

Understanding ISP Pipeline

刘斯宁  
Camera技术专家

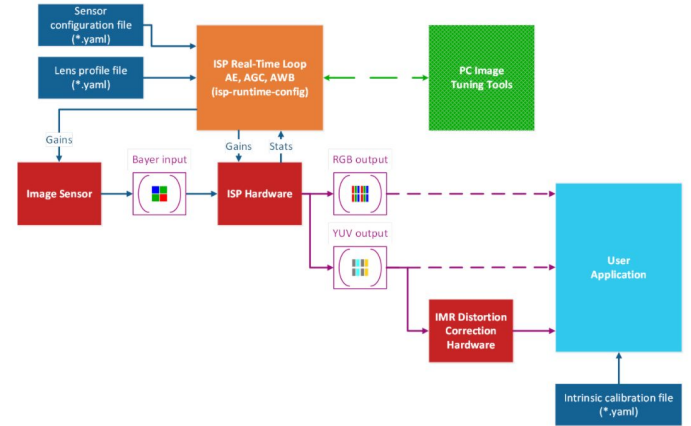
已关注

435 人赞同了该文章

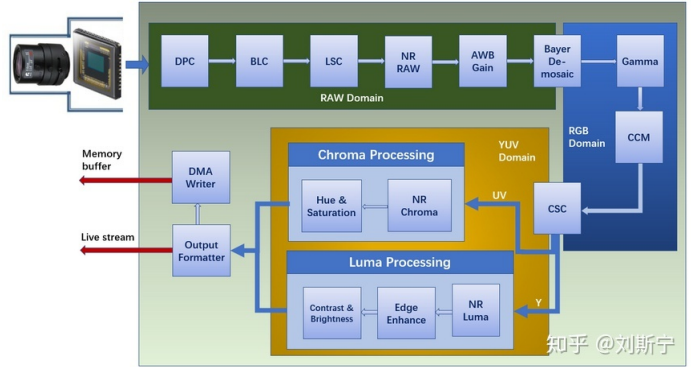
什么是ISP

主流的CMOS和CCD sensor几乎都是输出Bayer mosaic格式的RAW数据，这种数据格式是无法直接观看的，必须转换成常见的RGB或YUV格式才能被主流的图像处理软件支持。对于camera产品而言，一般还需要将RGB或YUV图像进一步转换成JPEG格式以方便进行存储。上述图像处理过程统称图像信号处理（Image Signal Processing，ISP），广义的ISP包含了JPEG和H.264/265图像压缩处理，而狭义的ISP仅包括从RAW格式变换到RGB或YUV的处理过程。

由于图像信号处理涉及大量的数据和严格的实时性要求，所以ISP通常必须采用硬件方案实现。有些Image sensor自身支持一定的ISP功能，用户可以选择启用或者关闭。有些ISP是作为独立的芯片或者SoC IP产品可以从供应商那里采购。下图显示了一个典型camera系统的功能框图，主要包括图像传感器（Image Sensor）、ISP硬件（ISP Hardware）、ISP软件（ISP Real-Time Loop）等核心单元，以及ISP调试工具（PC Image Tuning Tools）、用户程序（User Application）、配置文件等。有些ISP硬件会内嵌支持一些CV算法功能，如镜头畸变校正（Distortion Correction）。



一个典型的ISP流水线由一系列处理模块组成，这些模块首尾相连，在几百MHz的时钟驱动下同时高速运转，图像数据不断从一个模块转移至下一个模块，直到完成所有的算法处理，最终以YUV或RGB的形式从流水线的末级流出ISP。下图所示的是一个支持常见基本功能的ISP流水线。

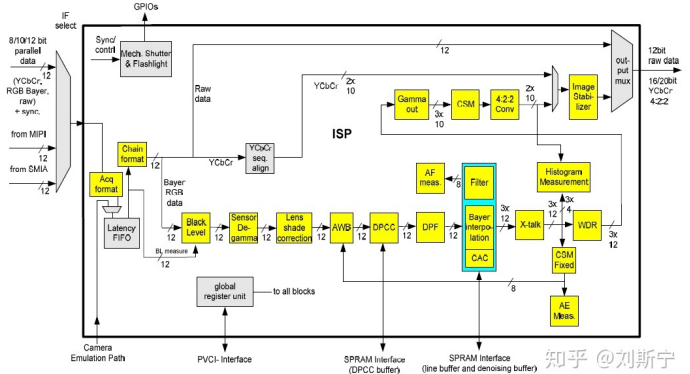


从图中可以看到，图像数据在ISP内部经历了两次颜色空间变换，第一次变换发生在Demosaic模块，它把像素从即RAW域变换到RGB域，第二次变换发生在CSC模块，它把像素从RGB变到YUV域。下表对ISP 各模块的作用给予了简要说明。



序号	模块名	英文全名	功能描述	备注
1	DPC	Defective Pixel Concealment	坏点校正	
2	BLC	Black Level Correction	黑电平校正	
3	LSC	Lens Shading Correction	镜头阴影校正	
4	NR RAW	Noise Reduction for RAW	RAW域降噪	
5	AWB Gain	Auto White Balance Gain	白平衡增益	
6	Bayer Demosaic	Bayer Demosaic	RGB插值	
7	Gamma	Gamma	Gamma校正	
8	CCM	Color Correction Matrix	颜色矫正矩阵	
9	CSC	Color Space Conversion	颜色空间变换	
10	NR Chroma	Noise Reduction for Chroma	颜色降噪	知乎 @刘斯宁
11	Hue & Saturation	Hue & Saturation Control	色调、色饱和度控制	
12	Edge Enhance	Edge Enhance	边缘增强	
13	Contrast & Brightness	Contrast & Brightness Control	对比度、亮度控制	
14	Data Formatter	Data Formatter	图像格式转换	
15	DMA Writer	DMA Write Controller	写DMA	知乎 @刘斯宁

下面是Silicon Image ISP的功能框图，这个ISP历史比较久了，主要以IP授权的形式卖给芯片设计公司，在中国也有不少客户。这个IP最初设计的时候是面向非专业的消费领域，所以只支持一些最基本的功能，对降噪、白平衡、WDR等重点特性的支持十分薄弱，目前已经难以满足主流需求了。笔者曾在2018年的一个车载流媒体项目中使用某国产廉价sensor搭配这款ISP，堪称是职业生涯中最惨痛的一段遭遇。



ACQ (Acquisition) ， 适配输入RAW图的格式

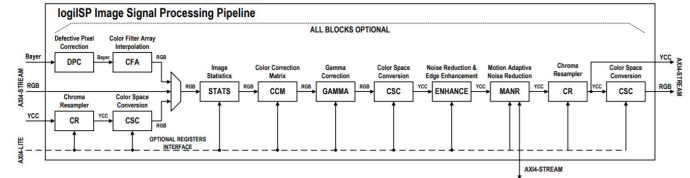
CNR (Chromatic Noise Reduction) ， 色度降噪

X-talk (Crosstalk) ， 串扰，即CCM。其实用Crosstalk指代CCM并不准确，因为它并不是引入CCM的主要原因，这是一种流传颇广的误解。

VSM (Video Stablization Measurement) 视频稳定

CSM (Color Space Matrix) ， 即CSC(Color Space Conversion), RGB2YUV 空间变换

下面是logiiISP的功能框图，可以配合Xilinx Zync FPGA使用，提供最简单的ISP功能。

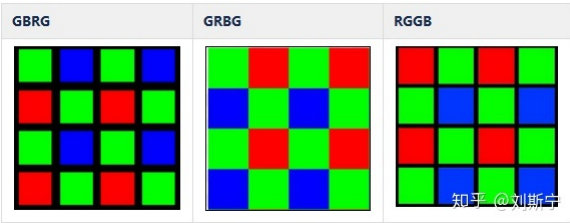


ISP输入图像的格式

目前主流的CMOS sensor几乎都是输出Bayer mosaic格式的RAW数据。Bayer格式图片是伊士曼·柯达公司科学家Bryce Bayer（1929 –2012）发明的，拜耳阵列被广泛运用与数字图像处理领域。



常用的Bayer格式有RGGB、GRBG、GBRG等多种，因此需要正确配置ISP以反应sensor的数据格式。



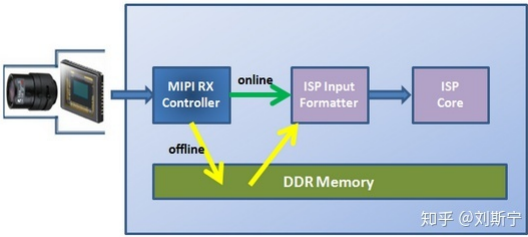
RAW数据的精度常见有8/10/12/14bit等规格。安防监控行业较多使用10/12bit精度的sensor，医疗行业则主要使用12bit以上精度sensor，单反和广电行业则主要使用14bit精度sensor。当精度高于8bit时，就会自然带来如何存储和表示数据的问题。多数sensor支持固定用16bit数据表示10/12/14bit的有效数据，这种表示方法简单易行，但是在传输过程中会浪费一些带宽，在存储环节会浪费一些存储空间。对与一些带宽和存储资源特别紧张的场合，有些sensor会支持压缩表示以节约带宽，但是这就需要ISP能够支持相应的压缩格式，否则就需要增加格式适配处理环节，无论是采用硬件还是软件方式实现都会增加系统的复杂度和成本。

另外，很多相机、摄像机产品都会在镜头光路中插入一个红外截止滤光片，其作用是允许波长小于截止频率的光进入系统，而阻止波长大于截止频率的光。常用的截止频率有630nm和650nm两种，它们允许不同程度的红光进入成像系统，因此会对白平衡和颜色校正算法产生影响，所以ISP的参数配置必须要与实际选用的截止频率相配置。

图像接入ISP的方式

使用ISP处理图像数据时有两种常用的数据接入方式，即

- 在线模式，online mode，sensor产生的实时数据和时序控制信号以行为单位送入ISP进行处理
- 离线模式，offline mode，待处理的图像以帧为单位存储于系统内存，需要处理时由一个控制逻辑通过DMA从内存中读取数据，并添加模拟sensor行为的时序控制信号，然后送给ISP进行处理。



这两种方式在ISP本身看来并没有本质区别，但是从系统角度看，在线模式具备低延迟(low latency)的优点，具体表现是一帧图像的第一个像素数据流出sensor后马上就进入ISP流水线开始处理，而在离线模式下，ISP通常需要等到一帧图像的最后一个像素数据到齐之后才开始启动处理。某些海思系列的ISP芯片支持一种特殊的“低延迟”离线模式，即一帧图像的第X行数据到齐之后即启动ISP进行处理，而不必等到最后一个像素到齐。这种模式自然会引入一个问题，就是如果ISP处理的速度比sensor输出数据的速度快，就会发生ISP读指针很快越过sensor写指针，导致ISP读取到无效数据。下面举一个典型的例子，

假设一个1080p的sensor，主频(master clock frequency)是76MHz，即一个clock cycle是13.15ns。

假设sensor的配置是每行2000个cycle，这是一个比较典型的配置，其中包含了1920个有效输出和80个horizontal blanking，则sensor每输出一行数据将消耗 $13.15ns \times 2000 = 26.3us$ ，而输出全部1080行有效像素则需要 $26.3us \times 1080 = 28.4ms$ 。

而一个典型的ISP处理完这帧1080p图像需要多久呢？只需要3ms！也就是说，ISP只有在26ms之后启动处理才不至于发生读指针越过写指针的情况。这种方法可以简易地规避ISP读到无效数据的问题，但是也只能争取到不足3ms的低延迟收益，延迟性能的提升起始并不明显，可以说是聊胜于无。因此海思又设计了一个更加积极的办法，即增加了一个硬件机制可以确保读指针不会越过写指针，这样用户就可以选择更早地开始启动ISP处理，当ISP读指针追上sensor写指针时，硬件自动插入延时周期令ISP空转等待，以保证数据完整性。

在线模式除了具有低延迟的优点，由于数据走专用的硬件通路，不经过内存总线，所以还具有节省内存带宽的优点。当图像分辨率很大时，比如4K、8K等高分辨率场景，这两个优点就会起到显著



的作用。

行缓冲 Line Buffer

不论是在线模式还是离线模式，ISP处理图像都是以行为单位的，所以ISP模块都会设计一个line buffer可以缓存若干行图像。通常这个line buffer的大小就决定了这个ISP所支持的最大分辨率。举例来说，如果一个ISP的line buffer可以容纳每行2048个像素，则它无法支持超过2k/1080p的分辨率规格。

低延迟系统 Low-Latency System

延迟（Latency）是camera的一个重要技术指标，某些应用场景对视频延迟非常敏感，比如竞速无人机应用，参赛选手需要操纵高速飞行的无人机躲避障碍，做出各种特技动作。当无人机时速200km时，每秒会飞行55米，如果参赛选手看到的camera图像总是0.1秒以前的画面，这会对选手预判无人机的实际位置带来不少困难。因此凡是涉及到高速运动的场合都会希望camera输出的画面尽可能是实时的。

低延迟是一个系统级特性，它需要camera内部各个子系统都要按照最小延迟的原则做设计。一个基本的原则是，所有针对图像的处理都必须是以行为单位，而不能以帧为单位。假设H.264编码器能够工作的最低要求是16行，这就意味着一帧图像的的第一个像素最快也要等到16行数据全部到齐后才能被编码。假设CMOS sensor输出速率为每秒120帧，则每帧图像8.3ms，输出每行数据需要7.7us，16行缓存意味着camera的理论延迟最低可以做到123us。如果采用非优化的常规技术，编码器等一帧图像到齐后才开始编码，则第一个像素需要空等8.3ms之后才能开始倍编码器处理，而编码一帧图像本身可能需要消耗8ms时间，于是第一个像素实际上需要等待16ms以上才能真正进入信道传输环节。与123us的理论极限相比，效率相差100倍以上的。

数据对齐 Alignment

ISP、CODEC等硬件单元在处理图像时通常都会有粒度（granularity）要求，即必须将8/16/32/64/128个像素作为一组来处理，这样就可以通过硬件并行化来提高吞吐率。这个需求称为ISP的数据对齐（alignment）需求，多数sensor都支持一个linesize属性，以保证sensor输出的每行数据的宽度符合ISP的对齐要求。与linesize等价的术语还有stride，pitch，均表示每行数据所占用的实际存储空间。当linesize与图像的实际分辨率不相等时，sensor会用一个数据填充对齐部分，这个数据可以是固定值0（zero-padding），也可以是本行最后一个像素的值（copy-padding）。

在使用通用CPU处理图像时也广泛存在类似的概念。如Intel架构支持的MMX/SSE指令，ARM架构支持的NEON指令，以及Ceva，Cadence等DSP架构支持的Vector指令，都是基于SIMD（Single Instruction Multiple Data）原理，即单个指令并行处理多个数据，从而获得几十倍以上的效率提升。另外，在DSP架构中有时会设计一个Predicate寄存器，当某个指令（比如向量加法）引用了Predicate时，Predicate寄存器的每个bit决定对应的向量lane是否保存计算结果。一个常见的例子是，某vector指令支持32个操作数，但其中只有前8个是有效数据，输出buffer也只预留了8个数据的位置，相邻内存不允许踩踏。尽管DSP在执行指令时必须同时操作32个数据，但计算完成后只能允许前8个数据写入内存。此时需要将Predicate寄存器的对应bit置1，其余bit置0，以通知哪些数据是允许保存的。

图像撕裂 Image Tearing

在离线模式下，从sensor出来的数据需要先进入内存然后再被ISP处理。

在任何模式下，从ISP输出的数据通常都需要先写入内存（ring buffer）等待下级模块处理。

数据写入内存前，会先进入DMA硬件模块的缓冲，当DMA模块申请到总线使用权后才能真正写入存储器。于是问题就出现了，当系统非常繁忙的时候，DMA可能长时间未能申请到总线，而sensor数据却通过专用通道源源不断地进入DMA输入缓冲，当DMA缓冲溢出时就会发生数据踩踏事件，未能进入缓冲的数据就丢失了。结果就是图像撕裂、错位，即类似下图所示的情况。



数据踩踏不仅会破坏这一帧图像，根据系统设计还有可能影响下一帧图像。有的系统判断一帧图像完成的指标是DMA输出计数，当DMA输出足够数量的像素（比如1920x1080）时认为当前图像传输完毕，此时会发出中断信号。于是问题来了，当已经发生数据踩踏问题时，DMA输出像素的数量实际上小于应有的数量，于是DMA会一直等到下一帧图像（N+1）到来，把（N+1）图像的起始部分计入当前帧才能结束当前帧的工作循环，而下一个工作循环需要从检测到FrameStart信号开始，结果就是（N+1）图像的后续部分丢失，直到（N+2）图像的FrameStart到来后系统才能恢复正常。





实际上，踩踏事件的真正原因是系统设计不合理，系统的峰值负荷超过了系统的最大承载能力，因此这个现象只要存在就必然是持续存在。只有把峰值负荷调整下来后才能真正解决问题。调整系统负荷的一个简单办法是降低输入帧率或分辨率，也可以考虑在总线仲裁策略上加以调整，对带宽大户加以限制，甚至可以在软件算法上加以调整，将大的操作分解称若干个小的操作，中间留出一定的空隙给其它任务。这些策略有点像那个买房的笑话，说某国住房资源紧张，房价很高，于是国家鼓励错峰买房，有些人这辈子买房，有些人下辈子买房。。。

WDR图像接入ISP的方式

下图是在实验室中评估camera动态范围的一种常用方法，三个灯箱分别提供低、中、高三种亮度，通过精确控制高亮和低亮的照度比值可以计算出camera的实际动态范围。



由于材料和工艺的限制，普通的sensor 一般可以提供50~70dB的动态范围，比较优秀的技术（SONY）可以提供80dB的动态范围。但是室外应用经常需要捕捉100~110dB的动态范围，因此人们非常希望sensor 能够提供120dB以上的动态范围。笔者在这篇相关作品中分析过，

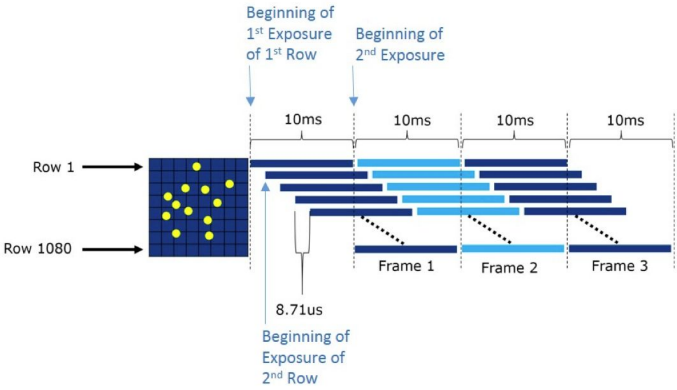
单纯增大像素面积以提高sensor动态范围会遇到成本和技术瓶颈，目前能够落地的80dB其实已经到达了这条路线的极限。想要进一步提高sensor动态范围只能另辟蹊径，于是人们转向了将多帧不同曝光的图像融合成一帧宽动态图像的技术，基本上解决了宽动态的需求。

最常用的宽动态是两帧融合，一帧短曝光图像重点采集亮部信息，一帧长曝光图像重点采集暗部信息，两帧图像同时输入ISP，经过一定的算法处理后生成一帧输出图像，能够同时还原亮部和暗部信息。算法对图像进行融合的过程称为frame stitching，或者称为WDR fusion，意思是将多个图像融合拼合为一体。

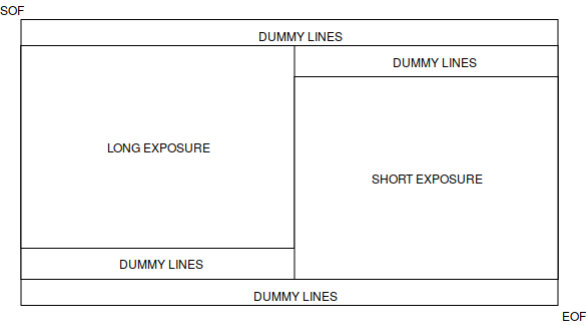
如果希望camera输出帧率保持30fps不变，则两帧融合WDR需要sensor输出60fps，而三帧融合WDR需要sensor输出90fps，四帧融合WDR需要sensor输出120fps。这也是为什么主流产品以两帧合成WDR为主的原因。

同样是两帧合成一帧，也可以有不同的实现方法。传统的做法是sensor先后输出两帧图像到内存中，两帧图像全部就位后再交给ISP进行处理。由于前后两帧时间的间隔比较大，图像中如果有运动的物体就会产生一些问题，如下图所示。这种问题叫做motion artifacts，目前基本没有太好的处理办法。

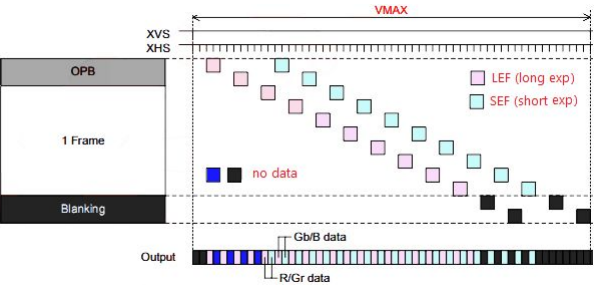
如果sensor 不再以帧为单位输出，而是以行位单位输出，则可以缓解运动伪影问题。如下图所示，sensor 先输出一行长曝光像素，再输出一行短曝光像素，然后开始输出下一行。当最后一个像素扫描完毕时，sensor已经完成了两帧图像的输出。三/四帧合成WDR也是同理。



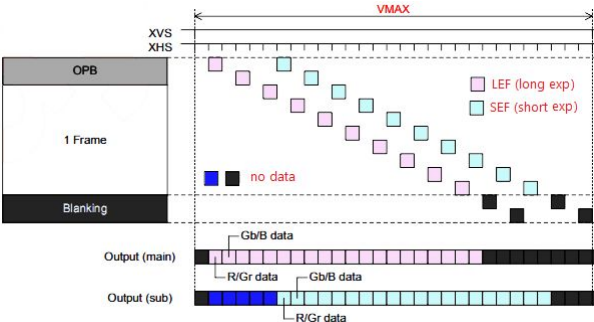
这种输出方式被OmniVision 称为Staggered WDR技术，下面是OV Staggered WDR sensor 的帧结构示意图，



SONY支持WDR的方式称为DOL(Digital OverLap)技术，最多支持三帧曝光。与Staggered WDR 技术不同点在于，它的输出格式种使用了一些特别的标志数据，所以需要专用的逻辑电路进行解析。它支持两种像素输出方式，方式1是使用1个stream输出，每行由长曝光+短曝光行交替，方式2是使用两个stream并行输出，如下图所示。



SONY DOL - single stream 交替输出方式



SONY DOL - main/sub stream 并行输出方式

仔细观察SONY和OV的原理图，很容易注意到长、短行是错开一定间隔的，即最开始出来的数据全部是长行+dummy，最后出来的几行全部是dummy+短行。

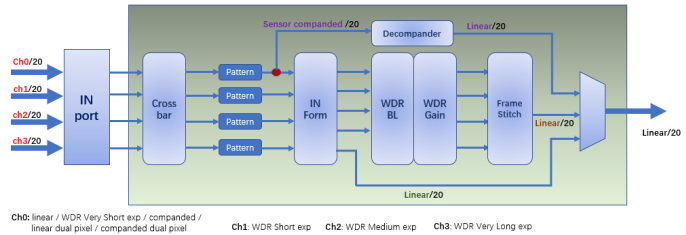
不难理解，ISP 做长、短曝光融合时，必须要同时取走相同行号、相同坐标位置的一对像素数据。因此，在sensor 端错开输出的行数据必须在进入ISP 之前重新对齐 (re-align)。通常的做法是在ISP前端设计若干行line buffer 缓存长行数据，等短行数据也到达后，ISP从长行line buffer中取长曝光像素，同时从短行line buffer中取实时到达的短曝光像素。

第二个问题是缓存长行的line buffer设计成多大比较合理。从SONY的原理图上看，预留8行line buffer 似乎应该够用了。如果未来出了一款sensor需要16行line buffer，则很不幸就不能支持在线模式了，只能先把长、短数据写入内存，然后同时从内存读取同一行号的长、短数据。



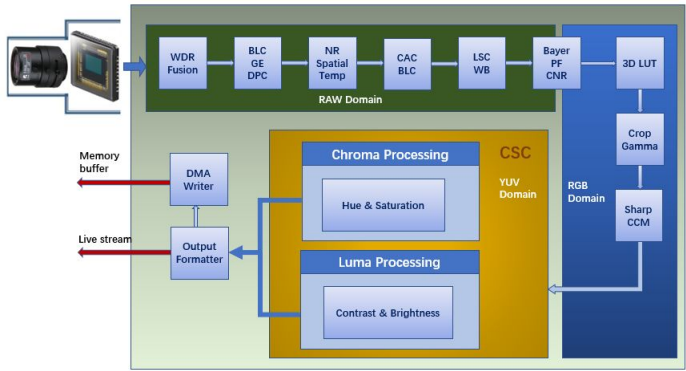
第三个问题是数据踩踏。对于离线的场景，ISP需要通过DMA从内存中读入RAW数据。如果系统总线特别繁忙，DMA可能出现长时间无法获得总线使用权的情况。这种情况在芯片同时运行神经网络算法时非常容易出现。一般ISP会允许5000个时钟周期左右的等待时间，如果超过这个时间，ISP就会产生时序错误，此时ISP会发出一个硬件中断，请求软件干预。

下面是一个主流ISP方案对WDR输入的处理流程图。除了正常的线性输入模式外，该ISP还支持最多四帧合成WDR，其中ch0支持线性输入和WDR的最短帧，ch1/2/3支持WDR的短/中/长帧。



该ISP还支持companded和dual pixel数据格式，这是因为某些sensor输出的不是线性数据，而是经过函数变换的数据，或者把两个像素的值合并到一个16位数据中输出，ISP也可以处理这些比较特殊的输入格式。

下面是该ISP的完整流程图，除了WDR功能之外的亮点是增加了空域和时域降噪模块，增加了颜色处理（PurpleFringing/PF, ColorNoiseReduction/CNR）和3D颜色变换表（LUT），可以实现更好的色彩还原效果，以及应对高挑战的低光夜视场景。



图像压缩

4K分辨率的图像有800W个像素，RAW格式占1600W字节，YUV422格式占1200W字节。如果客官对1200W没啥概念，可以想想在上海陆家嘴买一套135平大三房需要多少钱，差不多就是这么大的压力。其实还没完，想想8K分辨率，也就是7680x4320，RAW格式占6600W字节，这个压力就大得不敢想象了，估计汤臣一品大平层也差不多够了。

好了，想必客官已经清楚在内存中将这么大的数据搬来搬去不是一件很容易的事情了。为了减轻传输带宽和存储的压力，支持4K 以上的芯片都会在DMA上设计一个压缩算法。当DMA向内存中写入数据时，实际进入内存的是压缩后的数据。当DMA从内存中读取数据时，用户得到的是解压缩后的数据。

Arm 出售的图像压缩技术叫AFBC，即Arm Frame Buffer Compression，这是一种基于脉冲编码调制(Pulse Code Modulation, PCM) 技术实现的无损压缩技术，典型情况下可以实现50% 左右的压缩率，可以节省存储空间和传输带宽。有情报显示，华为在其手机芯片中实现了基于小波变换的有损压缩技术，压缩效率应该更高。

在工程实践种还存在另外一种做法，也是基于PCM技术，但压缩后的数据不是紧密排列在一起，而是以块为单位，压缩数据仍放在自己所在的块内，于是两个块之间会存在一些空洞。这样做的好处是可以方便地找到每一个块的数据，弊端是不能够节省存储空间，但仍然可以起到节省带宽的效果。

色调映射 Tone Mapping

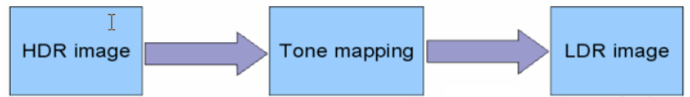
摄像机拍摄室外场景时，晴朗夏天的光照度可以达到10万~20万lux，理论上拍摄这种场景需要提供高达5000:1的动态范围，在摄像机内部则需要使用至少13位的数据才能表示5000:1的动态范围，在通用CPU架构中使用16位整数则更加方便。由于数据在处理环节经常涉及除法、开方、指数等浮点运算，所以还需要预留若干个小数位以保持浮点精度，4位二进制小数可以提供0.0625精度，8位二进制小数可以提供0.0039精度。上述的主流ISP方案中使用20位数据。

当图像在显示设备上输出时，普通的LDR显示器只能提供256级灰度，按数量级是100:1的动态范围。符合HDR10标准的显示器可以提供1000:1的动态范围，已经可以较好地还原自然场景的动态。如果摄像机的适配输出设备是LDR显示器，则摄像机的ISP内部需要完成从5000:1到100:1的动态范围压缩。

当WDR模块完成多帧合成（frame stitch）后，接下来就需要对数据位宽进行压缩以节约后续步骤的计算资源。比较合理的做法是采取逐级压缩策略，比如在WDR模块先压缩到12位精度，经过CCM、Gamma等颜色处理后进一步压缩到10位精度，经过CSC模块后进行最后一次压缩得到最终的8位精度输出。



从16/20位精度压缩到12位精度的过程称为色调映射，这一步骤的主要任务是压缩图像的动态范围，将HDR图像映射到LDR图像，并尽量保证图像细节不损失。



色调映射的方法大致分为两类，即全局算法和局部算法。

全局算法 (Global Tone Mapping, GTM)

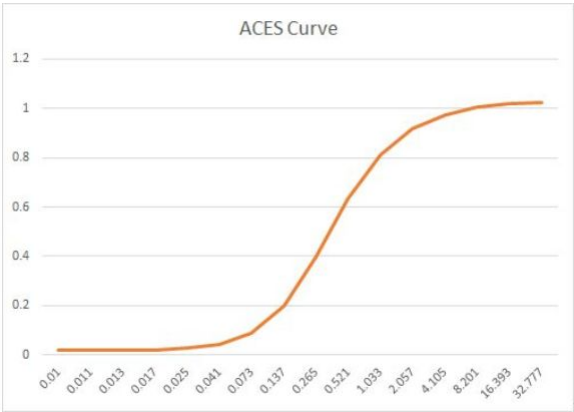
全局算法可以理解为每幅图像有一个颜色映射表，GTM算法通过查表的方法把一个输入颜色映射为一个输出颜色。有些算法对所有图像都使用固定的表，有些算法则是针对每一帧图像创建不同的表。

GTM算法特点：

- 1. 任意相同颜色的像素点，在映射后，还是相同的颜色；
- 2. 全局算法一般较简单，速度快；
- 3. 全局算法的性能一般劣于局部方法；

存在算法：直方图均衡化、Gamma、对数校正、直方图规定化、分段灰度变换

一种基于Gamma的算法叫做Academy Color Encoding System (ACES)，它由美国电影艺术与科学学会提出，有人认为这是目前最好的一种GTM算法。ACES本质上是一个通用的数据交换格式，既可以把不同的输入设备转成ACES，也可以把ACES在不同的显示设备上正确地显示。不管是LDR还是HDR都可以在ACES里表达出来。ACES的转换曲线如下图所示。



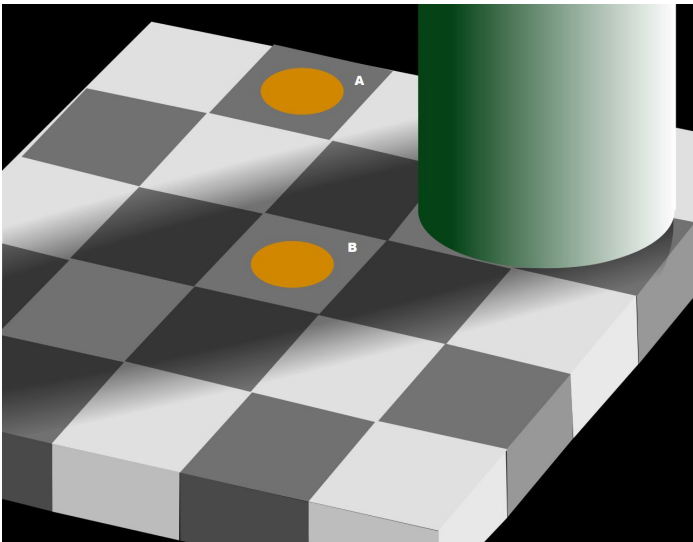
代码也很简洁。

```
1 float3 ACES ToneMapping(float3 color, float adapted_lum)
2 {
3     const float A = 2.51f;
4     const float B = 0.03f;
5     const float C = 2.43f;
6     const float D = 0.59f;
7     const float E = 0.14f;
8
9     color *= adapted_lum;
10    return (color * (A * color + B)) / (color * (C * color + D) + E);
11 }
```

局部算法 (Local Tone Mapping, LTM)

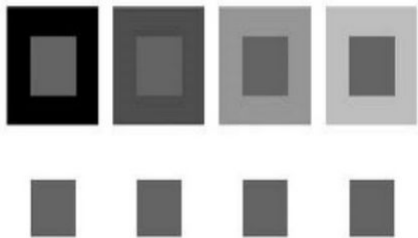
人眼的感知特性具有局部性特点，举例来说，在普通人眼看来下图中的A色块的亮度明显低于B色块的亮度。而事实真相是，这两个色块的真实像素亮度是一模一样的，人眼感觉B更亮，主要是因为B的周围是暗色块，而A的周围是亮色块，人的知觉系统针对这种场景自动做了对比度提升。





这种现象在总体上可以归结为人的知觉恒常特性，具体地说就是亮度恒常特性。下图是另外一个亮度恒常的例子。在这个例子中，所有小灰色块的真实亮度都是一样的，但是如果周围环绕的颜色不同，人对灰色块的颜色知觉也不相同。

Brightness constancy



局部算法借鉴了人眼的知觉原理，在映射一个像素时，不仅考虑该像素的绝对值，还会考虑该像素周围区域的平均亮度值，将对比度大的像素映射为高亮，对比度小的像素映射为低亮，往往可以取得更好的效果。

LTM算法特点：

- 1.映射前颜色相同的像素点，映射后颜色可能不同
- 2.局部算法一般较全局方法更复杂，速度相对较慢；
- 3.局部算法的性能一般优于全局方法；
- 4.会出现光晕等现象

存在算法： 分块中值直方图，基于Retinex原理的算法等

关于色调映射的更多内容可参考主题文章

RAW域处理

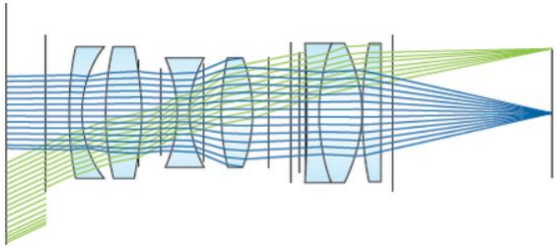
镜头阴影校正 LSC

镜头阴影有两种表现形式，分别是

- Luma shading， 又称vignetting， 指由于镜头通光量从中心向边缘逐渐衰减导致画面边缘亮度变暗的现象。
- Chroma shading， 指由于镜头对不同波长的光线折射率不同引起焦平面位置分离导致图像出现伪彩的现象。

Vignetting原理

- 1. 画面边缘镜头能量衰减

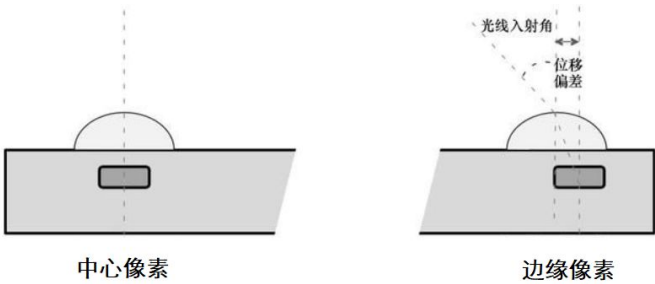


如上图所示，由于镜头中都会存在多处光阑，当入射光线偏离光轴角度较大时，部分光线就会被光阑遮挡而不能参与成像，因此越靠近sensor边缘的像素接收到的曝光量就越低。

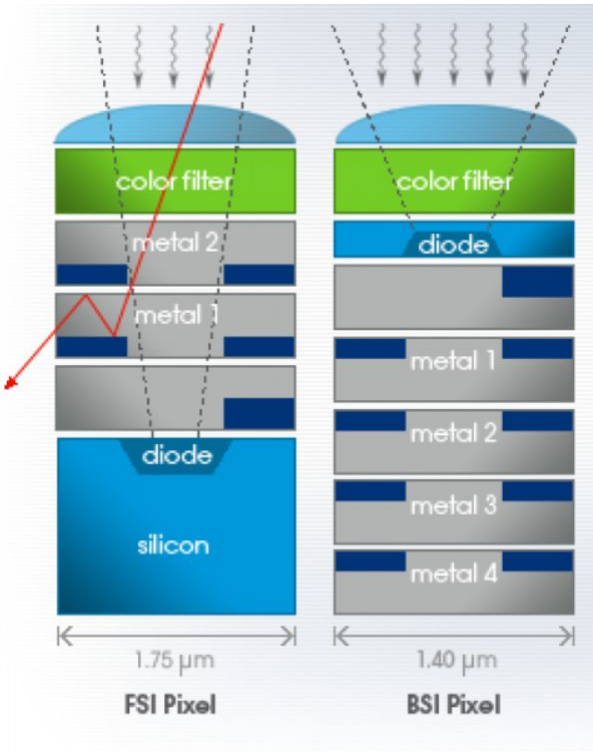
2. 边缘像素微透镜和感光面的错位

焦点错位导致能量损失

这个问题在手机sensor 上通常会更严重一些，因此设计手机sensor 的厂家会采取一些特定的方法去缓解这个问题。一种常用的方法是在微透镜上做文章，即从中心像素开始，微透镜的尺寸略小于感光面的面积一点点，这样越往边缘微透镜与感光面之间的错位就越大，刚好可以补偿入射光线角度增大导致的焦点偏移，使光线可以更好地聚焦到感光面上，如下图所示。



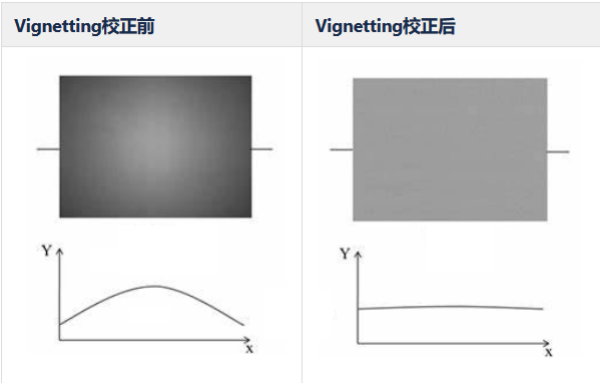
不过深入研究会发现，这个补偿办法其实也是有局限的，如果sensor采用的是下图左所示的FSI工艺（前照式），从像素微观结构来看，当入射光线角度比较大时，会有较多光线与像素中的金属布线层发生吸收、散射从而产生损失，单纯移动微透镜的位置并不能有效解决这个问题。但是，如果像素采用的是右图所示的BSI工艺（背照式），因为布线层在硅片的另外一侧，所以光线损失会少，补偿效果更加有效。





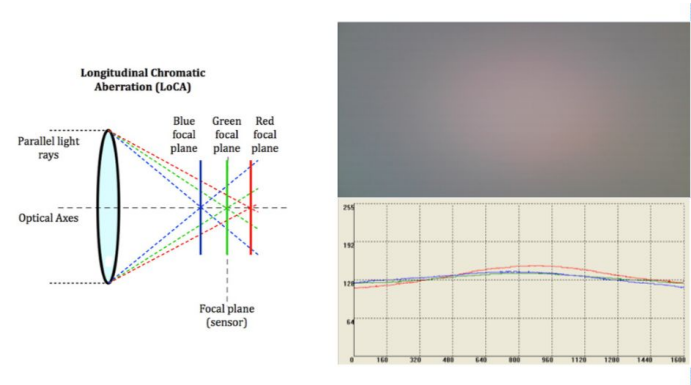
Vigetting是由镜头引起的现象，所以LSC校正也是针对特定镜头的。若果产品的适配镜头发生变化，原则上需要重新进行LSC校正。

另外，Vigetting现象在sensor靶面较大、镜头焦距较短时表现更加明显。采用非球面镜头通常可以改善vignetting。



Chroma shading 原理

镜头对不同波长的光线折射率不同会导致色差问题，即不同波长的焦点在空间上不重合，导致焦平面分裂为三个不完全重合的曲面，这会破坏图像的白平衡，使图像出现伪彩，如下图所示。根据sensor所处的前后位置不同，伪彩可能偏红也可能偏蓝。



Chroma shading 一般主要通过镜头选型来控制其影响。如果ISP支持chroma shading校正，则可以通过标定三个颜色平面的增益来修正。为了控制标定表格的存储空间，通常只标定MxN个关键点，任意位置处的像素增益可以使用相邻四个标定关键点通过双线性插值的方法动态计算得到。在一个典型的ISP实现中，可以取M=N=17即可得到令人满意的效果。

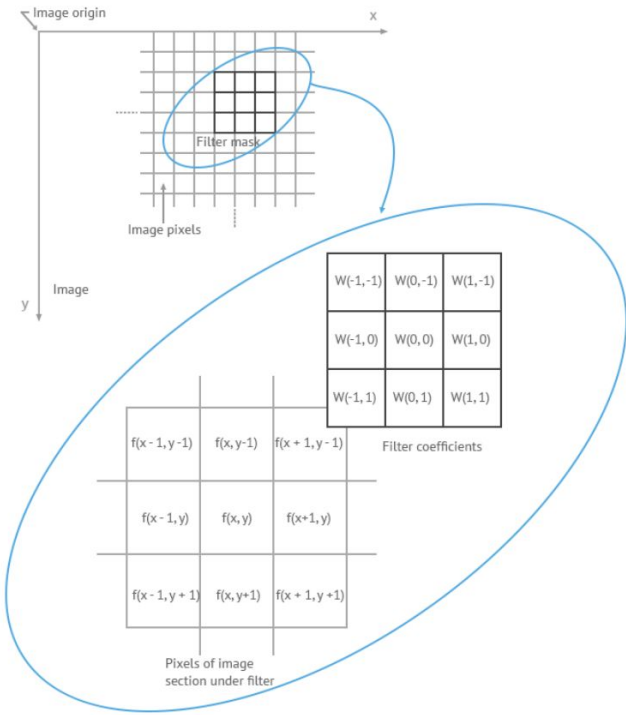
空域滤波器 (Spatial Filter)

对图像降噪的主要方法是使用空域滤波器对图像进行滤波。滤波操作通常是针对以某个像素为中心的滤波窗口上进行的，滤波窗口的大小与具体的算法有关，常用的大小有3x3、5x5、7x7等尺寸。滤波操作在数学上称为卷积，需要使用一个与滤波窗口大小一致的卷积核，卷积核的每个元素代表一个权重，与对应位置的图像像素值相乘，然后所有乘积累加到一起就是滤波后的结果。

卷积滤波用公式表示是，

$$g(x,y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s,t)f(x+s,y+t)$$

下面是卷积滤波操作的示意图。



原始图像:

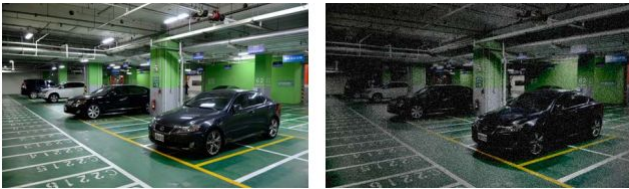
均值滤波结果:

中值滤波结果:

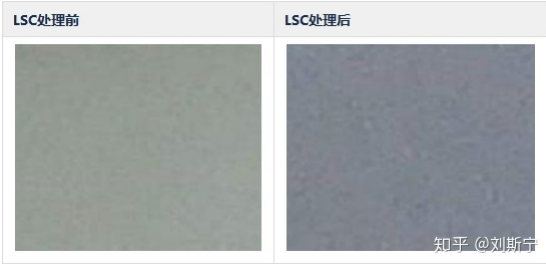
**RAW域降噪**

Sensor输出的RAW图像本身是携带了噪声的，前面提到过sensor噪声的种类主要包括热噪声、光散粒噪声、读出噪声、固定模式噪声等。当sensor温度较高、增益较大、环境较暗的情况下各种噪声会变得更加明显，成为影响图像质量的主要因素。在下图所示的例子中，左图光线较充足，图

像清晰无噪点，而右图中光照不足，camera 需要提高增益才能使画面达到正常亮度，同时也放大了噪声，图像出现明显的噪点。



除了Sensor图像本身携带的噪声之外，图像每次会经过ISP模块的处理之后都会引入一些新的噪声，或者对原有噪声进行了放大。以LSC模块为例，LSC校正的实质是在输入图像上乘以一个与像素位置有关的增益系数以补偿光信号的衰减，而补偿的规律是越远离图像中心的地方增益越大。根据噪声传播的基本原理，当增益系数大于1时，图像中的噪声是与信号一起被同步被放大的。另外，由于ISP所用乘法器的精度是有限的，每做一次乘法就会重新引入一次截断误差，这是新增的噪声来源，所以经LSC处理后图像的整体噪声水平会有所增加，而且在图像的边缘处表现会更加明显，典型的效果如下图所示。



Shading固然是不好的，需要校正，但是为了校正shading而给图像引入噪声同样也不好的，所以需要权衡在多大程度上校正shading能够收到满意的效果。这是在主观图像质量调试阶段需要考虑的问题之一。

研究发现，噪声在ISP流水线各模块中会不断产生、传播、放大、改变统计特性，对图像质量的影响会越来越大，而且越来越不容易控制。因此处理噪声的基本原则是越早越好，随时产生随时处理，尽可能将问题消灭在萌芽状态。目前主流的ISP产品中一般会选择在RAW域、RGB域、YUV域等多个环节设置降噪模块以控制不同类型和特性的噪声。在YUV域降噪的方法已经得到了广泛的研究并且出现了很多非常有效的算法，但是在RAW域进行降噪则因为RAW数据本身的一些特点而受到不少限制。主要的限制是RAW图像中相邻的像素点分别隶属于不同的颜色通道，所以相邻像素点之间的相关性较弱，不具备传统意义上的像素平滑性，所以很多基于灰度图像的降噪算法都不能直接使用。又因为RAW数据每个像素点只含有一个颜色通道的信息，所以很多针对彩色图像的降噪算法也不适用。

和很多图像处理算法一样，降噪即可以在空域(spatial domain)上实现，也可以在频域(frequency domain)上实现，比较有常用频域方法有傅里叶变换，离散余弦变换(DCT)，小波变换，多尺度几何分析等。随着人工智能技术的发展，近些年来还涌现了一批基于深度学习技术实现的降噪算法。后面提到的这些方法虽然都有不错的性能，但是对算力要求都比较高，并不一定适合处理高分辨率的实时视频流，所以在ISP产品中应用的并不广泛，目前适合ISP应用的降噪算法还是以经典低通滤波器的改进版本更为常见。

目前在RAW域降噪基本都需要将RAW图像按照颜色分成四个通道(R,Gr,Gb,B)，然后在各个通道上分别应用滤波器进行平滑，根据滤波器的特点和复杂度大致可以分成以下几类：

1. 经典低通滤波器，如均值滤波、中值滤波、高斯滤波、维纳滤波等。这类方法的优点是比较简单，占用资源少，速度快，缺点是滤波器是各向同性的，容易破坏图像中的边缘。另外由于没有考虑颜色通道之间的相关性所以也容易引入伪彩等噪声，而人眼对这种颜色噪声是比较敏感的。
2. 改进的经典滤波器，如Eplison滤波、双边滤波(bilateral filter)，在经典滤波器的基础上增加了阈值检测用于区分同类像素和异类像素，同类像素分配较大的滤波权重，异类像素则权重很小因而基本不参与滤波。这类方法的优点是可以有效地保护图像边缘，复杂度增加也不大，其它特点与经典滤波器基本相同。
3. 引导滤波器(guided filter)，由何凯明博士早期提出的一种算法，引入了引导图像的概念。对于任一颜色通道的图像，以当前位置像素 $P(x,y)$ 为中心，在一个固定大小的滤波窗口内为 $P(x,y)$ 的所有邻近像素 $\{P(i,j)\}$ 计算权重 $\{W(i,j)\}$ ，计算权重的方法是以某个引导图像作为参考，在引导图像的对应滤波窗口内，凡是与像素 $l(x,y)$ 性质“类似”的像素都得到较大的权重，与 $l(x,y)$ 性质“相反”的像素则得到较小的权重。举例来说，如果当前像素 $P(x,y)$ 是亮的，则滤波窗口 $\{P(i,j)\}$ 中的全部亮像素会参与平滑，平滑结果仍是亮的，同理，如果与 $P(x,y)$ 相邻的某个像素 $P(x',y')$ 是暗的，则对应的 $\{P(i',j')\}$ 中的全部暗像素会参与平滑，平滑结果仍是暗的，这样 $P(x,y)$ 与 $P(x',y')$ 之间的明暗边界就得到了保持。这种方法的基本假设是图像各通道的颜色梯度分布与引导图像是一致的，如果假设不成立，则从引导图像计算出的权重反而容易破坏其它通道中的边缘。有人建议使用demosaic还原出的G通道图像作为引导图像会得到较理想的效果，因为G通道集中了图像的大部分能量，也包含了丰富的梯度信息，是比较理想的候选对象。
4. 基于块匹配的滤波算法，利用图像的自相似特性，在以当前像素为中心的一个滤波窗口内找到与当前块最相似的几个块，当前像素的滤波值即等于几个相似块的中心像素的加权平均值。此类算法以非局部均值滤波(Non-Local Means)和BM3D(Block Matching 3D)算法为代表，它们的优点是平滑性能和边缘保持性能很好，缺点是计算量很大，资源消耗大，不太适合处理实时视频。已有情报显示，华为在某些型号手机芯片中实现了BM3D算法用于对照相图片做降噪处理，估计这主要是为了体现行业龙头企业占领技术高地的决心。





名称	权重图像																		
中值滤波	<div><table border="1"><tr><td>82</td><td>81</td><td>82</td></tr><tr><td>81</td><td>200</td><td>83</td></tr><tr><td>80</td><td>83</td><td>84</td></tr></table> <div>Ordered List 80 81 81 82 82 83 83 84 200 selected value</div> <table border="1"><tr><td>82</td><td>81</td><td>82</td></tr><tr><td>81</td><td>82</td><td>83</td></tr><tr><td>80</td><td>83</td><td>84</td></tr></table> Result</div>	82	81	82	81	200	83	80	83	84	82	81	82	81	82	83	80	83	84
82	81	82																	
81	200	83																	
80	83	84																	
82	81	82																	
81	82	83																	
80	83	84																	
均值滤波	<div><table border="1"><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></div>	1	1	1	1	1	1	1	1	1									
1	1	1																	
1	1	1																	
1	1	1																	

知乎 @刘斯宁

高斯滤波

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

知乎 @刘斯宁

Epsilon滤波

$$w(i,j,x,y) = |p(i,j) - p(x,y)| < \text{threshold} ? 1:0$$

9pixel

9pixel

center selected rejected

知乎 @刘斯宁

双边滤波

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_s^2} - \frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

output

input

filtering input p

知乎 @刘斯宁

引导滤波

$$q_i = a_k I_i + b_k, \forall i \in \omega_k.$$
$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}$$
$$b_k = \bar{p}_k - a_k \mu_k.$$

filtering input p

guide I

filtering output q

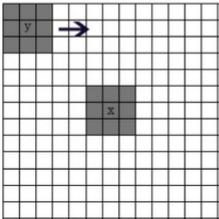
约束条件

局部线性模型

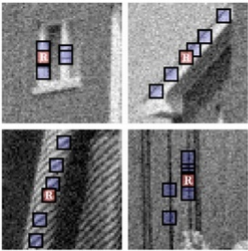
知乎 @刘斯宁



非局部均值滤波

$$w(x,y) = \frac{1}{Z(x)} \exp(-\frac{\|V(x) - V(y)\|^2}{h^2})$$
$$\|V(x) - V(y)\|^2 = \frac{1}{d^2} \sum_{\|z\|_\infty \leq d} \|v(x+z) - v(y+z)\|^2$$
$$Z(x) = \sum_y \exp(-\frac{\|V(x) - V(y)\|^2}{h^2})$$


BM3D



<https://www.cnblogs.com/whw19818/p/5765990.html>

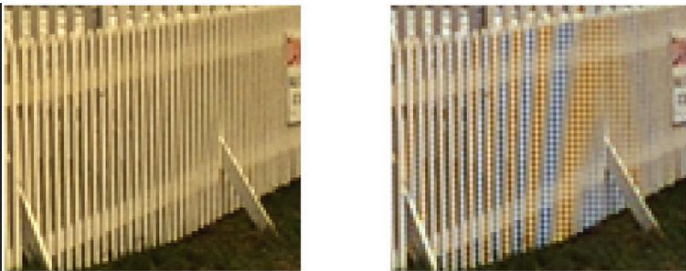
下图是双边滤波的效果示例，可以看到滤波和边缘保持的效果都是很理想的。



关于降噪算法的更多讨论可参考本专栏的另一篇主题文章。

Bayer Demosaic

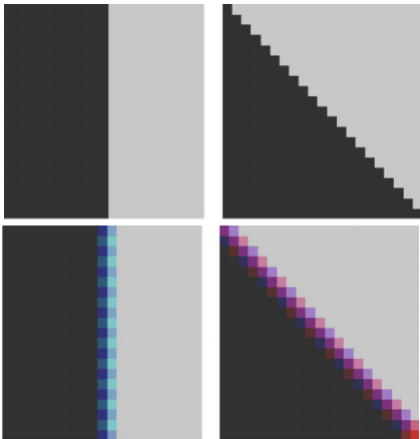
RAW域的最后一步处理是Demosaic，将像素从RAW域变换到RGB域进行下一阶段的处理。  
Demosaic 算法的主要难点在于，RAW域的任何一个个点（photosite）只包含一个真实的采样值，而构成像素（R,G,B）的其它两个值需要从周围像素点中预测得到。既然是预测，就一定会发生预测不准的情况，这是不可避免的，而预测不准会带来多种负面影响，包括拉链效应（zipper artifacts），边缘模糊，颜色误差等。



Demosaic 拉链效应和伪彩



Demosaic 伪彩



Demosaic 拉链效应，边缘模糊，伪彩

所以Demosaic 算法的主要挑战就是尽量提高算法的准确性，减少图像边缘损失和颜色误差。  
下面是引用一位知友的文章关于Demosaic的算法简介。

ISP Tuning

to be continued...

编辑于 2021-01-25 13:18

[图像信号处理器ISP \(Image Signal Processor\)](#)   [Camera](#)