

## 赵春江的专栏

目录视图

摘要视图

RSS 订阅

## 个人资料



zhaocj



访问: 493628次

积分: 7037

等级: BLOG &gt; 6

排名: 第2217名

原创: 84篇 转载: 0篇

译文: 0篇 评论: 2527条

## 文章搜索

## 文章分类

ARM系统 (24)

u-boot-2011.06 (11)

arm-linux驱动 (25)

opencv (24)

## 文章存档

2016年09月 (1)

2016年06月 (2)

2016年05月 (2)

2016年02月 (1)

2016年01月 (4)

展开

## 阅读排行

s3c2440启动文件详细分 (20702)

s3c2440裸奔之结束语 (18074)

Opencv2.4.9源码分析— (16499)

s3c2440的LCD应用

深度学习代码专栏 攒课--我的学习我做主 开启你的知识管理, 知识库个人图谱上线

## Opencv2.4.9源码分析——adaptiveBilateralFilter

2014-09-25 17:41

5031人阅读

评论(7)

收藏

举报

分类: opencv (23)

版权声明: 本文为博主原创文章, 未经博主允许不得转载。

上一篇文章我们介绍了双边滤波, 它的公式为:

$$h(x) = k^{-1}(x) \sum f(\xi) c(\xi, x) s(\xi, x) \quad (1)$$

其中,  $c(\xi, x) = e^{-\frac{1}{2}(\frac{d(\xi, x)}{\sigma_d})^2}$ ,  $s(\xi, x) = e^{-\frac{1}{2}(\frac{\sigma(f(\xi), f(x))}{\sigma_r})^2}$ ,  $k(x) = \sum c(\xi, x) s(\xi, x)$ ,  $f(\xi)$ 表示原图。

$c(\xi, x)$ 表示的是高斯距离的权值,  $\sigma_d$ 值大则滤波结果会受到更远的像素影响;  $s(\xi, x)$ 表示的是高斯相似度的权值,  $\sigma_r$ 值大则意味着更无关的像素强度值(或颜色值)会影响滤波器结果。因此这两个值的选取会直接影响到滤波效果。

关于高斯距离的权值, 还会受到滤波内核大小的影响, 因此它的方差 $\sigma_d$ 值对滤波结果的影响会受到一定的约束, 但 $\sigma_r$ 值的选取就难以把握, 因此本算法的目的就是自适应的选取 $\sigma_r$ 值的大小。

在opencv文档中没有说明该算法的出处, 但从它的程序源码中可以分析得到,  $\sigma_r$ 值是通过领域内的像素值得到, 具体公式为:

$$\sigma_r^2 = \frac{n \sum_{i=1}^n I(i)^2 - \sum_{i=1}^n I(i)}{n^2} \quad (2)$$

其中,  $n$ 表示邻域内的像素个数, 该邻域指的是滤波内核,  $I(i)$ 表示的是像素值。

下面我们来分析一下具体的代码, 该函数的原型为:

```
void adaptiveBilateralFilter(InputArray src, OutputArray dst, Size ksize, double sigmaSpace, double maxSigmaColor=20.0, Point anchor=Point(-1, -1), int borderType=BORDER_DEFAULT)
```

`_src`为输入原图像; `_dst`为滤波后的图像; `ksize`为滤波内核的大小; `sigmaSpace`为距离权值公式中的方差, 即公式1中的 $\sigma_d$ ; `maxSigmaColor`为相似度权值公式中的方差( $\sigma_r$ )的最大值, 自适应双边滤波的相似度方差是通过公式2计算得到, 但如果计算的结果太大, 超过了该值, 则以该值为准; `anchor`为内核锚点; `borderType`表示用什么方式来处理加宽后的图像四周边界。

该函数的源码是在/sources/modules/imgproc/src/smooth.cpp内:

s3c2440外部中断操作 (15635)  
s3c2440的UART用法 (14939)  
Win7下qt5.3.1+opencv2 (14672)  
s3c2440的触摸屏应用与 (13075)  
s3c2440的IIC应用——读 (12299)  
s3c2440的LCD字符显示 (11883)  
s3c2440的LCD字符显示 (11459)

#### 评论排行

s3c2440的IIS应用——放 (180)  
s3c2440的LCD字符显示 (161)  
s3c2440的摄像接口应用 (152)  
s3c2440的UART用法 (151)  
s3c2440的网卡接口扩展 (148)  
s3c2440的IIC应用——读 (129)  
s3c2440的触摸屏应用与 (117)  
s3c2440外部中断操作 (111)  
s3c2440对nandflash的操 (108)  
s3c2440裸奔之结束语 (107)

#### 推荐文章

\* 2016 年最受欢迎的编程语言是什么?  
\* Chromium扩展 (Extension) 的页面 (Page) 加载过程分析  
\* Android Studio 2.2 来啦  
\* 手把手教你做音乐播放器 (二) 技术原理与框架设计  
\* JVM 性能调优实战之: 使用阿里开源工具 TProfiler 在海量业务代码中精确定位性能代码

#### 最新评论

Opencv2.4.9源码分析——MSCF zhaocj: @ZYK12:没有做过相关的内容, 也没有什么好的方法。  
Opencv2.4.9源码分析——MSCF ZYK12: 博主 您好 如果我左右两张图片检测出MSCR区域后 想匹配对应的区域 您有好的方法推荐吗  
Opencv2.4.9源码分析——Hougl guanyonglai: @zhaocj:好的, 谢谢了, 对照你说的我再看看吧  
Opencv2.4.9源码分析——Hougl zhaocj: @guanyonglai:208行是先初始化, 该数组是用于函数icvHoughSortDescent...  
Opencv2.4.9源码分析——Hougl zhaocj: @guanyonglai:1、要防止检测到的圆过小, 第一次位移的时候, 要把位移量直接定位到最小圆之外...  
Opencv2.4.9源码分析——Hougl zhaocj: @guanyonglai:性质是一样的呀, 都是int型  
Opencv2.4.9源码分析——Hougl zhaocj: @guanyonglai:这就是高等数学中取极限的具体体现, 你把一个圆无限放大就会发现, 其实它就是这...  
Opencv2.4.9源码分析——phase zhaocj: @u012739916:该算法是基于数字信号处理中, 时域平移对应于频域相移的原理。根据这个定理是无法...  
OpenCV2.4.9源码分析——Supp xiaoCCluo: Great job! Thank

[cpp]

```
01. void cv::adaptiveBilateralFilter( InputArray _src, OutputArray _dst, Size ksize,  
02.                                double sigmaSpace, double maxSigmaColor, Point anchor, int borde  
03. {  
04.     //得到输入图像矩阵和与其尺寸类型一致的输出图像矩阵  
05.     Mat src = _src.getMat();  
06.     _dst.create(src.size(), src.type());  
07.     Mat dst = _dst.getMat();  
08.     //该算法只能处理8位二进制的灰度图像和三通道的彩色图像  
09.     CV_Assert(src.type() == CV_8UC1 || src.type() == CV_8UC3);  
10.     //得到滤波内核的锚点  
11.     anchor = normalizeAnchor(anchor,ksize);  
12.     if( src.depth() == CV_8U )  
13.         adaptiveBilateralFilter_8u( src, dst, ksize, sigmaSpace, maxSigmaColor, anchor, borderType  
14.     else  
15.         CV_Error( CV_StsUnsupportedFormat,  
16.             "Adaptive Bilateral filtering is only implemented for 8u images" );  
17. }
```

[cpp]

```
01. static void adaptiveBilateralFilter_8u( const Mat& src, Mat& dst, Size ksize, double sigmaSpace, c  
02. {  
03.     Size size = src.size();  
04.     ///处理之前再次检查图像中的相关信息是否正确  
05.     CV_Assert( (src.type() == CV_8UC1 || src.type() == CV_8UC3) &&  
06.         src.type() == dst.type() && src.size() == dst.size() &&  
07.         src.data != dst.data );  
08.     //为了在图像边界处得到更好的处理效果, 需要对图像四周边界做适当的处理  
09.     //把原图的四周都加宽, 而加宽部分的像素值由borderType值决定  
10.     //待处理的图像由src换成了temp  
11.     Mat temp;  
12.     copyMakeBorder(src, temp, anchor.x, anchor.y, anchor.x, anchor.y, borderType);  
13.     //通过实例化adaptiveBilateralFilter_8u_Invoker类计算得到自适应双边滤波的结果  
14.     adaptiveBilateralFilter_8u_Invoker body(dst, temp, ksize, sigmaSpace, maxSigmaColor, anchor);  
15.     parallel_for_(Range(0, size.height), body, dst.total()/(double)(1<<16));  
16. }
```

我们先看一下adaptiveBilateralFilter\_8u\_Invoker类的构造函数:

[cpp]

```
01. adaptiveBilateralFilter_8u_Invoker(Mat& _dest, const Mat& _temp, Size _ksize, double sigma_space,  
02.     temp(&_temp), dest(&_dest), ksize(_ksize), sigma_space(_sigma_space), maxSigmaColor, Sigma  
03. {  
04.     if( sigma_space <= 0 ) //确保sigma值不能小于零  
05.         sigma_space = 1;  
06.     CV_Assert((ksize.width & 1) && (ksize.height & 1)); //确保滤波内核的宽和高是奇数  
07.     space_weight.resize(ksize.width * ksize.height);  
08.     double sigma2 = sigma_space * sigma_space;  
09.     int idx = 0;  
10.     int w = ksize.width / 2;  
11.     int h = ksize.height / 2;  
12.     //遍历整个内核, 计算高斯距离权值  
13.     for(int y=-h; y<=h; y++)  
14.         for(int x=-w; x<=w; x++)  
15.         {  
16.             //在程序中定义了宏ABF_GAUSSIAN, 因此高斯距离权值使用的是本文中给出的公式  
17.             #if ABF_GAUSSIAN  
18.                 space_weight[idx++] = (float)exp ( -0.5*(x * x + y * y)/sigma2);  
19.             #else  
20.                 space_weight[idx++] = (float)(sigma2 / (sigma2 + x * x + y * y));  
21.             #endif  
22.         }  
23. }
```

下面再来介绍adaptiveBilateralFilter\_8u\_Invoker类中的operator():

[cpp]

```
01. virtual void operator()(const Range& range) const
```

you very much!

Opencv2.4.9源码分析——Hougl  
guanyonglai: sobel算法用的是1  
阶 3\*3的孔径,也就是说用sobel  
得出的梯度方向只有45°,  
90,135,...



工业相机



超细粉碎机



滤波器



发电电

```
02. {
03.     int cn = dest->channels();    //得到图像的通道数,即是灰度图像还是彩色图像
04.     int anX = anchor.x;
05.
06.     const uchar *tptr;
07.
08.     for(int i = range.start;i < range.end; i++)
09.     {
10.         int startY = i;
11.         if(cn == 1) //灰度图像处理方法
12.         {
13.             float var;           //方差
14.             int currVal;         //当前像素值
15.             int sumVal = 0;      //方差和
16.             int sumValSqr = 0;   //方差的平方和
17.             int currValCenter;   //当前内核中心的像素值,即待处理的像素值
18.             int currWRTCenter;   //像素的权值
19.             float weight;
20.             float totalWeight = 0.;
21.             float tmpSum = 0.;
22.
23.             for(int j = 0;j < dest->cols *cn; j+=cn)
24.             {
25.                 sumVal = 0;
26.                 sumValSqr = 0;
27.                 totalWeight = 0.;
28.                 tmpSum = 0.;
29.
30.                 // Top row: don't sum the very last element
31.                 int startLMJ = 0;
32.                 int endLMJ = ksize.width - 1;
33.                 int howManyAll = (anX *2 +1)*(ksize.width );    //内核像素个数总和
34.                 //在程序前面定义了宏ABF_CALCVAR, 因此执行#if的内容
35.                 #if ABF_CALCVAR
36.                     //遍历整个内核, 计算高斯相似度权值的方差
37.                     for(int x = startLMJ; x< endLMJ; x++)
38.                     {
39.                         //内核中某个像素的指针
40.                         tptr = temp->ptr(startY + x) +j;
41.                         for(int y=-anX; y<=anX; y++)
42.                         {
43.                             currVal = tptr[cn*(y+anX)];    //像素值
44.                             sumVal += currVal;             //像素之和
45.                             sumValSqr += (currVal *currVal);    //像素平方之和
46.                         }
47.                     }
48.                     //由公式2得到方差
49.                     var = ( (sumValSqr * howManyAll)- sumVal * sumVal ) / ( (float)
(howManyAll*howManyAll));
50.                     //如果计算得到的方差太小, 则取值0.01
51.                     //如果计算得到的方差超过在调用该函数中所给定的方差, 则以给定的方差为准
52.                     if(var < 0.01)
53.                         var = 0.01f;
54.                     else if(var > (float)(maxSigma_Color*maxSigma_Color) )
55.                         var = (float)(maxSigma_Color*maxSigma_Color) ;
56.
57.                 #else
58.                     var = maxSigmaColor*maxSigmaColor;
59.                 #endif
60.
61.                 startLMJ = 0;
62.                 endLMJ = ksize.width;
63.                 tptr = temp->ptr(startY + (startLMJ+ endLMJ)/2);
64.                 currValCenter =tptr[j+cn*anX];    //内核中心像素, 即带处理的像素值
65.                 //再次遍历内核, 这次是由公式1得到输出图像
66.                 for(int x = startLMJ; x< endLMJ; x++)
67.                 {
68.                     tptr = temp->ptr(startY + x) +j;
69.                     for(int y=-anX; y<=anX; y++)
70.                     {
71.                         #if ABF_FIXED_WEIGHT
72.                             weight = 1.0;
73.                         #else
74.                             currVal = tptr[cn*(y+anX)];
75.                             //内核领域内的像素与内核中心像素之差,
76.                             currWRTCenter = currVal - currValCenter;
77.                             //在程序前面定义了宏ABF_GAUSSIAN, 因此利用高斯函数得到整个权值 (距离权值和相似度权值)
```

```

77.     #if ABF_GAUSSIAN
78.         weight = exp ( -0.5f * currWRTCenter * currWRTCenter/var ) * space_weight[
79.     #else
80.         weight = var / ( var + (currWRTCenter * currWRTCenter) ) * space_weight[x^
81.     #endif
82.
83. #endif
84.
85.         //得到公式1中的分子部分
86.         tmpSum += ((float)tptr[cn*(y+anX)] * weight);
87.         //得到公式1中的分母部分
88.         totalWeight += weight;
89.     }
90. }
91. tmpSum /= totalWeight; //得到公式1的最终结果
92. //把结果赋值给输出图像
93. dest->at<uchar>(startY ,j)= static_cast<uchar>(tmpSum);
94. }
95. else // 处理彩色图像
96. {
97.     assert(cn == 3);
98.     float var_b, var_g, var_r;
99.     int currVal_b, currVal_g, currVal_r;
100.    int sumVal_b= 0, sumVal_g= 0, sumVal_r= 0;
101.    int sumValSqr_b= 0, sumValSqr_g= 0, sumValSqr_r= 0;
102.    int currValCenter_b= 0, currValCenter_g= 0, currValCenter_r= 0;
103.    int currWRTCenter_b, currWRTCenter_g, currWRTCenter_r;
104.    float weight_b, weight_g, weight_r;
105.    float totalWeight_b= 0., totalWeight_g= 0., totalWeight_r= 0.;
106.    float tmpSum_b = 0., tmpSum_g= 0., tmpSum_r = 0.;
107.
108.    for(int j = 0;j < dest->cols *cn; j+=cn)
109.    {
110.        sumVal_b= 0, sumVal_g= 0, sumVal_r= 0;
111.        sumValSqr_b= 0, sumValSqr_g= 0, sumValSqr_r= 0;
112.        totalWeight_b= 0., totalWeight_g= 0., totalWeight_r= 0.;
113.        tmpSum_b = 0., tmpSum_g= 0., tmpSum_r = 0.;
114.
115.        // Top row: don't sum the very last element
116.        int startLMJ = 0;
117.        int endLMJ = ksize.width - 1;
118.        int howManyAll = (anX *2 +1)*(ksize.width);
119.    #if ABF_CALCVAR
120.        float max_var = (float)( maxSigma_Color*maxSigma_Color);
121.        //遍历内核，分别计算红、绿、蓝三个通道的相似度权重方差
122.        for(int x = startLMJ; x< endLMJ; x++)
123.        {
124.            tptr = temp->ptr(startY + x) +j;
125.            for(int y=-anX; y<=anX; y++)
126.            {
127.                currVal_b = tptr[cn*(y+anX)], currVal_g = tptr[cn*
(y+anX)+1], currVal_r = tptr[cn*(y+anX)+2];
128.                sumVal_b += currVal_b;
129.                sumVal_g += currVal_g;
130.                sumVal_r += currVal_r;
131.                sumValSqr_b += (currVal_b *currVal_b);
132.                sumValSqr_g += (currVal_g *currVal_g);
133.                sumValSqr_r += (currVal_r *currVal_r);
134.            }
135.        }
136.        var_b = ( (sumValSqr_b * howManyAll)- sumVal_b * sumVal_b ) / ( (float)
(howManyAll*howManyAll));
137.        var_g = ( (sumValSqr_g * howManyAll)- sumVal_g * sumVal_g ) / ( (float)
(howManyAll*howManyAll));
138.        var_r = ( (sumValSqr_r * howManyAll)- sumVal_r * sumVal_r ) / ( (float)
(howManyAll*howManyAll));
139.
140.        if(var_b < 0.01)
141.            var_b = 0.01f;
142.        else if(var_b > max_var )
143.            var_b = (float)(max_var) ;
144.
145.        if(var_g < 0.01)
146.            var_g = 0.01f;
147.        else if(var_g > max_var )
148.            var_g = (float)(max_var) ;

```

```

149.
150.         if(var_r < 0.01)
151.             var_r = 0.01f;
152.         else if(var_r > max_var )
153.             var_r = (float)(max_var) ;
154.
155.     #else
156.         var_b = maxSigma_Color*maxSigma_Color; var_g = maxSigma_Color*maxSigma_Color; var_
157.     #endif
158.         startLMJ = 0;
159.         endLMJ = ksize.width;
160.         tptr = temp->ptr(startY + (startLMJ+ endLMJ)/2) + j;
161.         currValCenter_b =tptr[cn*anX], currValCenter_g =tptr[cn*anX+1], currValCenter_r =t
162.         //再次遍历内核, 计算最终的结果
163.         for(int x = startLMJ; x< endLMJ; x++)
164.         {
165.             tptr = temp->ptr(startY + x) +j;
166.             for(int y=-anX; y<=anX; y++)
167.             {
168.                 #if ABF_FIXED_WEIGHT
169.                     weight_b = 1.0;
170.                     weight_g = 1.0;
171.                     weight_r = 1.0;
172.                 #else
173.                     currVal_b = tptr[cn*(y+anX)];currVal_g=tptr[cn*
174.                     (y+anX)+1];currVal_r=tptr[cn*(y+anX)+2];
175.                     currWRTCenter_b = currVal_b - currValCenter_b;
176.                     currWRTCenter_g = currVal_g - currValCenter_g;
177.                     currWRTCenter_r = currVal_r - currValCenter_r;
178.
179.                     float cur_spw = space_weight[x*ksize.width+y+anX];
180.                 #if ABF_GAUSSIAN
181.                     weight_b = exp( -0.5f * currWRTCenter_b * currWRTCenter_b/ var_b ) * cur_
182.                     weight_g = exp( -0.5f * currWRTCenter_g * currWRTCenter_g/ var_g ) * cur_
183.                     weight_r = exp( -0.5f * currWRTCenter_r * currWRTCenter_r/ var_r ) * cur_
184.                 #else
185.                     weight_b = var_b / ( var_b + (currWRTCenter_b * currWRTCenter_b) ) * cur_
186.                     weight_g = var_g / ( var_g + (currWRTCenter_g * currWRTCenter_g) ) * cur_
187.                     weight_r = var_r / ( var_r + (currWRTCenter_r * currWRTCenter_r) ) * cur_
188.                 #endif
189.                 #endif
190.
191.                 tmpSum_b += ((float)tptr[cn*(y+anX)] * weight_b);
192.                 tmpSum_g += ((float)tptr[cn*(y+anX)+1] * weight_g);
193.                 tmpSum_r += ((float)tptr[cn*(y+anX)+2] * weight_r);
194.                 totalWeight_b += weight_b, totalWeight_g += weight_g, totalWeight_r += wei
195.             }
196.         }
197.         tmpSum_b /= totalWeight_b;
198.         tmpSum_g /= totalWeight_g;
199.         tmpSum_r /= totalWeight_r;
200.
201.         dest->at<uchar>(startY,j )= static_cast<uchar>(tmpSum_b);
202.         dest->at<uchar>(startY,j+1)= static_cast<uchar>(tmpSum_g);
203.         dest->at<uchar>(startY,j+2)= static_cast<uchar>(tmpSum_r);
204.     }
205. }
206. }

```

下图是使用adaptiveBilateralFilter(src,dst,Size(7,7),75);的自适应双边滤波的结果。



这里还需要说明的是自适应双边滤波adaptiveBilateralFilter要比双边滤波bilareralFilter运行时间更长，而且从源码来看，明显感觉到两个函数不是一个人写的。更重要的是adaptiveBilateralFilter有bug，当滤波内核尺寸取得更大一些的话，比如Size(10,10)，会出现“The application has requested the Runtime to terminate in an unusual way, .....”的错误对话框。

顶 踩  
0 0

上一篇 [Opencv2.4.9源码分析——bilareralFilter](#)  
下一篇 [Opencv2.4.9源码分析——HoughLinesP](#)

我的同类文章

opencv (23)

•

Opencv2.4.9源码分析——E...

2016-09-02

阅读 202

•

Opencv2.4.9源码分析——R...

2016-06-03

阅读 2443

•

OpenCV2.4.9源码分析——...

2016-05-02

阅读 4034

•

Opencv2.4.9源码分析——N...

2016-01-31

阅读 733

•

Opencv2.4.9源码分析——D...

2016-01-12

阅读 1313

•

Opencv2.4.9源码分析——H...

2015-12-13

阅读 1165

•

Opencv2.4.9源码分析——E...

2016-06-12

阅读 2057

•

Opencv2.4.9源码分析——G...

2016-05-20

阅读 3688

•

Opencv2.4.9源码分析——K...

2016-02-29

阅读 639

•

Opencv2.4.9源码分析——B...

2016-01-18

阅读 1126

•

Opencv2.4.9源码分析——H...

2016-01-04

阅读 2073

更多文章

猜你在找

- [解码皮肤美化算法](#)
- [Opencv249源码分析bilareralFilter](#)
- [模板匹配的字符识别\(OCR\)算法原理](#)
- [Opencv249源码分析phaseCorrelate](#)

[查看评论](#)

4楼 mazhiran-persistence 2015-10-29 11:43发表



Re: [zhaocj](#) 2015-11-02 09:09发表



回复mazhiran-persistence: 字面的意思就是停靠点, 就是要计算该点的值。一般来说, 内核都是对称的, 因此该点都是内核的中心点。

3楼 [jerry\\_jewei](#) 2015-08-26 09:25发表



Re: [zhaocj](#) 2015-09-02 13:41发表



回复jerry\_jewei: 在imgproc/src/precomp.hpp文件内

2楼 silver\_R 2015-07-09 01:32发表



那不是BUG,是只能取单数,不能取复数

1楼 [xiaoluo91](#) 2015-05-06 14:28发表



```
var = ( (sumValSqr * howManyAll)- sumVal * sumVal ) / ( (float)(howManyAll*howManyAll));
```

与公式(2)不一致，笔误了。

Re: [zhaocj](#) 2015-05-15 13:46发表



回复xiaoluo91: 是的, 谢谢!

您还没有登录,请[登录](#)或[注册](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

## 核心技术类目

全部主题		Hadoop	AWS	手机游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack	
VPN	Spark	ERP	IE10	Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP	jQuery	
BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS	Fedora	XML	LBS	Unity
Splashtop		UML	components	Windows	Mobile	Rails	QEMU	KDE	Cassandra	CloudStack	FTC	
coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo				
Compuware	大数据	apttech	Perl	Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr		
Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap							

