SVC：scalable video coding 可伸缩视频编码

传统AVC，以图片链形式进行预测编码

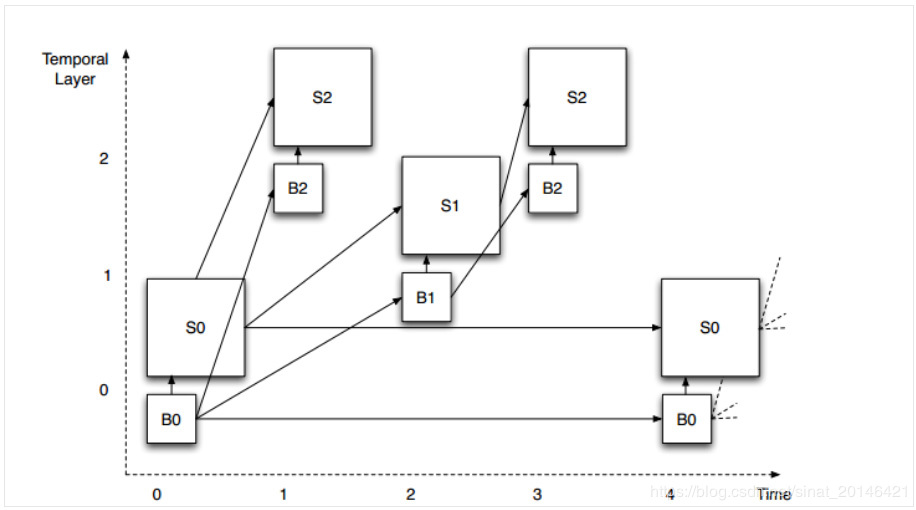
SVC将视频信号进行分层，根据需要传输所需的层来适应不同的终端和网络带宽

分层：

1. 时域分级，将视频中的图片分到不同的时间层中，低层编码不依赖高层。

可随时丢弃高层，以适应较低的帧率

1. 空域分级，将每一张图片分成不同的分辨率，如HME L0,L1,L2，先编码最低的分辨率B，再依次编码高分辨率S，根据需要选择不同的分辨率图层来使用



1. 质量分级 对一帧图片采用不同的量化步长来进行量化

VP9

VP9采用空间分级的方式，对于一帧图片，从低分辨率开始，分成三级图层分别进行编码，将编码后的三个帧统一存储为一帧超级帧。

Inter Prediction

--- compound prediction

\* Super frame:A super-frame is simply one or more non-displayable frames and one displayable frame all strung together as one chunk of data in the container.

\* internal references are all updated with the non-displayable frames

\* The reference frame is applied at 8x8 granularity

\* reference frame scaling. Each new inter frame can actually be coded using a different resolution than the previous frame. When creating inter predictions, the reference data is scaled up or down accordingly. The scaling filters are 16th pel accurate, 8-tap. This feature is allows for quick and seamless on-the-fly bitrate adjustment

--- 复合预测：

\*相当于双向预测，有两个MV，取平均值

\*三个参考帧每一个都选出两个MV

\*两个MV，

一个从使用相同参考帧的周围已预测块取MV，最多8个

一个从时间上的前一帧相同位置已预测块取MV

\*根据最终取的MV不同，MV可分为

- New MV,Nearest MV,Near MV,Zero MV

--- 参考帧列表：最多存储8帧。

\* 当前帧从列表中选择三帧作为last frame，golden frame，alt frame,来进行帧间预测。预测完成后更新参考帧列表。

\* 由任意帧组成的alt-ref frame 相当于B帧

AV1

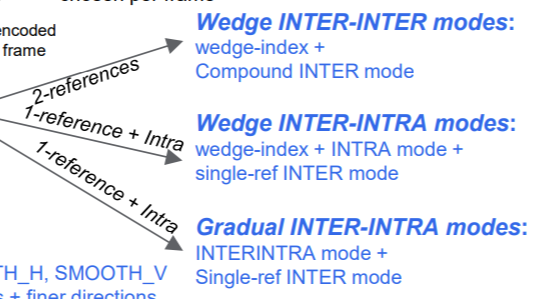
Inter-prediction

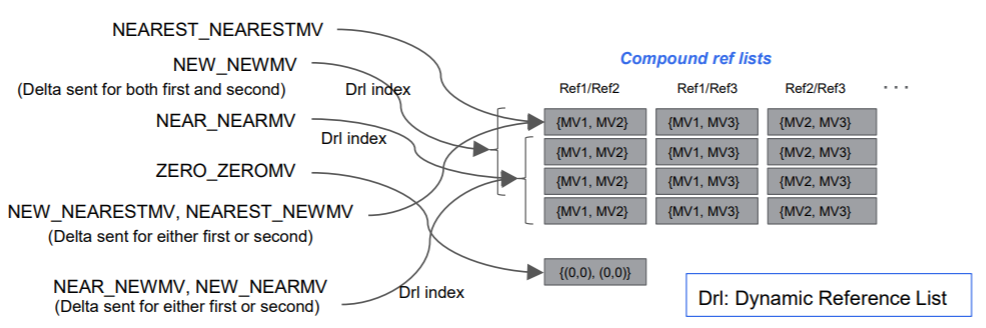
--- 参考帧列表：

\* 可从参考帧列表选7帧来进行帧间预测

--- 复合预测：

\* 每个参考帧选4个MV

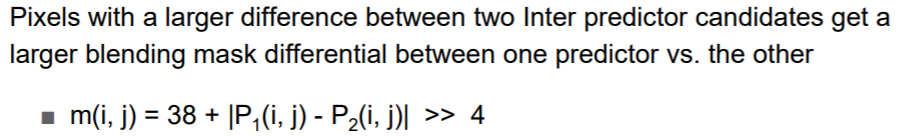
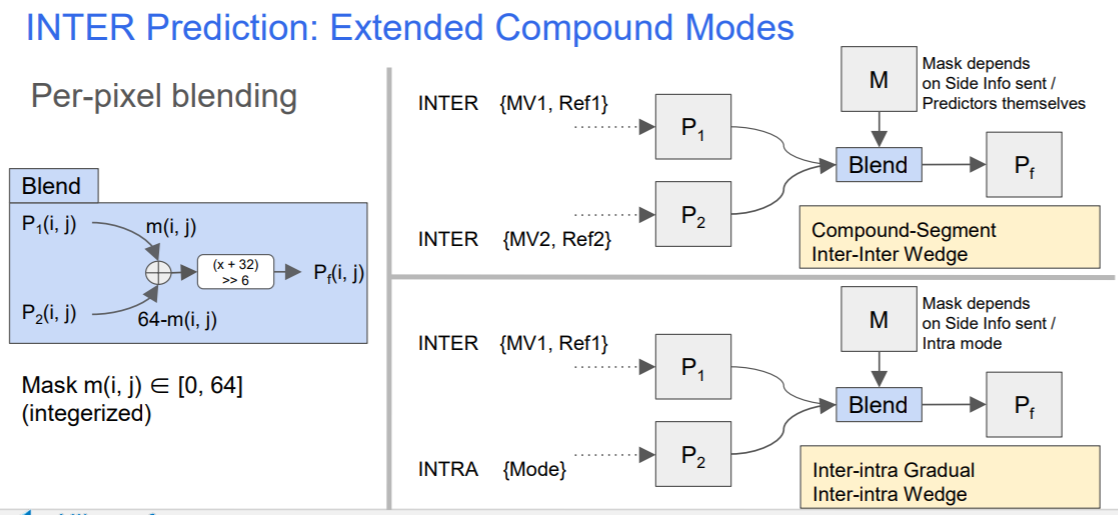




\* 新复合预测模式

--- Compund segment

每个参考帧所选出的MV之间进行复合

--- 帧间预测

\*选择 wedge/gradual inter-inter/intra 中的一种模式

- wedge inter-inter

\*wedge index

\*compound inter mode ：选两个参考帧进行MV选取与复合

- wedge inter-intra

\*wedge index

\*intra mode \*single inter mode

- gradual inter-intra

\*intra mode

\*single inter mode

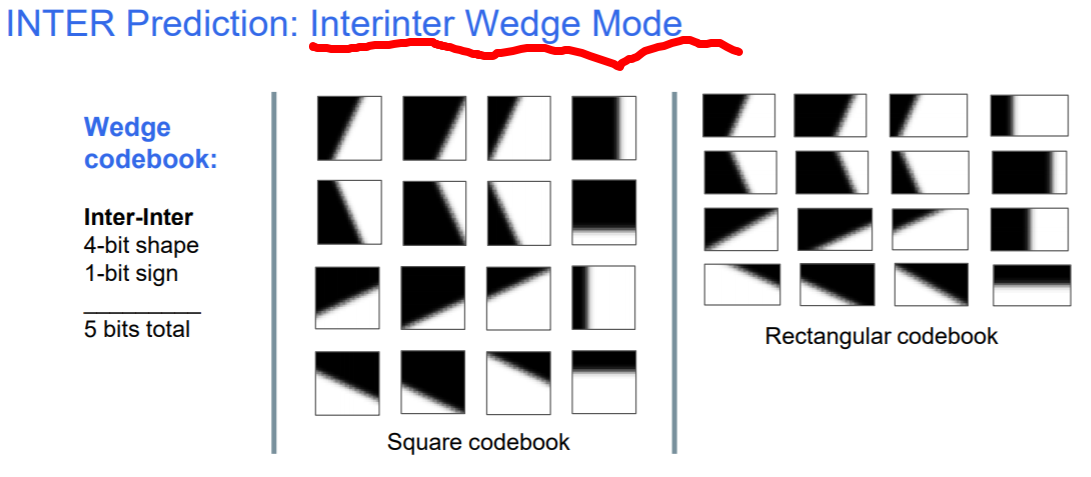
\*为每个参考帧选择 4个MV

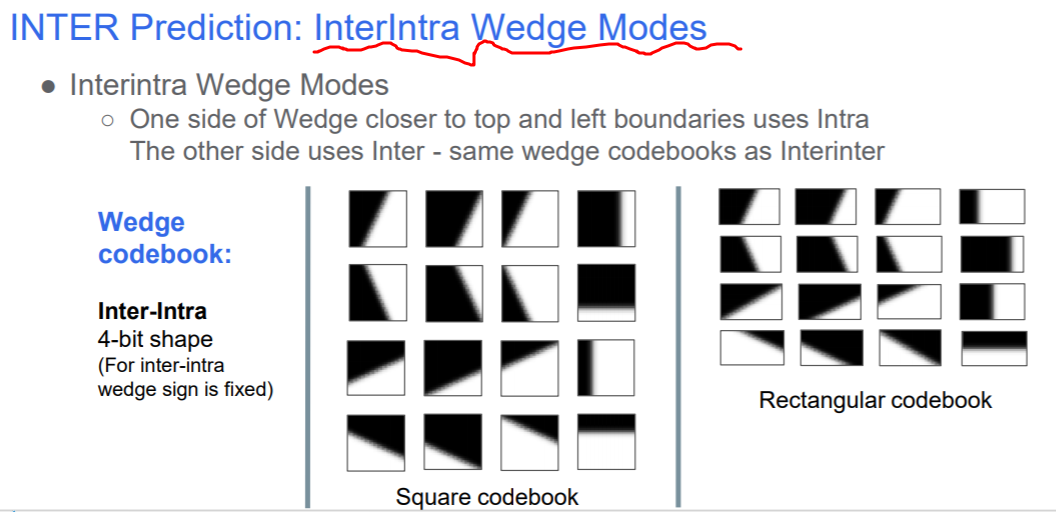
--- 为解决帧间预测分块无法完全贴合物体边界的问题

\* Wedge mask codebook :表示各种block在边界上的情况

\*对于block，inter和intra各自进行预测，

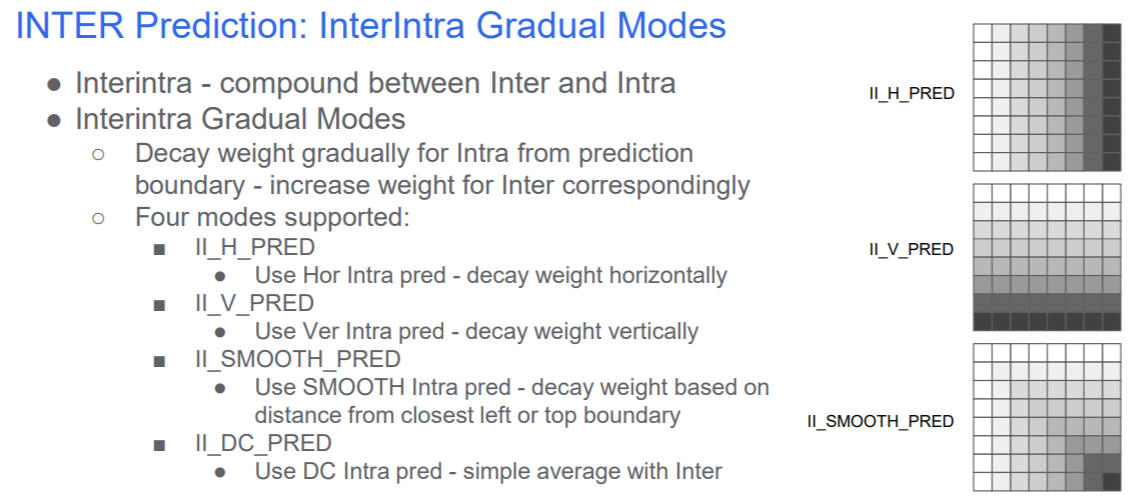
对于预测结果，根据边缘形状不同，不同的像素点选取inter/intra的结果





\*gradual inter-intra

- intra 和 inter的复合：按照某种方式，逐渐降低intra权重，边界处提升inter权重



AV1-alt-ref temporal filter

原理：

均值滤波：对一个像素点，设置一个搜索框，对搜索框内的所有像素点取均值，即当前像素点所对应的灰度值。

非均值滤波：在对搜索框内所有点设置不同的权重值，在求和取均值

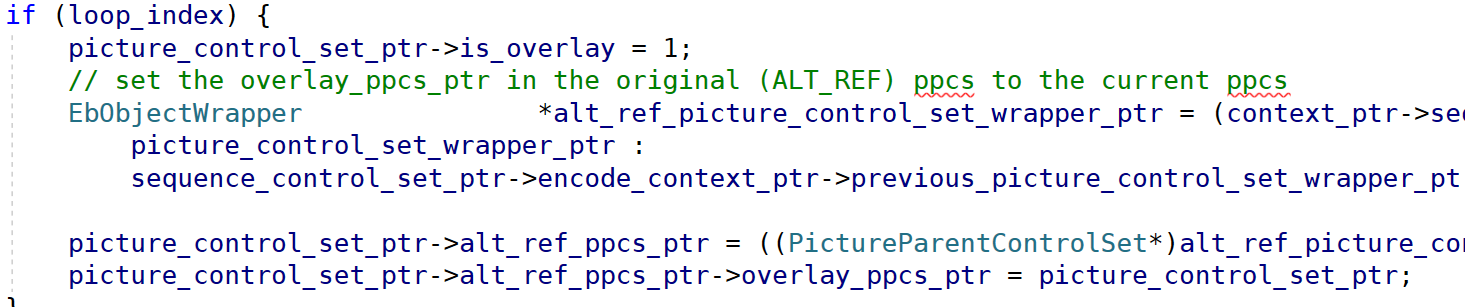
NL Mean: 根据相似性取权重值。

以当前块与参考块之间的欧氏距离作为相似性的依据，来求权重值。

距离越近，相似度越高，权重值就越大。

1. Resource coordinate kernel

初始化时，默认每一帧都是alt-ref。初始化overlay\_frame

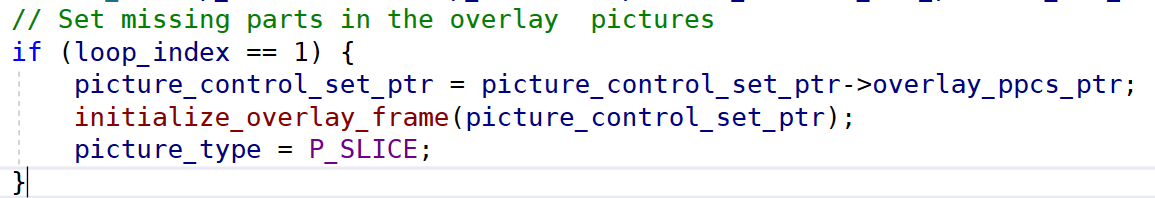


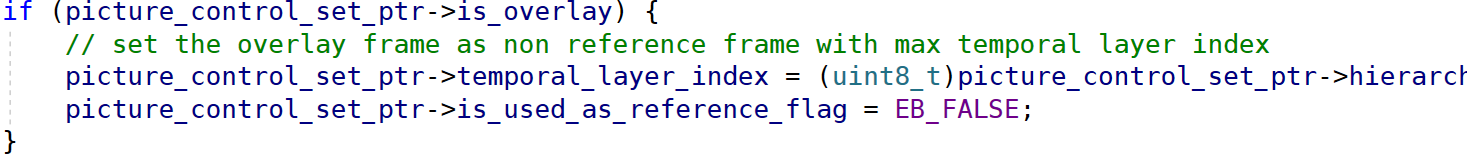
1. picture dicision kernel

到了picture decision kernel时，知道了predict structure，确定alt-ref位置。

满足以下条件的帧，均有alt-ref

predPositionPtr->temporal\_layer\_index == 0 && picture\_type != I\_SLICE





其他帧，释放overlay-frame

开启ME filter，发送object =encode\_context\_ptr->pre\_assignment\_buffer[pictureIndex]

将当前帧送出，并锁上。

eb\_block\_on\_semaphore(picture\_control\_set\_ptr->temp\_filt\_done\_semaphore)

1. Motion estimate kernel

当task\_type=1时，进行temporal filter，调用init\_temporal\_filtering

1. init\_temporal\_filtering

以temp\_list的每一帧为参考帧，进行帧间预测，得到残差

根据残差计算temporal filter权重，修正像素值。

累加temp\_list所有帧的修正值，做归一化。

保存于：

Alt\_ref\_buffer = buffer\_y,直接修改了图片的像素值

picture\_control\_set\_ptr\_central->filtered\_sse

picture\_control\_set\_ptr\_central->filtered\_sse\_uv

送出当前帧的temp\_list完成filter的信号

eb\_post\_semaphore(picture\_control\_set\_ptr\_central->temp\_filt\_done\_semaphore);

1. picture dicision kernel

收到信号，解锁，继续向下执行。

1. EbRateControlProcess.c

根据filtered\_sse,filtered\_sse\_uv的强度，调节帧的QP

\*将temporal filter结果应用在motion analysis

\* alt-ref帧是否进入了reference list

AV1 kernels

Picture manager kernel：对input进行处理，post给其他kernel进行其他处理

对input的ref-list进行检查，其所有参考帧是否准备好

其中，检查条件的各种flag处理在前面各种kernel中完成

如果准备好，创建CPCS，并post result。

\*根据inputPictureDemuxPtr->picture\_type不同，进行不同的处理

1.EB\_PIC\_INPUT: resource-coordinate-kernel处理后的帧

1.1将当前帧加入manager-queue，按照display-order存。

1.2遍历manager-queue

1.2.1 将当前帧加入input-queue，加入ref-queue

1.2.2 用当前帧在pred-struct中对应项内容填充进ref-queue中对应项

2.EB\_PIC\_REFERENCE:rest-kernel处理后的帧

2.1遍历ref-queue，寻找当前帧在ref-queue中的对应项

2.2 将对应项的ref-ptr指向当前帧的ref-ptr

2.3设置reference-available = true

3.EB\_PIC\_FEEDBACK:pack-kernel处理后的帧

3.1遍历ref-queue，寻找当前帧在ref-queue中的对应项

3.2设置frame\_context\_updated = EB\_TRUE

\*处理后，

1. 遍历input-queue
2. 当前帧在ref-queue中找到reflist0,1的对应项，计算delta-poc
3. 判断此对应项是否已完成所有处理，可以作为ref-frame使用

如果满足，设置availabilityFlag = true。

满足条件如下：？？？

1. 如果availabilityFlag = true，创建并填充ChildPCS

填充信息如下：

\*parentPCS公共信息

\*enc-dec-segments init

\*tile-info，entropy，rate-control

\*ref-arrays

1. Post给后续kernel

rateControlTasksPtr->task\_type = RC\_PICTURE\_MANAGER\_RESULT;

\*ref-frame检查条件，对应于哪些处理

Picture decision kernel：

1. 将输入帧加入decision-queue，只有原始帧能进入

\* 按照display order 放

1. 遍历decision-queue，只有首帧存在时才能进入

\*设置scene-change-flag

\*设置parent-window，WINDOW-AVAIL,FRAME-PASS

- 只有当填满window【0-7】时，W-A才为真

当前帧才能进入pre-assign-buffer

* 1. 将当前帧加入 pre-assign-buffer，只有当W-A为真时

2.2 当满足如下条件任一项时，创建GOP，遍历mini-gops

\*当pre-assign-buffer中有IDR/CRA帧时

更换pred-struct/scene change/更新ref-queue depend count等

\*当pre-assign-buffer中帧数满足pred-struct要求时

-为每一帧在pred-struct中找到对应帧，并从ref-queue选择其list0,1

中的对应参考帧

\*当pred-struct是延迟P/B帧时

只参考display order的前一帧

\*当eos-flag为真时 ：所有帧都已经进入seq

当不满足条件时，decision-queue-head后移一位

\*检查指针后移一位后的decision-queue中的帧，是否能填满window【0-7】

能填满，W-A为真，又一帧进入pre-assign-buffer。

直到满足count数，进入下一环。

当满足条件，遍历完pre-assign-buffer最后一帧

\*将pre-assign-buffer-count归0，等待新一轮的循环处理。

2.3 创建GOP

\*将pre-assign-buffer中的帧分成n个采用3/4层结构的gop

2.4 遍历GOP

2.4.1 第一遍遍历pre-assign-buffer

\*确定帧类型，当前帧的pre-struct-position

\*不满足生成alt-ref条件的，清空其复制帧，alt-ref=0

满足的，alt-ref=1

\*loop循环：

- loop0(alt-ref=0,1)：原始帧进ref-queue，并设置ME所需的参数

- loop1(alt-ref=1): 原始帧的复制帧，进一步填充参数，包括ME所需参数,但是不进ref-queue

\*需要的帧，生成alt-ref

- 建立temporal-filter-list，并post

- 触发motion-estimate-kernel 进行 temporal-filter

- 将当前帧替换为alt-ref。

- 替换完成之前，不想下执行

2.4.2 第二遍遍历pre-assign-buffer

\*每一帧在ref-queue中找到reflist0,1的对应项

计算delta-poc

填入当前帧的ref-pic/poc-array，为下一个ME准备。

\*IDR,CRA

\*RPS-INFO

Pred-structure-group

PredictionStructureConfig

- entry-count

- PredictionStructureConfigEntry \*entry\_array 每种结构均有一个数组，包括一个周期的每一帧的预测结构信息

PredictionStructureConfigEntry

- temporal layer

- ref list 0/1

- decode order

\* PredictionStructureCtor

-按照规则创建pred-struct

-pred-struct-array = leading + init + state

-参考steady-state，填充当前pred-struct的pred-struct-array中每一项的ref-list0,1

-参考已经填充的ref-list0，1，填充每一项的depend-list 0,1

-参考已经填充的depend-list 0,1，

填充当前pred-struct的timeline-map，decode-order-table

-参考已经填充的ref-list0,1，填充每一项的前向，后向个数，delta poc等

Enc-handle-ptr

->Sequence-control-set-instance【0】

-> sequence-constrol-set-ptr (only 1)

-> picture control set ptr

->encoder-contest-ptr