离散点（比如 $f = \{ \langle x_1, y_1\rangle, \langle x_2, y_2 \rangle \}$），而响应是一个分布集中在零附近的函数（比如 $h(x) = e^{-x^2}$）。现在，在输出中每个点都有一个响应分布在这个点周围，比如对于第一个点，输出就是：$g(x) = f(x_1)\ h(x - x_1)$

这里要感谢响应（或者说系统作出的响应）的**时不变**性质，解释起来很简单，就是它无论对哪个点发生响应都是这种水滴状，不会变形，也不会有幅度上的变化。

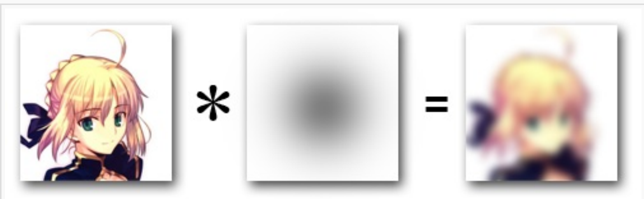
叠加原理



刚才那三点离得比较远，互不影响。现在我们把它靠近一点……它们之间的颜色就会混在一起了。加上这个叠加原理，就不是像手指涂抹颜料一样的混合（Blend），而是像2+3=5之类的简单加法。接着上面所设，设输入了两个点，如果有一点x，x1和x2都影响到了它，它的输出就是：$g(x) = f(x_1) h(x - x_1) + f(x_2) h(x - x_2)$

我们之所以能直接把它加起来，都是得益于响应的 **线性性** 性质，它保证了这个加号是成立的。*（为什么不能是混合：因为这里输出是跟响应顺序无关的，然而混合是有顺序的效应的）*

更密集……甚至连续



刚才的点，无论怎么说，还有一定的间距。但是当输入连续地分布、而且每一点都按照响应的形式扩散开来的时候，我们就可以用到积分或者连加。最后……这就是卷积的最终效果。

这个想法是很自然的：用连加号代替离散但是数量庞大的输入和它们的响应，用积分来处理连续的输入和响应。比如说，输入中有N个值：$[f_1, f_2, \cdots, f_N]$，在它后方产生的响应表示成：$[\cdots, h_{-1}, h_0, h_1, \cdots]$，输出是另一个向量，其中的元素：$g[k] = \sum_{n = 1}^N f[n]\ h[k - n]$

如果是连续函数，式子便是：$g(x) = \int_{-\infty}^{+\infty} f(t) h(x - t)\ \mathrm{d}t$

现在，这两种形式我们分别叫做离散形式下和连续形式下的卷积，记作 $g(x) = f(x) * h(x)$。其中，$h(x)$ 有一个名字，叫做卷积核。

二维离散卷积和算法

以此类推，用二元组（向量）代替标量，$[i, j]$ 代替 k，$[m, n]$ 代替 n，二维的离散卷积的公式应该是这样：$g[i, j] = \sum_{n=1}^N\ \sum_{m=1}^M f[m, n]\ h[i - m, j - n]$

到具体算法，有两个特殊问题要考虑：

- 边界方案：最简单的方法是把边界外的输入当作0，但是这样效果不好。我选用的方案是**镜面**，也就是： $f[m, n] \rightarrow f[(M-\left| m-M\right|)\ \mathrm{mod}\ 2M, (N-\left|n-N\right|)\ \mathrm{mod}\ 2N]$
- 离散卷积核：按需舍弃一些看上去已经很接近0的点来简化计算，比如高斯函数，大多值分布在 $\pm 3\sigma$ 之间，这样我们卷积核的大小也定为 $2 \lfloor 3\sigma\rfloor + 1$就好了。现在，能影响到点 (i, j) 的输入也就是只有附近的有限个点了，它们满足 $\left| n - i \right| \leq A;\ \left| m -j \right| \leq B$，其中2A+1和2B+1分别是卷积核的长宽，换进式子里，就是： $\sum_{n=1}^N\ \sum_{m=1}^M \rightarrow \sum_{n=j-B}^{j+B}\ \sum_{m=i-A}^{i+A}$

```
void convolution(const Mat& in, const Mat& ker, Mat& out)
{
    assert(in.rows == out.rows && in.cols == out.rows);
    assert(in.type == CV_64FC3 && ker.type == CV_64F && out.type == CV_64FC3);

    for(int i = 0; i < out.rows; i++)
    for(int j = 0; j < out.cols; j++) {
        out.at<Vec3d>(i, j) = Vec3d(0, 0, 0);
        for(int m = i - ker.rows; m <= j + ker.rows; m++)
        for(int n = j - ker.cols; n <= i + ker.cols; n++) {
            Point src_point(
                (in.rows - abs(m - in.rows)) % (2 * in.rows),
                (in.cols - abs(n - in.cols)) % (2 * in.cols));
            out.at<Vec3d>(i, j) +=
                in.at<double>(src_point) *
                ker.at<Vec3d>(i - m, j - n);
        }
    }
}
```

我们刚才算法的“卷积”是这样的理解：**各点按照核给出的模式/图案，影响到附近的点**，现在我们换一个方式去理解：**某一个点按照给出的模式/图案收集附近的点的影响**，就可以更加直观理解这个算法。

2 天前发布 (/a/1190000004644078)

2 推荐 收藏

你可能感兴趣的文章

Python-OpenCV 图像与视频处理 (https://segmentfault.com/a/1190000003742481) 32 收藏，2.2k 浏览

Lecture 3 opencv2系列之遍历Mat (https://segmentfault.com/a/1190000000598650) 2 收藏，2.1k 浏览

Python-OpenCV 处理图像（八）：图像二值化处理 (https://segmentfault.com/a/1190000003755115) 4 收藏，2.1k 浏览

讨论区

请先 登录 () 后评论

本文隶属于专栏

Shihira (https://segmentfault.com/blog/shihira)

桜、雪、電車

Shihira (https://segmentfault.com/u/shihira)
作者

关注专栏

分享扩散：

...