

KERNEL\_INFO kernelInfoConcat(LBP\_CASC\_K)

(

LBP\_CASC\_KN, //kernel 名字

9,

\_\_port(\_\_index(0),

\_\_identifier(LBP\_CASC\_IN\_INTEGR\_IMG), //定义输入积分图像的名称（无符号32位）

\_\_attributes(ACF\_ATTR\_VEC\_IN),

\_\_spatial\_dep(1, 0, 1, 0),

\_\_e0\_data\_type(d32u),

\_\_e0\_size(1, 1),

\_\_ek\_size(1, 1)),

\_\_port(\_\_index(1),

\_\_identifier(LBP\_CASC\_LINE\_IDX), //定义行索引缓冲区的名称（无符号16位）

\_\_attributes(ACF\_ATTR\_SCL\_OUT\_STATIC\_FIXED),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d16u),

\_\_e0\_size(1, 1),

\_\_ek\_size(1, 1)),

\_\_port(\_\_index(2),

\_\_identifier(LBP\_CASC\_WND\_BUF), //定义积分图像窗口缓冲区的名称（无符号32位）

\_\_attributes(ACF\_ATTR\_VEC\_OUT\_STATIC\_FIXED),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d32u),

\_\_e0\_size(1, 1),

\_\_ek\_size(MAX\_WINDOW\_WIDTH\_LBP + 1, 32)),

\_\_port(\_\_index(3),

\_\_identifier(LBP\_CASC\_IN\_CASC\_SZ\_AND\_SKIP), //定义特征和阶段数的名称+是否应跳过奇数行和列（3个16位值）（无符号16位）

\_\_attributes(ACF\_ATTR\_SCL\_IN\_STATIC\_FIXED),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d16u),

\_\_e0\_size(1, 1),

\_\_ek\_size(3, 1)),

\_\_port(\_\_index(4),

\_\_identifier(LBP\_CASC\_IN\_CASC\_FEAT), //定义Haar-like特征数组的名称（无符号8位）

\_\_attributes(ACF\_ATTR\_SCL\_IN\_STATIC\_FIXED),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d08u),

\_\_e0\_size(1, 1),

\_\_ek\_size(10000, 1)),

//\_\_ek\_size(4096, 1)),

\_\_port(\_\_index(5),

\_\_identifier(LBP\_CASC\_IN\_CASC\_STAGES), //定义级联级数的名称（无符号8位）

\_\_attributes(ACF\_ATTR\_SCL\_IN\_STATIC\_FIXED),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d08u),

\_\_e0\_size(1, 1),

\_\_ek\_size(200, 1)),

//\_\_ek\_size(64, 1)),

\_\_port(\_\_index(6),

\_\_identifier(LBP\_CASC\_IN\_PIX\_SHFT), //定义LUT的名称，其中包含水平像素偏移（无符号8位）所需的平铺移位

\_\_attributes(ACF\_ATTR\_SCL\_IN\_STATIC\_FIXED),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d08u),

\_\_e0\_size(1, 1),

\_\_ek\_size(64, 1)),

\_\_port(\_\_index(7),

\_\_identifier(LBP\_CASC\_IN\_PIX\_OFFS), //定义LUT的名称，其中包含水平像素偏移（无符号8位）所需的in-tile偏移

\_\_attributes(ACF\_ATTR\_SCL\_IN\_STATIC\_FIXED),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d08u),

\_\_e0\_size(1, 1),

\_\_ek\_size(64, 1)),

\_\_port(\_\_index(8),

\_\_identifier(LBP\_CASC\_OUT), //图像输出 （8位无符号）

\_\_attributes(ACF\_ATTR\_VEC\_OUT),

\_\_spatial\_dep(0, 0, 0, 0),

\_\_e0\_data\_type(d08u),

\_\_e0\_size(1, 1),

\_\_ek\_size(1, 1))

);

void lbp\_cascade(vec08u\* apOut, //指向目标缓冲区的指针

vec32u\* apInII, //指向源积分图像缓冲区的指针

int /\*aOutStride\*/, 目标数据的行距

int /\*aInStride\*/, 源数据的行距

int aTileWidth, 一个数据图块的宽度

int /\*aTileHeight\*/, 一个数据图块的宽高度

int16u aLineIndex, 当前处理行的索引

vec32u\* apWindowBuffer,指向输入积分图像窗口缓冲区的指针

const APEX\_lbpFeature\* apcFeatures,指向LBP特征数组的指针

int aStageCount, LBP特征级联阶段数

const APEX\_lbpStage\* apcStages,指向LBP特征级联阶段的指针

const int08u\* apcXshifts, 指向LUT的指针，该指针包含水平像素偏移所需的平铺移位

const int08u\* apcXoffsets,指向LUT的指针，该指针包含水平像素偏移所需的In-tile偏移

int skipOdd)如果应跳过偶数列和行，则为1，否则为0

const int STAGE\_FRACTIONAL\_BITS = 28

const int STAGE\_FIXED\_POINT\_MULTIPLIER = (1 << STAGE\_FRACTIONAL\_BITS)

该算法使用用户提供的局部二值模式（LBP）分类器搜索24x24像素的对象，对于每个输入像素，如果该像素是对象的左下角，则输出255，否则输出0。 分类器由多个阶段组成，每个阶段包含多个LBP特征。窗口必须经过每个级才能被分类为包含对象。

struct APEX\_lbpStage

{

uint8\_t featureCount; //其中featureCount是阶段中包含的许多LBP特征

int32s threshold; //是定点阈值（28个小数位），必须经过测试特征结果的总和才能通过该阶段。

};

LBP特征是细分为统一3x3网格的矩形。 将9个子区域中的每个子区域中的像素值相加。 将外部区域的总和与中心区域的总和进行比较，从而得到一个8位代码（从左上方区域开始并顺时针旋转）。 然后，此代码用作256位表的索引，结果位指示该特征是否通过。

struct APEX\_lbpFeature

{

int32\_t values[8];

int32s leafValuesFixed[2];

uint8\_t x, y, width, height;

};

values [8]是一个256位分类表，用于确定具有给定代码的LBP特征是否应通过，为了使特征通过，其中threshold是一个定点数（13个小数位），必须超过特征矩形的加权和。

leafValuesFixed是定点值（28个小数位），如果特征通过，则leafValuesFixed [0]添加到阶段总和；如果未通过，则将leafValuesFixed [1]添加到阶段总和。

x，y定义特征图左上角宽度，高度的位置，定义大小

内核期望完整的图像。 该实现为整体图像保留滚动窗口，因此必须提供足够大小的缓冲区。

汇总表计算（Sat）

KERNEL\_INFO kernelInfoConcat(SAT\_K)

(

SAT\_KN,

3,

\_\_port(\_\_index(0),

\_\_identifier(SAT\_KN\_IN),

\_\_attributes(ACF\_ATTR\_VEC\_IN),

\_\_spatial\_dep(0,0,0,0),

\_\_e0\_data\_type(d08u),

\_\_e0\_size(1, 1),

\_\_ek\_size(1, 1)),

\_\_port(\_\_index(1),

\_\_identifier(SAT\_KN\_OUT),

\_\_attributes(ACF\_ATTR\_VEC\_OUT),

\_\_spatial\_dep(0,0,0,0),

\_\_e0\_data\_type(d32u),

\_\_e0\_size(1, 1),

\_\_ek\_size(1, 1)),

//TODO: enforce height = 1?

\_\_port(\_\_index(2),

\_\_identifier(SAT\_KN\_OUT\_ROW), 定义上一个图块缓冲区的最后一行的名称

\_\_attributes(ACF\_ATTR\_VEC\_OUT\_STATIC),

\_\_spatial\_dep(0,0,0,0),

\_\_e0\_data\_type(d32u),

\_\_e0\_size(1, 1),

\_\_ek\_size(1, 1))

);

void sat32(vec32u\* apDest, //32bit输入的积分图

vec32u\* apPrevRow, //32位图块中的最后整数行。

const vec08u\* apcSrc, 8bit源块指针

int inStrideW, 源块宽度（以字节为单位），包括填充

int outStrideW, 目标块宽度（以字节为单位），包括填充

int bw, 一个数据区块的宽度

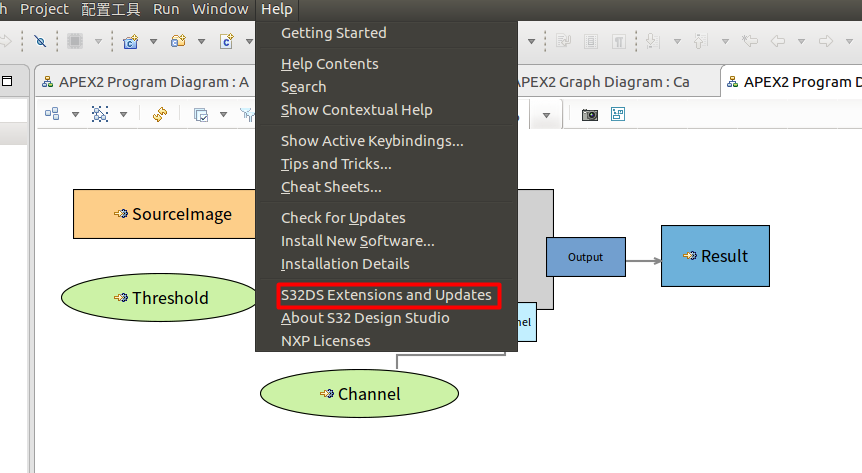
int bh, 一个数据区块的高度

int08u aFirstTile) 布尔值。 是，如果计算了第一个图块

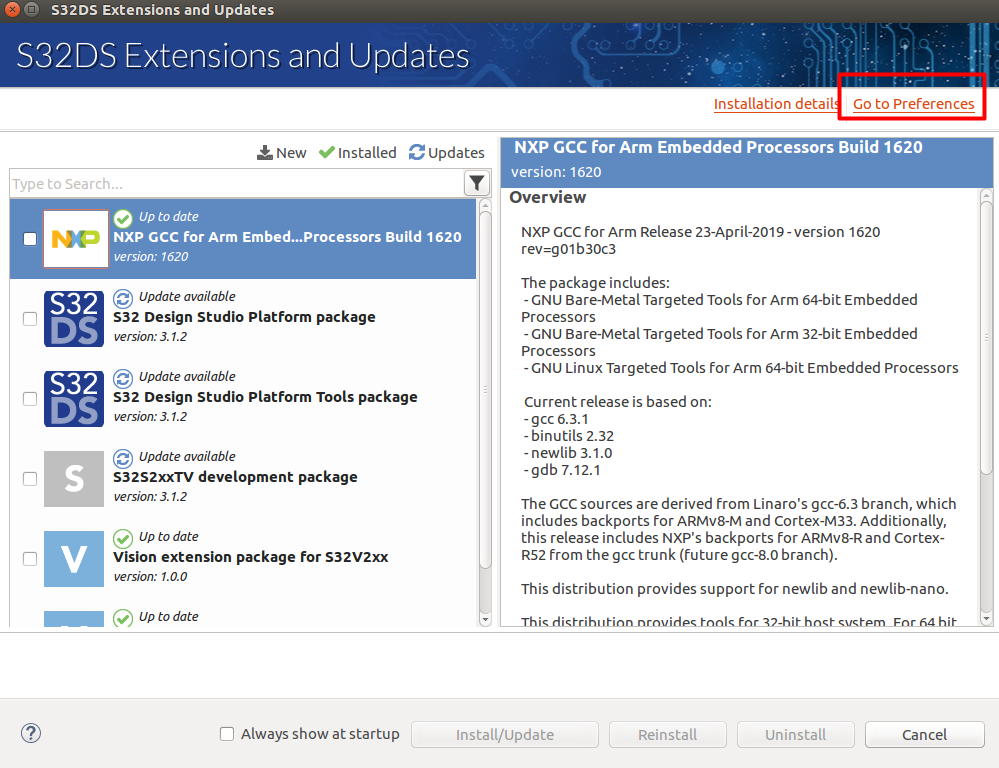


S32DS 离线更新

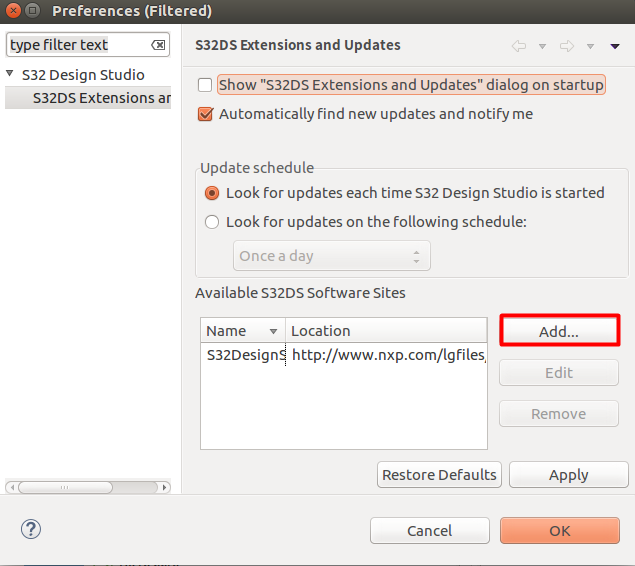
选择Help下面的S32DS Extensions and Updates

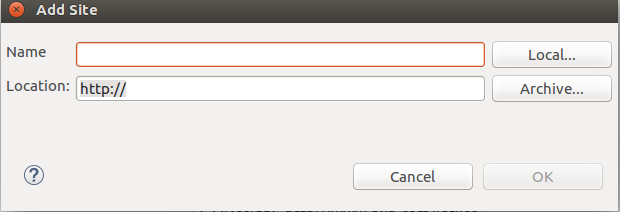


选择 go to Preferences

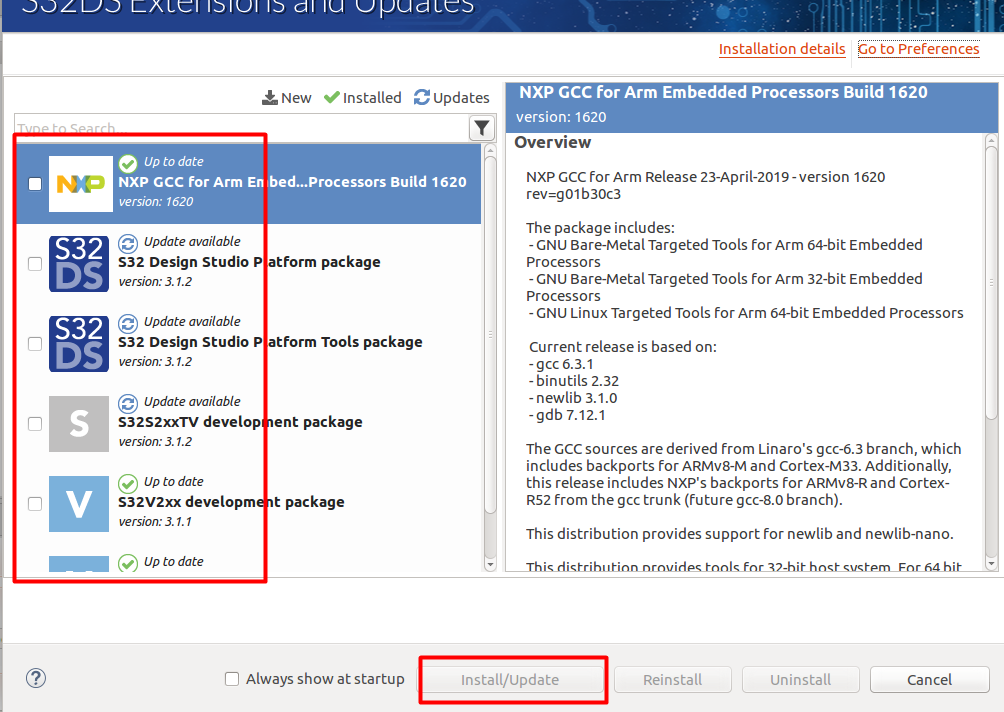


选择add添加离线下载Zip





点击ok



选择自己添加的更新就ok!