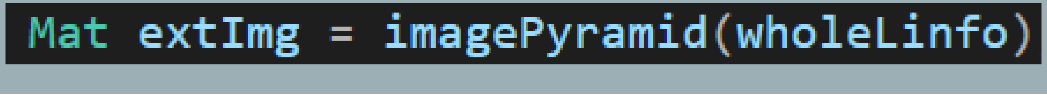
Modification for Synthesis & Co-sim Optimization

Feature Extraction

鄧文瑜

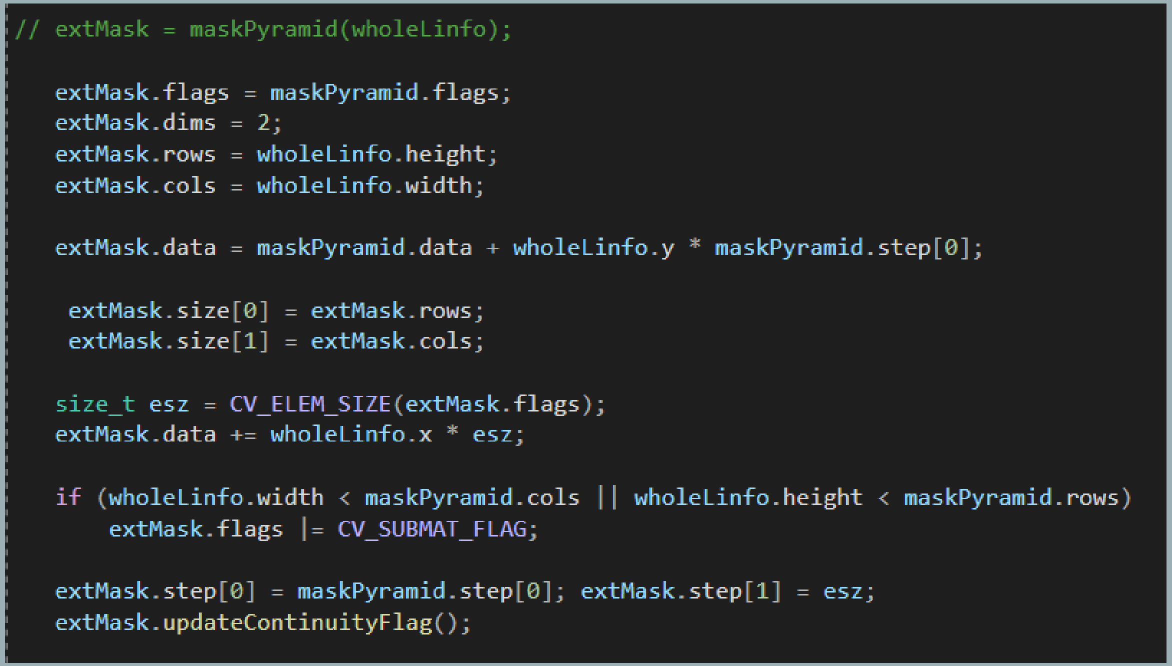
Pointer to pointer cannot be synthesized



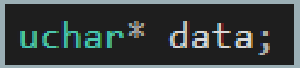
```
Mat extImg = imagePyramid(wholeLinfo)
```

用這種分式創建Mat的物件時，由於Mat的成員data為pointer，而呼叫operator()又呼叫Constructor時會無法合成。

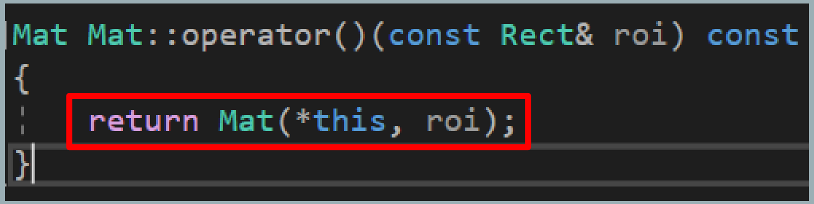
Solution: 改為直接inline到呼叫的位置，如下



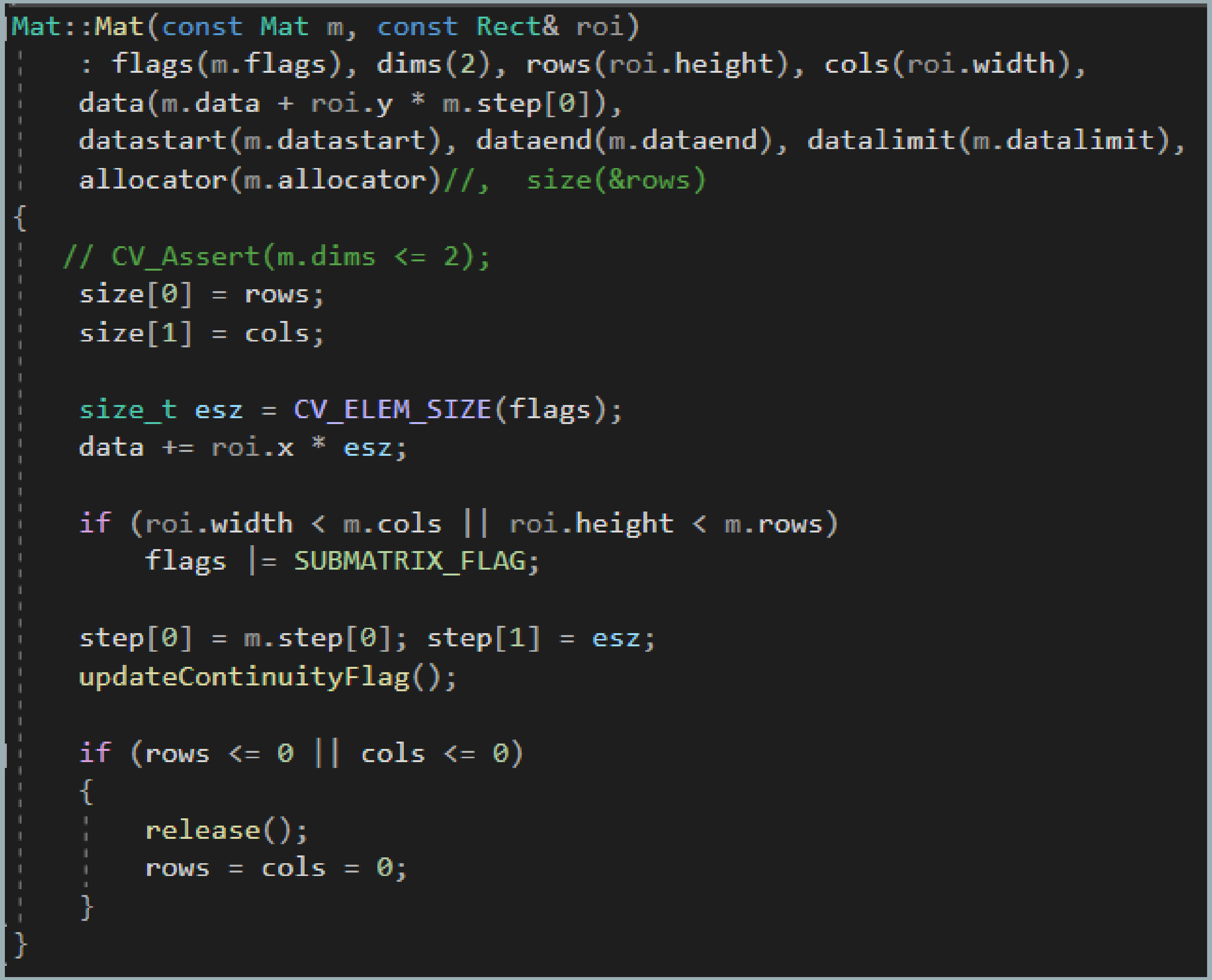
```
// extMask = maskPyramid(wholeLinfo);

    extMask.flags = maskPyramid.flags;
    extMask.dims = 2;
    extMask.rows = wholeLinfo.height;
    extMask.cols = wholeLinfo.width;

    extMask.data = maskPyramid.data + wholeLinfo.y * maskPyramid.step[0];

     extMask.size[0] = extMask.rows;
     extMask.size[1] = extMask.cols;

    size_t esz = CV_ELEM_SIZE(extMask.flags);
    extMask.data += wholeLinfo.x * esz;

    if (wholeLinfo.width < maskPyramid.cols || wholeLinfo.height < maskPyramid.rows)
        extMask.flags |= CV_SUBMAT_FLAG;

    extMask.step[0] = maskPyramid.step[0]; extMask.step[1] = esz;
    extMask.updateContinuityFlag();
```



```
uchar* data;
```

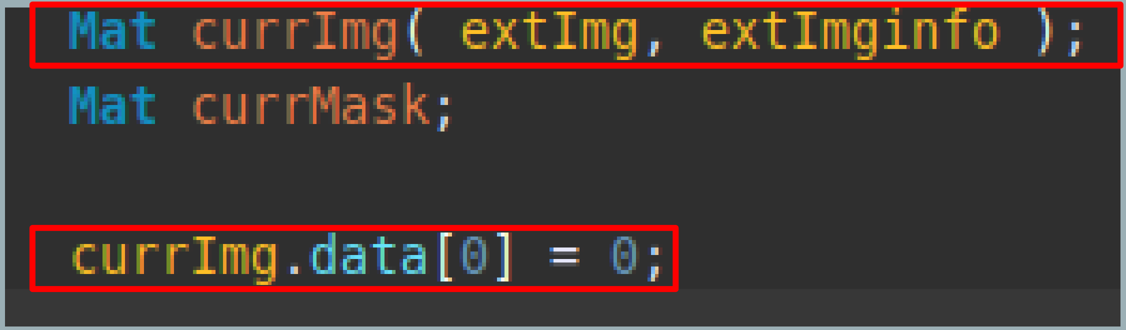


```
Mat Mat::operator()(const Rect& roi) const
{
    return Mat(*this, roi);
}
```



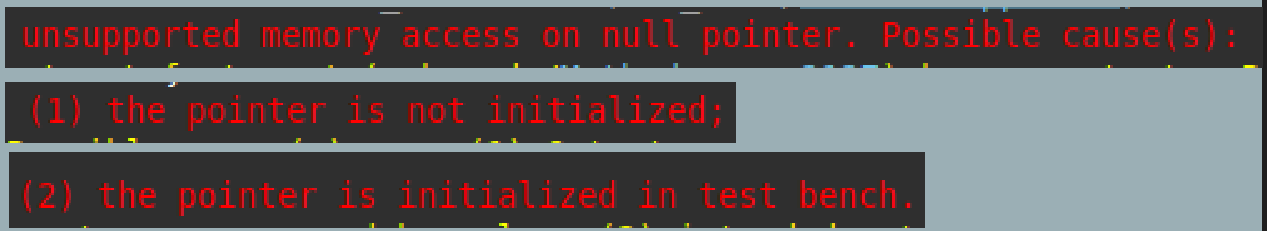
```
Mat::Mat(const Mat m, const Rect& roi)
    : flags(m.flags), dims(2), rows(roi.height), cols(roi.width),
    data(m.data + roi.y * m.step[0]),
    datastart(m.datastart), dataend(m.dataend), datalimit(m.datalimit),
    allocator(m.allocator)//,  size(&rows)
{
   // CV_Assert(m.dims <= 2);
    size[0] = rows;
    size[1] = cols;

    size_t esz = CV_ELEM_SIZE(flags);
    data += roi.x * esz;

    if (roi.width < m.cols || roi.height < m.rows)
        flags |= SUBMATRIX_FLAG;

    step[0] = m.step[0]; step[1] = esz;
    updateContinuityFlag();

    if (rows <= 0 || cols <= 0)
    {
        release();
        rows = cols = 0;
    }
}
```

1. Synthesis error



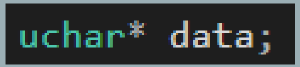
```
Mat currImg( extImg, extImginfo );
Mat currMask;

currImg.data[0] = 0;
```

用這種constructor創建Mat的物件，並寫入其中的data時，會出現以下報錯

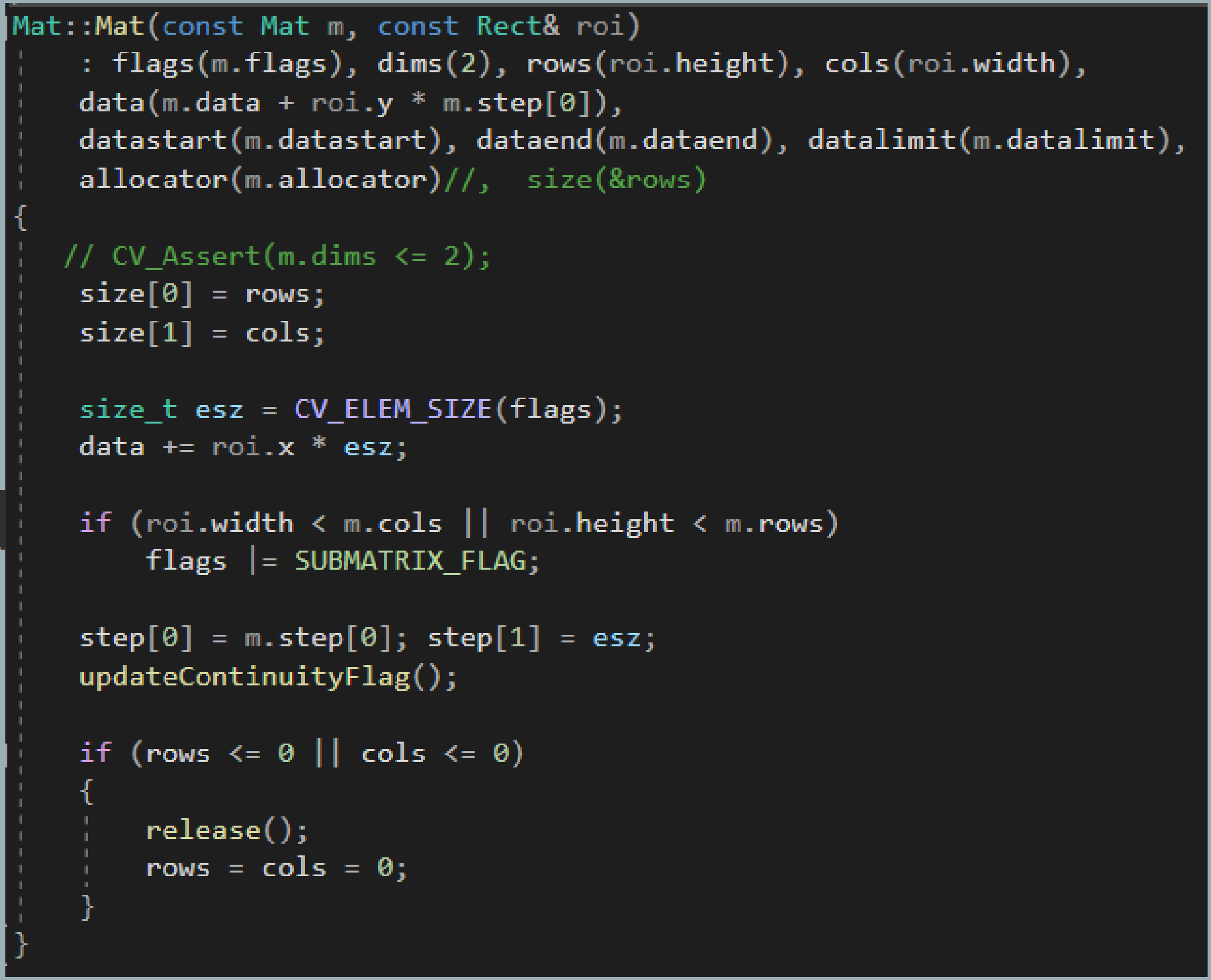


```
unsupported memory access on null pointer. Possible cause(s):
(1) the pointer is not initialized;
(2) the pointer is initialized in test bench.
```

Solution: 同樣改為直接inline到呼叫的位置

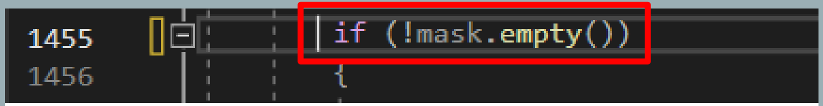


```
uchar* data;
```



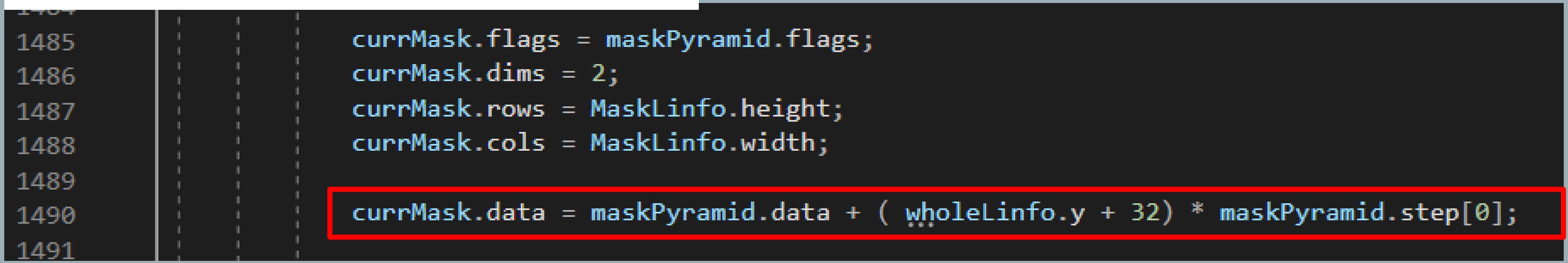
```
Mat::Mat(const Mat m, const Rect& roi)
    : flags(m.flags), dims(2), rows(roi.height), cols(roi.width),
    data(m.data + roi.y * m.step[0]),
    datastart(m.datastart), dataend(m.dataend), datalimit(m.datalimit),
    allocator(m.allocator)//,  size(&rows)
{
   // CV_Assert(m.dims <= 2);
    size[0] = rows;
    size[1] = cols;

    size_t esz = CV_ELEM_SIZE(flags);
    data += roi.x * esz;

    if (roi.width < m.cols || roi.height < m.rows)
        flags |= SUBMATRIX_FLAG;

    step[0] = m.step[0]; step[1] = esz;
    updateContinuityFlag();

    if (rows <= 0 || cols <= 0)
    {
        release();
        rows = cols = 0;
    }
}
```

2. Synthesis error

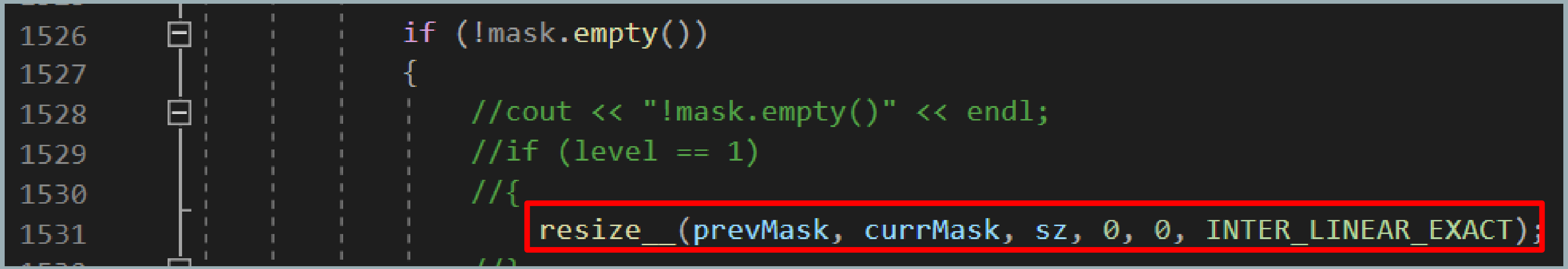


```
if (!mask.empty())
{
```

== 略 ==

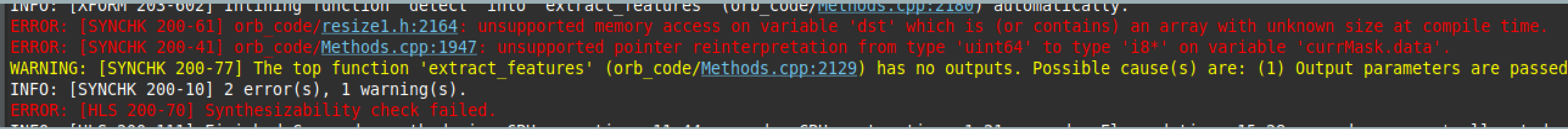


```
        currMask.flags = maskPyramid.flags;
        currMask.dims = 2;
        currMask.rows = MaskLinfo.height;
        currMask.cols = MaskLinfo.width;

        currMask.data = maskPyramid.data + ( wholeLinfo.y + 32) * maskPyramid.step[0];

```



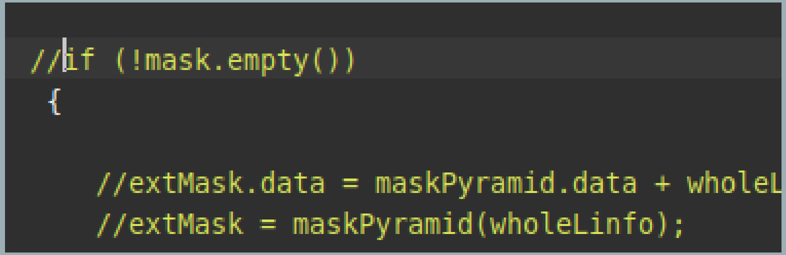
```
if (!mask.empty())
{
    //cout << "!mask.empty()" << endl;
    //if (level == 1)
    //{
        resize__(prevMask, currMask, sz, 0, 0, INTER_LINEAR_EXACT);
```

用 If 來決定currMask的data，否則先前只是宣告而已。
C simulation是可以的，但是合成時會產生以下報錯



```
INFO: [XFORM 203-602] Inlining function 'detect' into 'extract_features' (orb_code/Methods.cpp:2180) automatically.
ERROR: [SYNCHK 200-61] orb_code/resize1.h:2164: unsupported memory access on variable 'dst' which is (or contains) an array with unknown size at compile time.
ERROR: [SYNCHK 200-41] orb_code/Methods.cpp:1947: unsupported pointer reinterpretation from type 'uint64' to type 'i8*' on variable 'currMask.data'.
WARNING: [SYNCHK 200-77] The top function 'extract_features' (orb_code/Methods.cpp:2129) has no outputs. Possible cause(s) are: (1) Output parameters are passed
INFO: [SYNCHK 200-10] 2 error(s), 1 warning(s).
ERROR: [HLS 200-70] Synthesizability check failed.
```

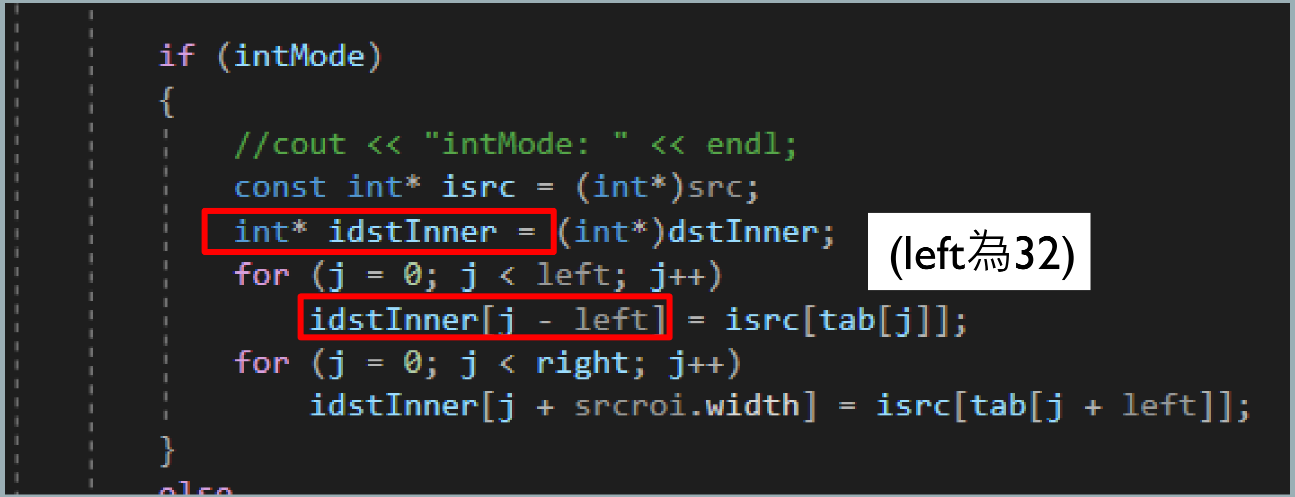
2. Synthesis error



```
//if (!mask.empty())
 {

     //extMask.data = maskPyramid.data + wholeL
     //extMask = maskPyramid(wholeLinfo);
```

Solution: 將 If 註解(刪除)，使得無論如何都將currMask初始化好

1. Synthesis failed

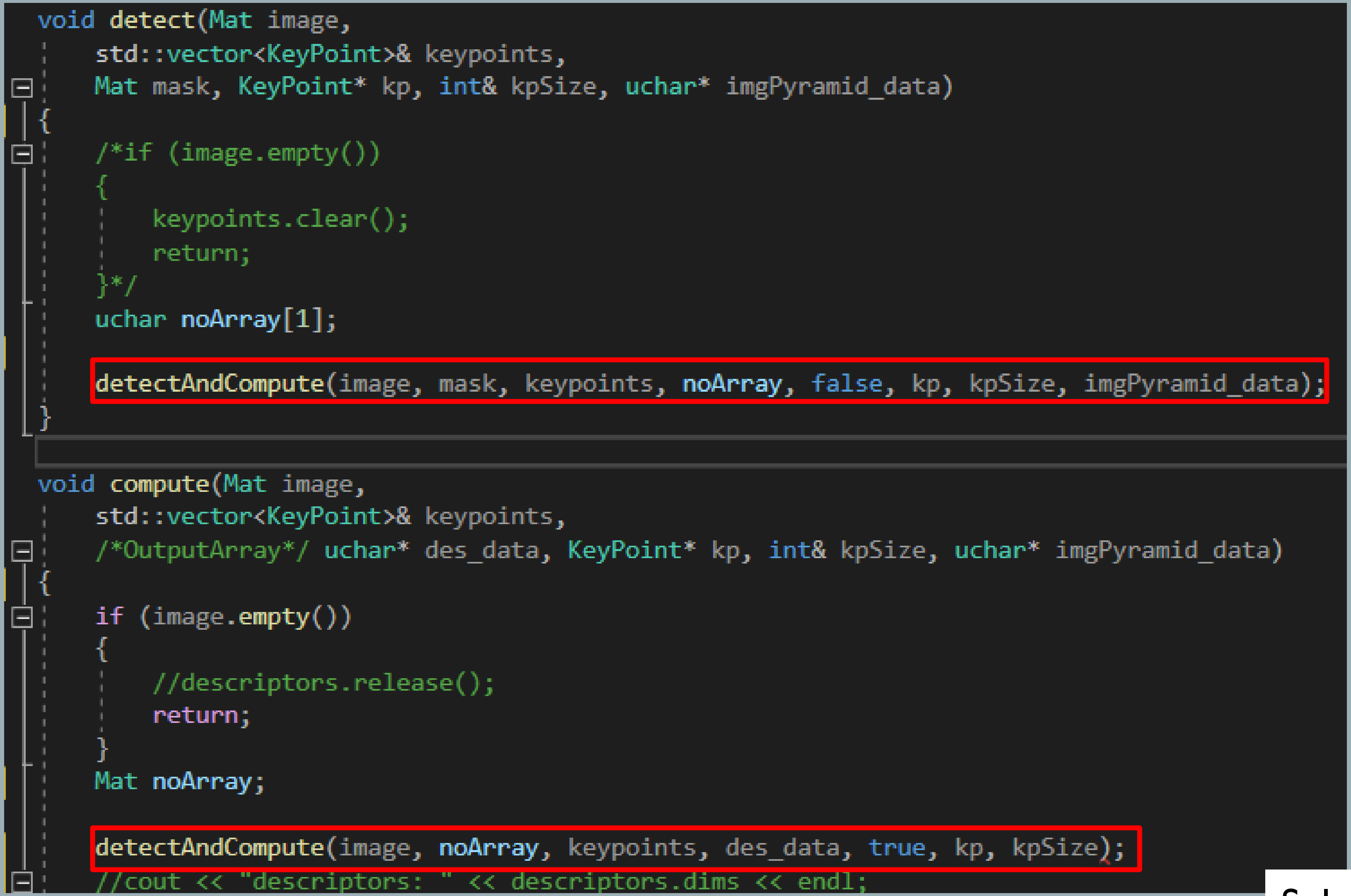


```
if (intMode)
{
    //cout << "intMode: " << endl;
    const int* isrc = (int*)src;
    int* idstInner = (int*)dstInner;
    for (j = 0; j < left; j++)
        idstInner[j - left] = isrc[tab[j]];
    for (j = 0; j < right; j++)
        idstInner[j + srcroi.width] = isrc[tab[j + left]];
}
else
```

(left為32)

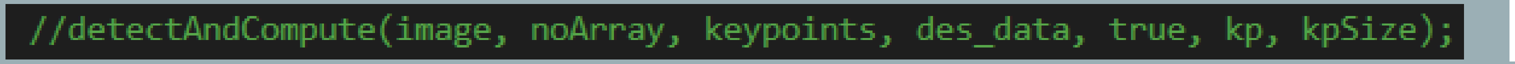
由於 j – left < 0 ，寫入<0的位置會無法合成。
Solution: 需重新改寫code，但是這邊檢測後發現不會是intMode，因此直接將這段刪除

2. Synthesis failed



```
void detect(Mat image,
    std::vector<KeyPoint>& keypoints,
    Mat mask, KeyPoint* kp, int& kpSize, uchar* imgPyramid_data)
{
    /*if (image.empty())
    {
        keypoints.clear();
        return;
    }*/
    uchar noArray[1];

    detectAndCompute(image, mask, keypoints, noArray, false, kp, kpSize, imgPyramid_data);
}

void compute(Mat image,
    std::vector<KeyPoint>& keypoints,
    /*OutputArray*/ uchar* des_data, KeyPoint* kp, int& kpSize, uchar* imgPyramid_data)
{
    if (image.empty())
    {
        //descriptors.release();
        return;
    }
    Mat noArray;

    detectAndCompute(image, noArray, keypoints, des_data, true, kp, kpSize);
    //cout << "descriptors: " << descriptors.dims << endl;
```

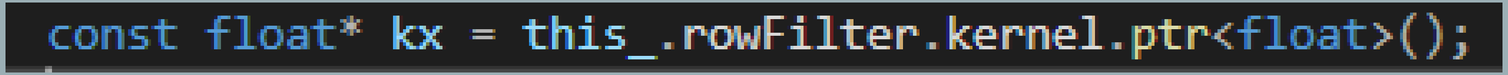
原先detect與compute都會呼叫detectAndCompute，但是會導致Synthesis failed



```
//detectAndCompute(image, noArray, keypoints, des_data, true, kp, kpSize);
```

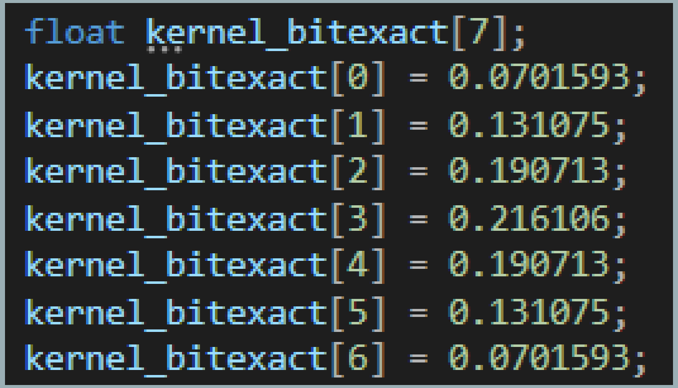
Solution:改為將detectAndCompute 的內容inline到呼叫的位置

3. Synthesis failed

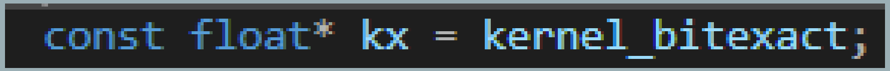


```
const float* kx = this_.rowFilter.kernel.ptr<float>();
```

取用class中的class會導致 Synthesis failed



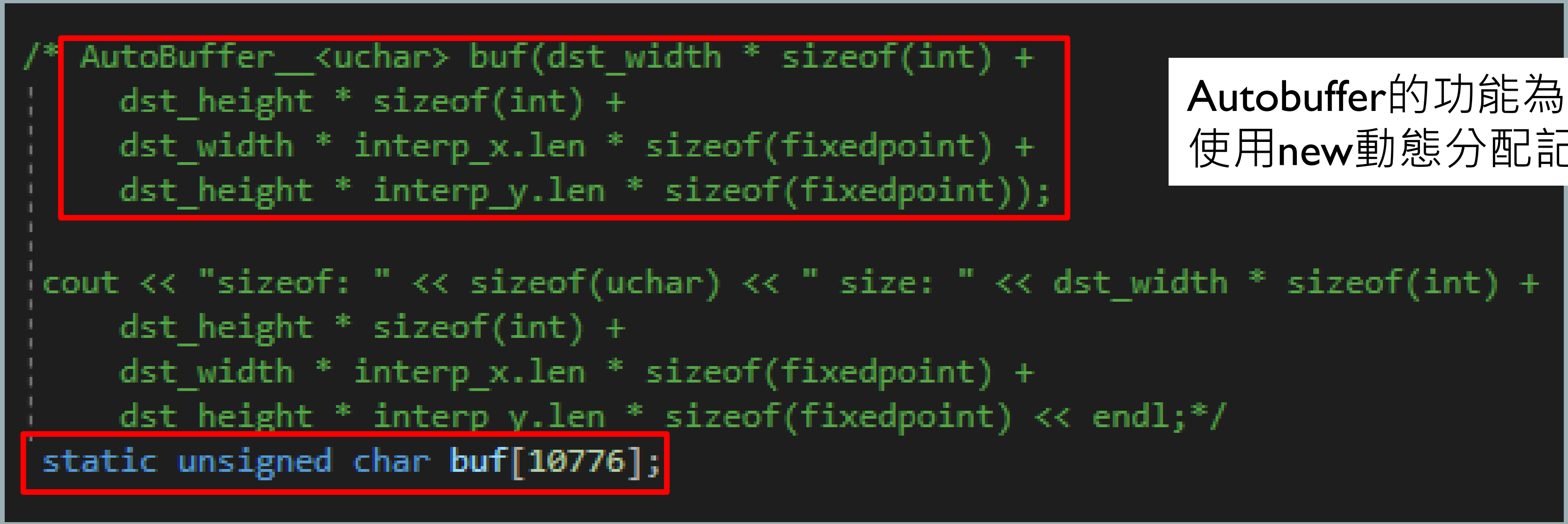
```
float kernel_bitexact[7];
kernel_bitexact[0] = 0.0701593;
kernel_bitexact[1] = 0.131075;
kernel_bitexact[2] = 0.190713;
kernel_bitexact[3] = 0.216106;
kernel_bitexact[4] = 0.190713;
kernel_bitexact[5] = 0.131075;
kernel_bitexact[6] = 0.0701593;
```



```
const float* kx = kernel_bitexact;
```

Solution:
改為直接宣告要取用的內容在此class中

無法動態分配記憶體(new delete)



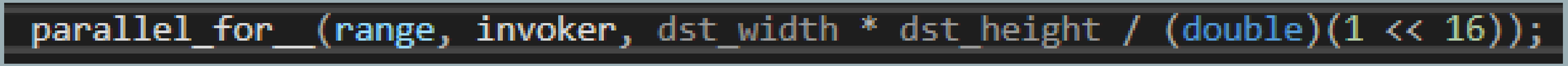
```
/* AutoBuffer__<uchar> buf(dst_width * sizeof(int) +
    dst_height * sizeof(int) +
    dst_width * interp_x.len * sizeof(fixedpoint) +
    dst_height * interp_y.len * sizeof(fixedpoint));

 cout << "sizeof: " << sizeof(uchar) << " size: " << dst_width * sizeof(int) +
    dst_height * sizeof(int) +
    dst_width * interp_x.len * sizeof(fixedpoint) +
    dst_height * interp_y.len * sizeof(fixedpoint) << endl;*/
 static unsigned char buf[10776];
```

Autobuffer的功能為根據傳入的數值，使用new動態分配記憶體

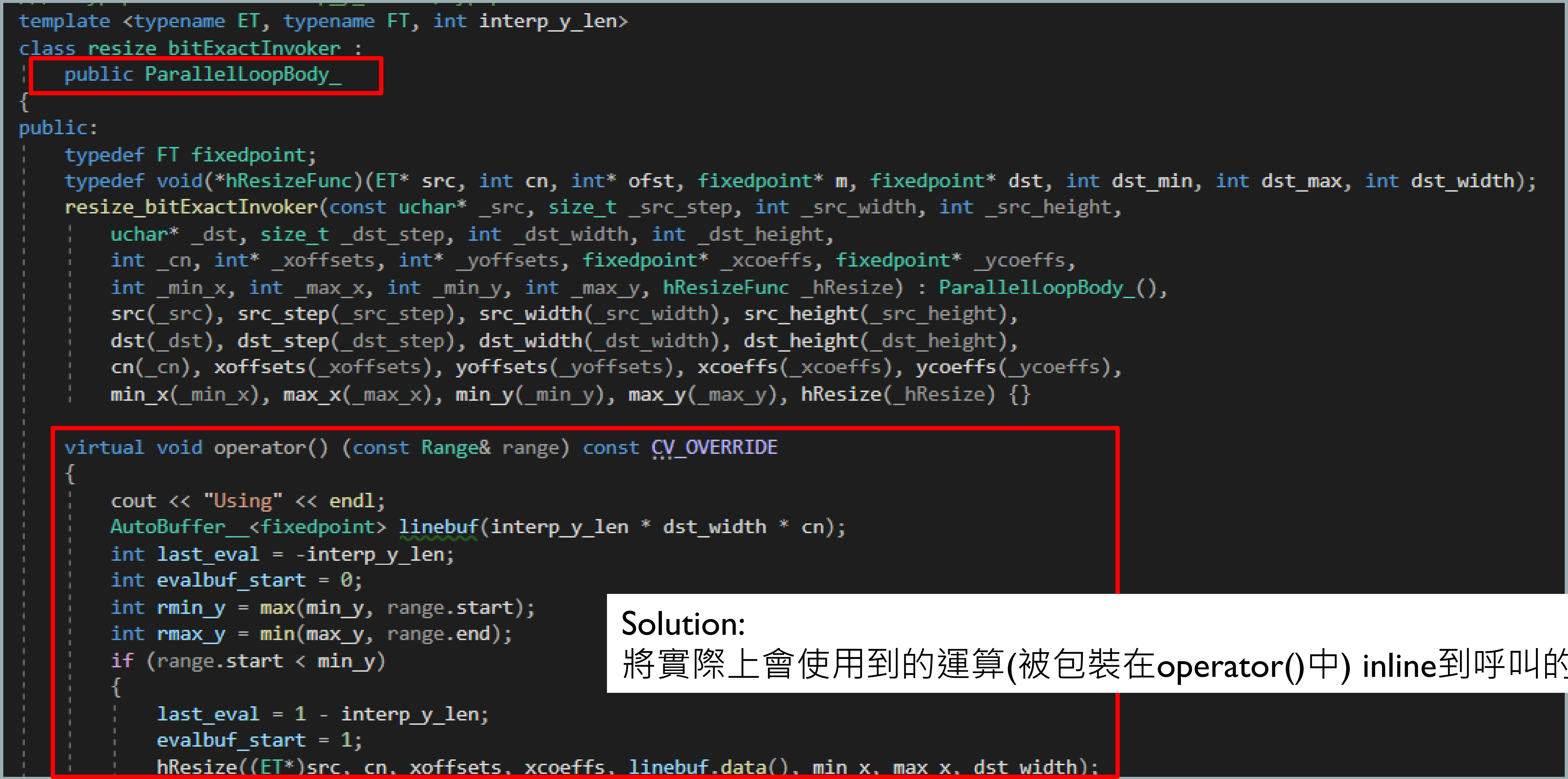
Solution:
由於我們使用相片的長寬固定為376*1241，這邊所需的記憶體大小皆固定，因此直接分配所需容量

無法使用OpenCV 的並行計算函數parallel_for_

由於牽涉到軟體中多線程的運算，合成時須做修改



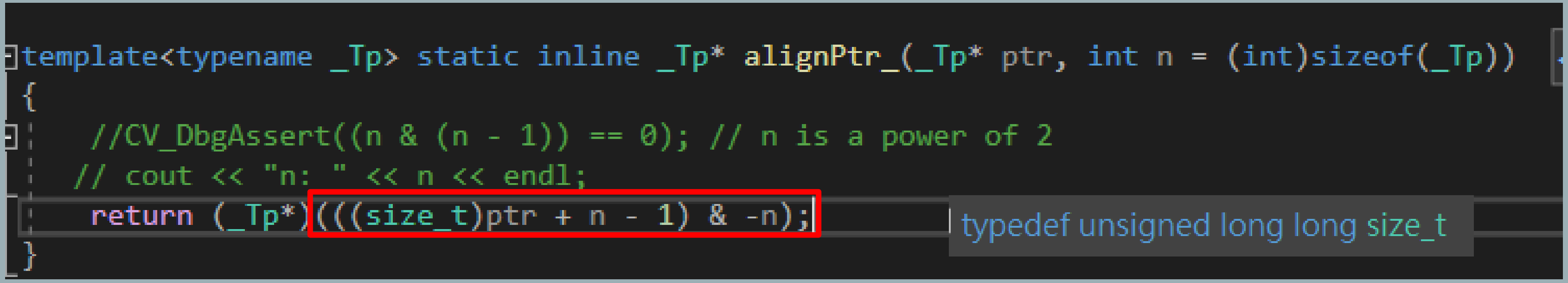
```
parallel_for__(range, invoker, dst_width * dst_height / (double)(1 << 16));
```



```
template <typename ET, typename FT, int interp_y_len>
class resize_bitExactInvoker :
    public ParallelLoopBody_
{
public:
    typedef FT fixedpoint;
    typedef void(*hResizeFunc)(ET* src, int cn, int* ofst, fixedpoint* m, fixedpoint* dst, int dst_min, int dst_max, int dst_width);
    resize_bitExactInvoker(const uchar* _src, size_t _src_step, int _src_width, int _src_height,
        uchar* _dst, size_t _dst_step, int _dst_width, int _dst_height,
        int _cn, int* _xoffsets, int* _yoffsets, fixedpoint* _xcoeffs, fixedpoint* _ycoeffs,
        int _min_x, int _max_x, int _min_y, int _max_y, hResizeFunc _hResize) : ParallelLoopBody_(),
        src(_src), src_step(_src_step), src_width(_src_width), src_height(_src_height),
        dst(_dst), dst_step(_dst_step), dst_width(_dst_width), dst_height(_dst_height),
        cn(_cn), xoffsets(_xoffsets), yoffsets(_yoffsets), xcoeffs(_xcoeffs), ycoeffs(_ycoeffs),
        min_x(_min_x), max_x(_max_x), min_y(_min_y), max_y(_max_y), hResize(_hResize) {}

    virtual void operator() (const Range& range) const CV_OVERRIDE
    {
        cout << "Using" << endl;
        AutoBuffer__<fixedpoint> linebuf(interp_y_len * dst_width * cn);
        int last_eval = -interp_y_len;
        int evalbuf_start = 0;
        int rmin_y = max(min_y, range.start);
        int rmax_y = min(max_y, range.end);
        if (range.start < min_y)
        {
            last_eval = 1 - interp_y_len;
            evalbuf_start = 1;
            hResize((ET*)src, cn, xoffsets, xcoeffs, linebuf.data(), min_x, max_x, dst_width);
```

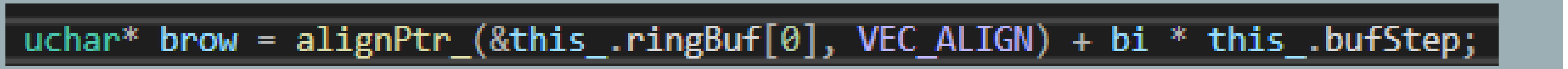
Solution:
將實際上會使用到的運算(被包裝在operator()中) inline到呼叫的位置

不能做地址的運算



```
template<typename _Tp> static inline _Tp* alignPtr_(_Tp* ptr, int n = (int)sizeof(_Tp))
{
    //CV_DbgAssert((n & (n - 1)) == 0); // n is a power of 2
   // cout << "n: " << n << endl;
    return (_Tp*)(((size_t)ptr + n - 1) & -n);
}
```

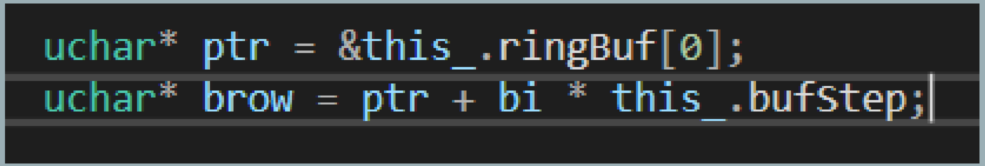
typedef unsigned long long size_t



```
uchar* brow = alignPtr_(&this_.ringBuf[0], VEC_ALIGN) + bi * this_.bufStep;
```

呼叫alignPtr_時，會做地址的運算，但是使用(size_t)會無法合成，也不能直接做&的計算

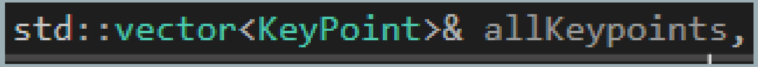
電腦在做運算時，若地址後面6個bit為0作為初始位置會加快運算的速度，但是fpga的on chip buffer沒有關聯



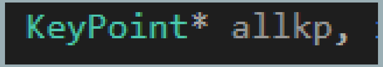
```
uchar* ptr = &this_.ringBuf[0];
uchar* brow = ptr + bi * this_.bufStep;
```

Solution:
alignPtr_的功用就是給ptr一個初始位置，之後就用這個位置開始做運算，因此可以直接指派ptr一個想要的初始位置即可

不能使用 std::vector

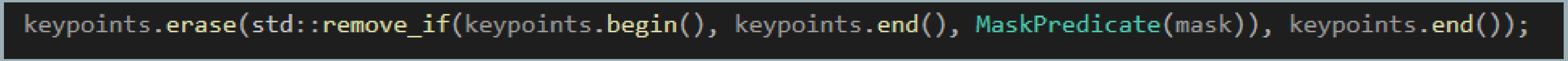


```
std::vector<KeyPoint>& allKeypoints,
```

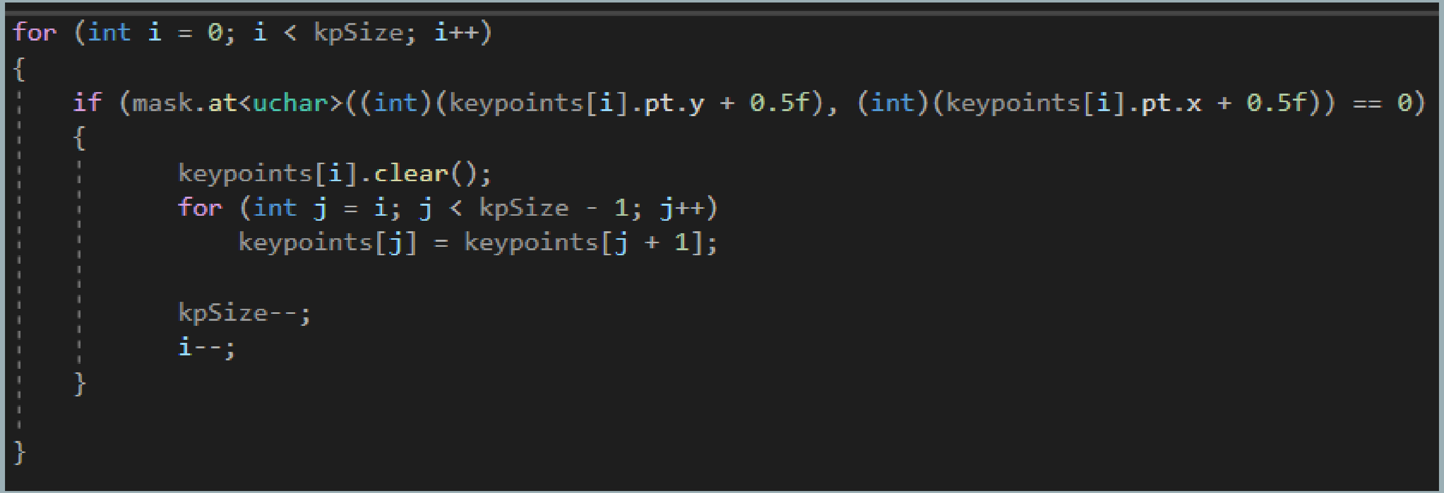


```
KeyPoint* allkp,
```

Solution:
將vector改寫為array或pointer



```
keypoints.erase(std::remove_if(keypoints.begin(), keypoints.end(), MaskPredicate(mask)), keypoints.end());
```

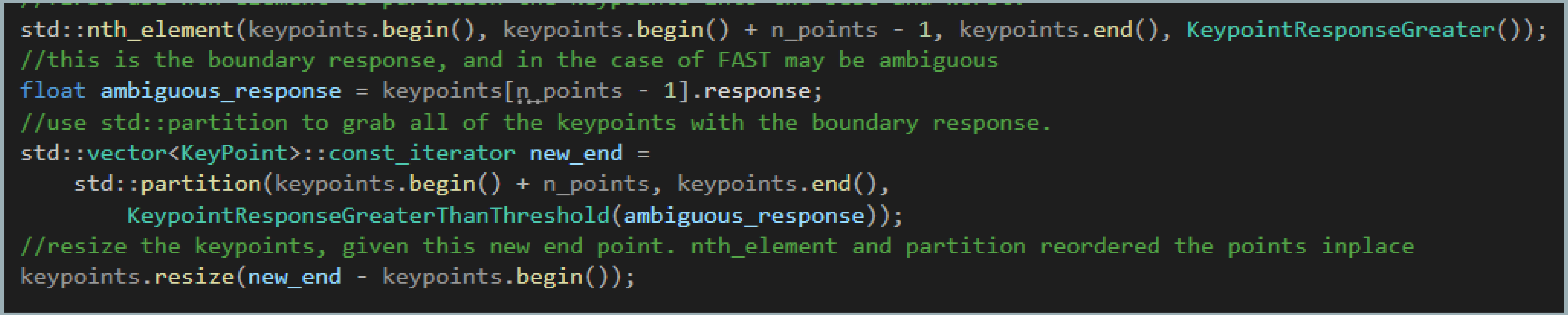


```
for (int i = 0; i < kpSize; i++)
{
    if (mask.at<uchar>((int)(keypoints[i].pt.y + 0.5f), (int)(keypoints[i].pt.x + 0.5f)) == 0)
    {
           keypoints[i].clear();
           for (int j = i; j < kpSize - 1; j++)
               keypoints[j] = keypoints[j + 1];

           kpSize--;
           i--;
    }

}
```

使用到 vector 與 std 的內建函數
erase 與 remove_if

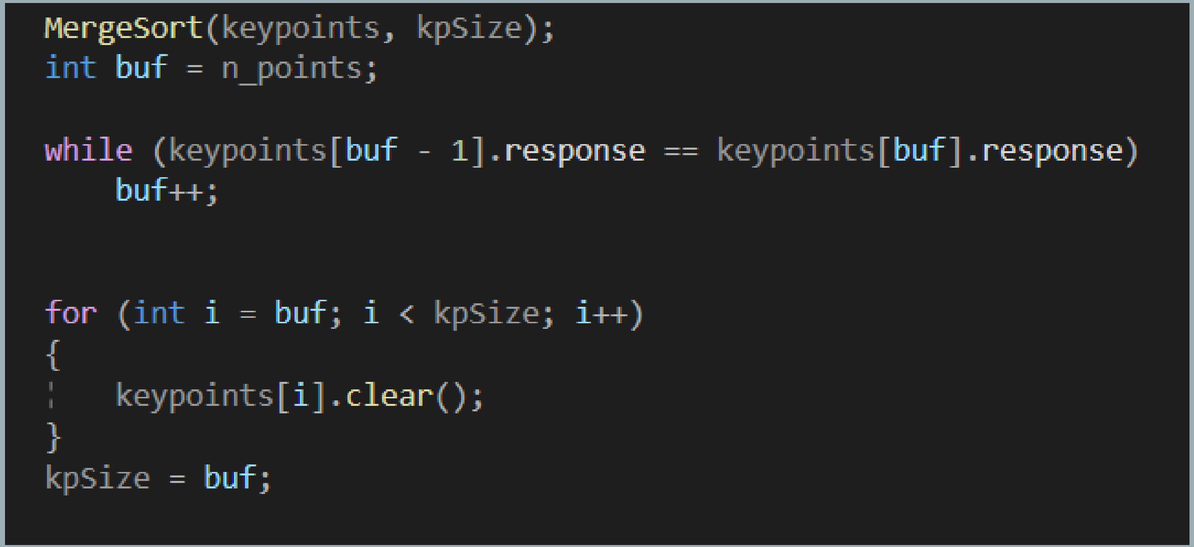
Solution:
自行改寫為可以合成的版本

不能使用 std::vector



```
std::nth_element(keypoints.begin(), keypoints.begin() + n_points - 1, keypoints.end(), KeypointResponseGreater());
//this is the boundary response, and in the case of FAST may be ambiguous
float ambiguous_response = keypoints[n_points - 1].response;
//use std::partition to grab all of the keypoints with the boundary response.
std::vector<KeyPoint>::const_iterator new_end =
    std::partition(keypoints.begin() + n_points, keypoints.end(),
        KeypointResponseGreaterThanThreshold(ambiguous_response));
//resize the keypoints, given this new end point. nth_element and partition reordered the points inplace
keypoints.resize(new_end - keypoints.begin());
```

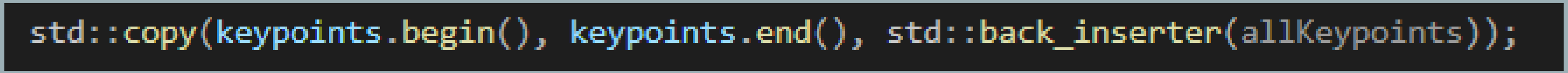
使用到 std 與 vector 的內建函數
nth_element、partition與 resize



```
MergeSort(keypoints, kpSize);
int buf = n_points;

while (keypoints[buf - 1].response == keypoints[buf].response)
    buf++;


for (int i = buf; i < kpSize; i++)
{
    keypoints[i].clear();
}
kpSize = buf;
```

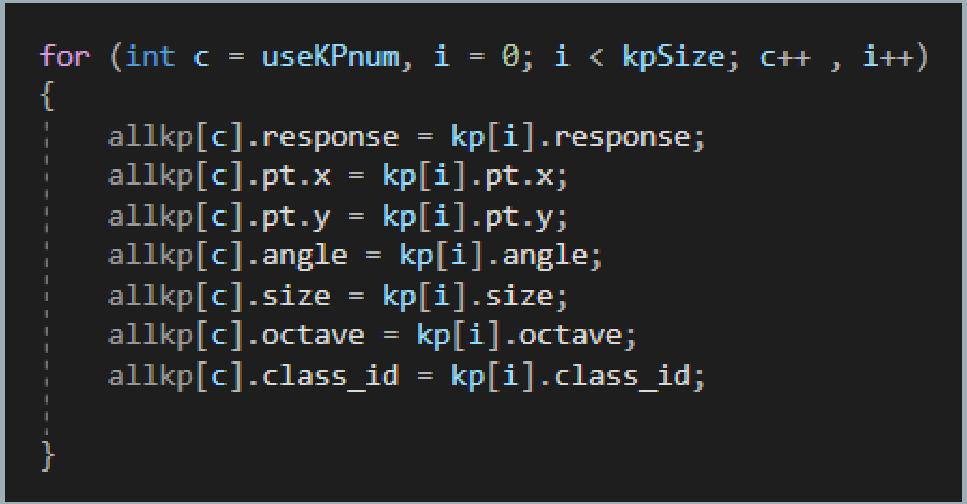
Solution:
由於原先的運算就是找出response排序前n_points個的keypoint，因此使用merge sort改寫為可以合成的版本

不能使用 std::vector



```
std::copy(keypoints.begin(), keypoints.end(), std::back_inserter(allKeypoints));
```

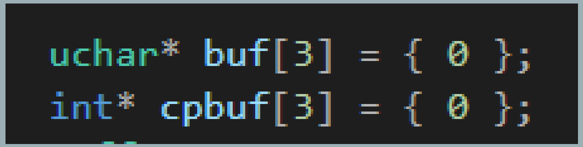
使用到 std 的內建函數 copy



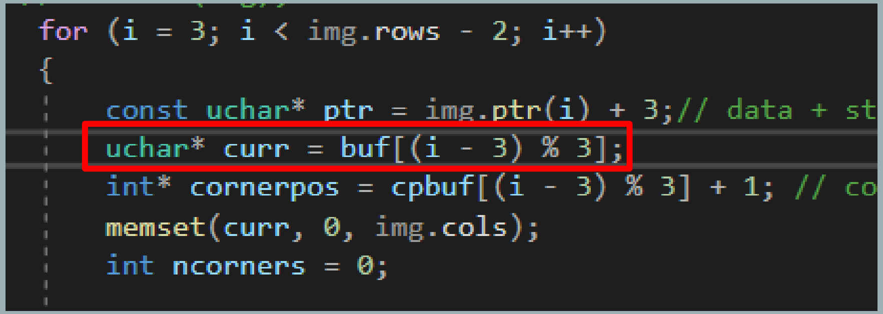
```
for (int c = useKPnum, i = 0; i < kpSize; c++ , i++)
{
    allkp[c].response = kp[i].response;
    allkp[c].pt.x = kp[i].pt.x;
    allkp[c].pt.y = kp[i].pt.y;
    allkp[c].angle = kp[i].angle;
    allkp[c].size = kp[i].size;
    allkp[c].octave = kp[i].octave;
    allkp[c].class_id = kp[i].class_id;

}
```

Solution:
用 for 迴圈改寫為可以合成的版本

不能使用 pointer to array

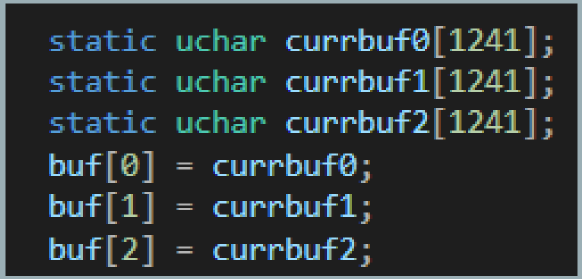


```
uchar* buf[3] = { 0 };
int* cpbuf[3] = { 0 };
```

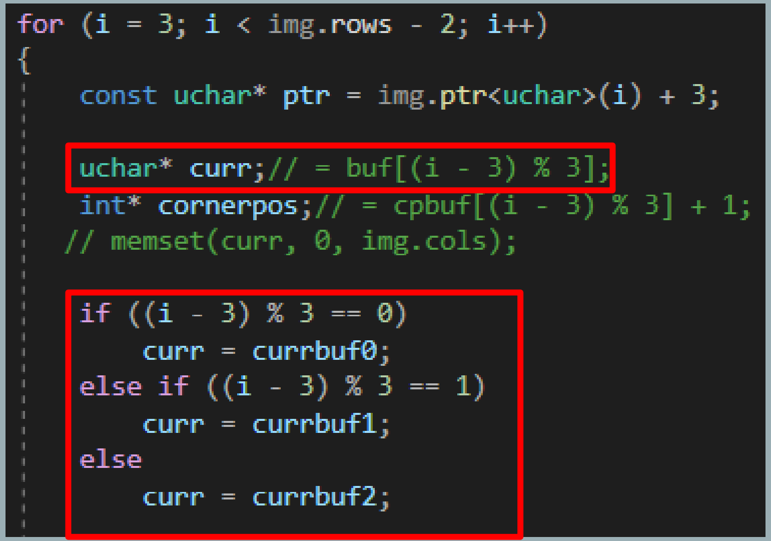


```
for (i = 3; i < img.rows - 2; i++)
{
    const uchar* ptr = img.ptr(i) + 3;// data + st
    uchar* curr = buf[(i - 3) % 3];
    int* cornerpos = cpbuf[(i - 3) % 3] + 1; // co
    memset(curr, 0, img.cols);
    int ncorners = 0;
```

無法合成 pointer to array



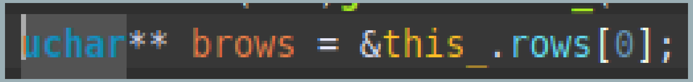
```
static uchar currbuf0[1241];
static uchar currbuf1[1241];
static uchar currbuf2[1241];
buf[0] = currbuf0;
buf[1] = currbuf1;
buf[2] = currbuf2;
```



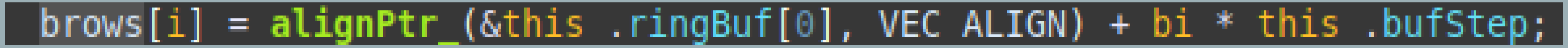
```
for (i = 3; i < img.rows - 2; i++)
{
    const uchar* ptr = img.ptr<uchar>(i) + 3;

    uchar* curr;// = buf[(i - 3) % 3];
    int* cornerpos;// = cpbuf[(i - 3) % 3] + 1;
   // memset(curr, 0, img.cols);

    if ((i - 3) % 3 == 0)
        curr = currbuf0;
    else if ((i - 3) % 3 == 1)
        curr = currbuf1;
    else
        curr = currbuf2;
```

Solution:
建立所需的 array 數量，並且計算需要用到哪個 array 後指派給他

不能使用 pointer to pointer

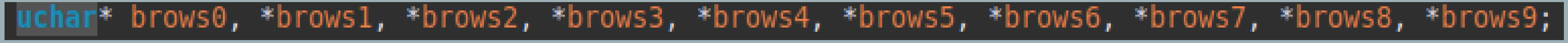


```
uchar** brows = &this_.rows[0];
```

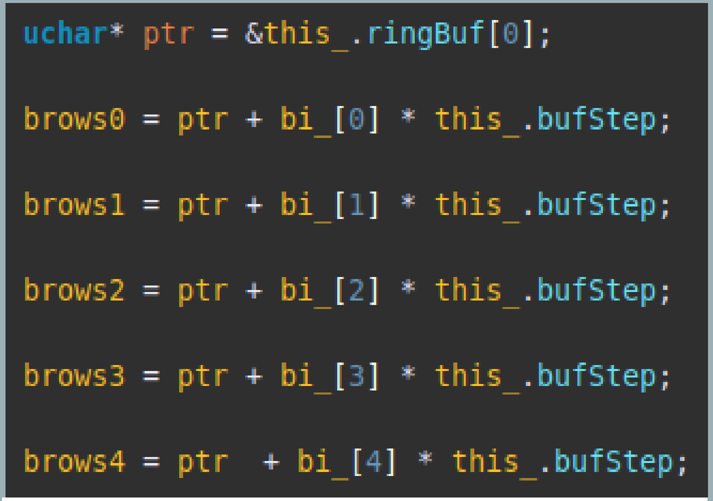


```
brows[i] = alignPtr_(&this_.ringBuf[0], VEC_ALIGN) + bi * this_.bufStep;
```

無法合成 pointer to pointer

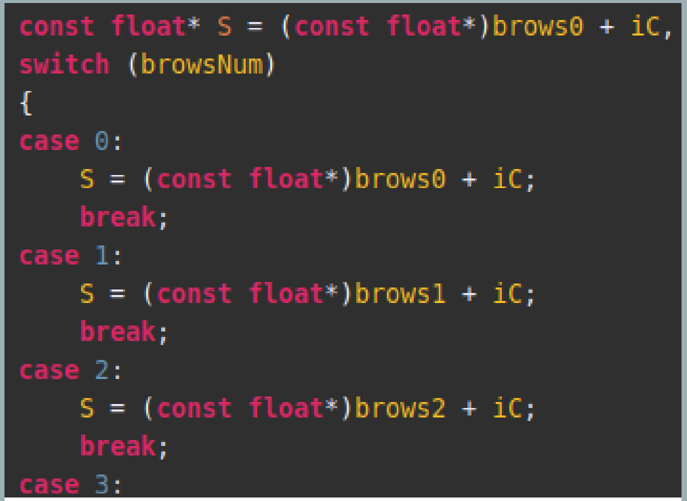


```
uchar* brows0, *brows1, *brows2, *brows3, *brows4, *brows5, *brows6, *brows7, *brows8, *brows9;
```



```
uchar* ptr = &this_.ringBuf[0];

brows0 = ptr + bi_[0] * this_.bufStep;

brows1 = ptr + bi_[1] * this_.bufStep;

brows2 = ptr + bi_[2] * this_.bufStep;

brows3 = ptr + bi_[3] * this_.bufStep;

brows4 = ptr  + bi_[4] * this_.bufStep;
```

== 略 ==

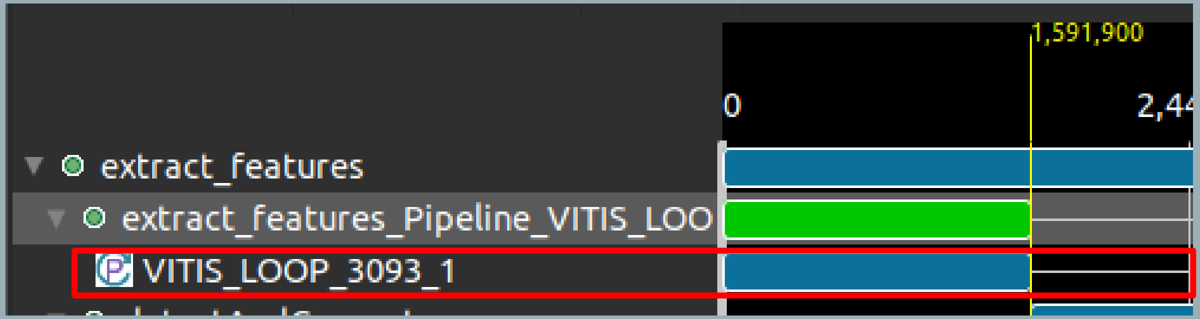


```
const float* S = (const float*)brows0 + iC,
switch (browsNum)
{
case 0:
    S = (const float*)brows0 + iC;
    break;
case 1:
    S = (const float*)brows1 + iC;
    break;
case 2:
    S = (const float*)brows2 + iC;
    break;
case 3:
```

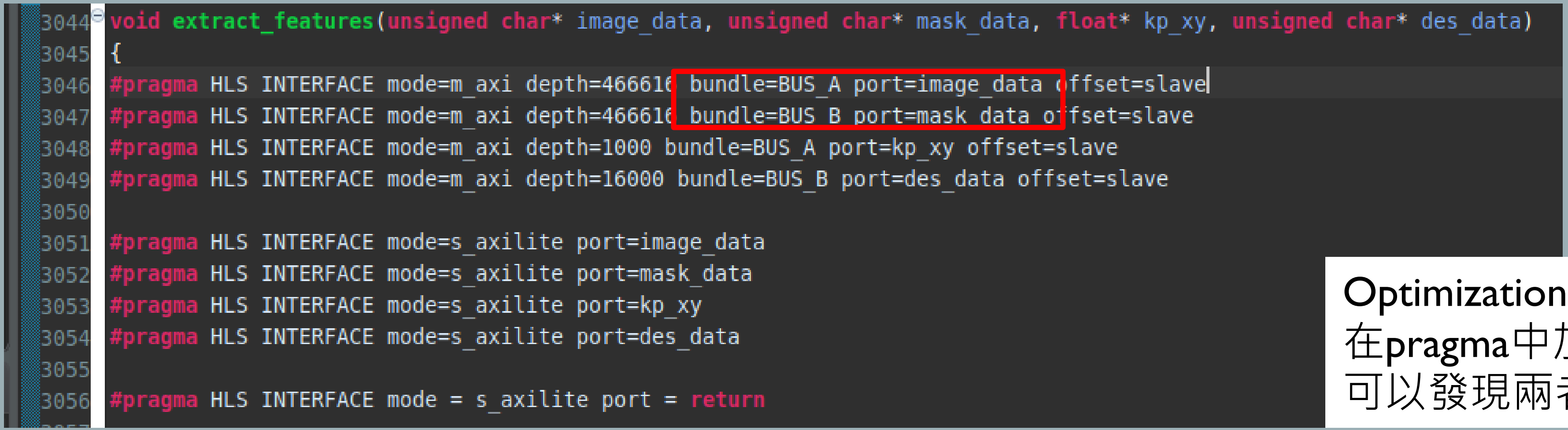
== 略 ==

Solution:
建立所需的 pointer 數量，並且個別賦予所需的地址。
取用時，使用switch來選擇。

兩張圖片同時傳遞

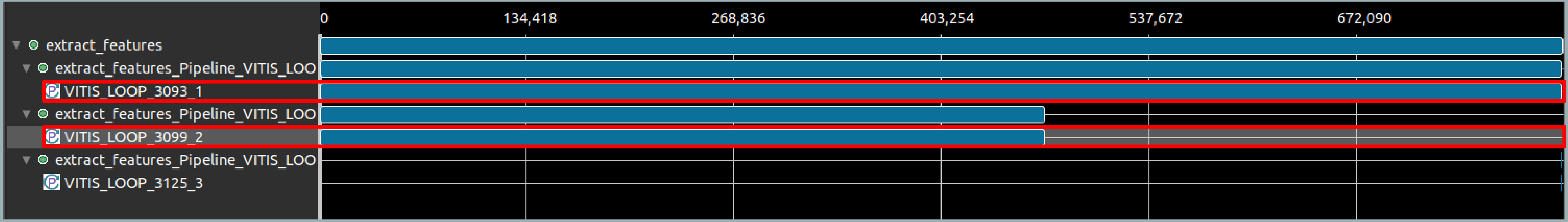


原先傳輸image_data與mask_data
共需花約1,594,900 cycle

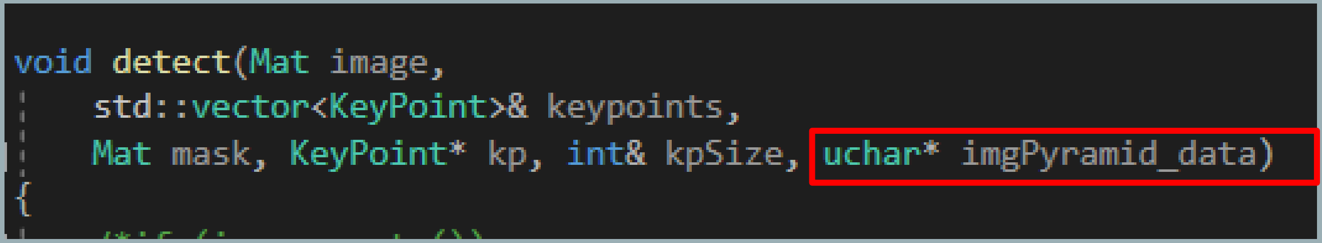


```
void extract_features(unsigned char* image_data, unsigned char* mask_data, float* kp_xy, unsigned char* des_data)
{
#pragma HLS INTERFACE mode=m_axi depth=466616 bundle=BUS_A port=image_data offset=slave
#pragma HLS INTERFACE mode=m_axi depth=466616 bundle=BUS_B port=mask_data offset=slave
#pragma HLS INTERFACE mode=m_axi depth=1000 bundle=BUS_A port=kp_xy offset=slave
#pragma HLS INTERFACE mode=m_axi depth=16000 bundle=BUS_B port=des_data offset=slave

#pragma HLS INTERFACE mode=s_axilite port=image_data
#pragma HLS INTERFACE mode=s_axilite port=mask_data
#pragma HLS INTERFACE mode=s_axilite port=kp_xy
#pragma HLS INTERFACE mode=s_axilite port=des_data

#pragma HLS INTERFACE mode = s_axilite port = return
```

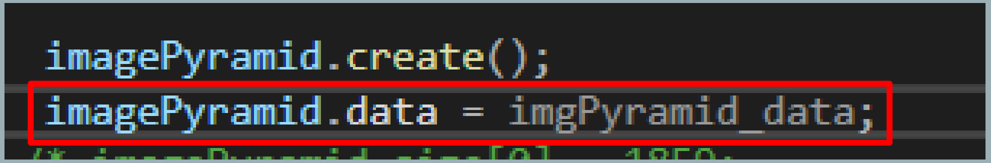
Optimization：
在pragma中加入bundle BUS_A與BUS_B，
可以發現兩者變為同時傳輸。


0
134,418
268,836
403,254
537,672
672,090
extract_features
extract_features_Pipeline_VITIS_LOO
VITIS_LOOP_3093_1
extract_features_Pipeline_VITIS_LOO
VITIS_LOOP_3099_2
extract_features_Pipeline_VITIS_LOO
VITIS_LOOP_3125_3

刪減重複計算

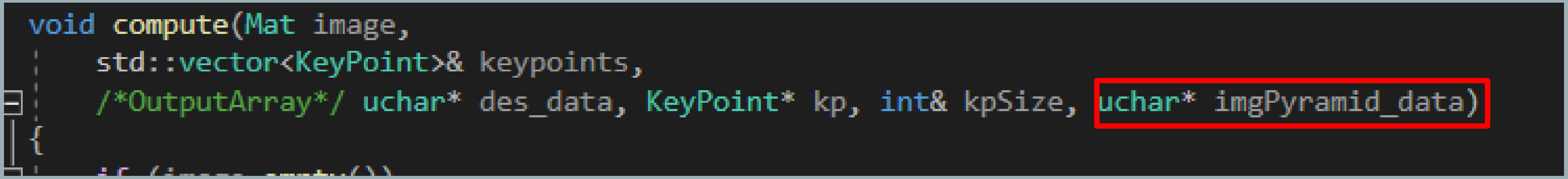


```
void detect(Mat image,
    std::vector<KeyPoint>& keypoints,
    Mat mask, KeyPoint* kp, int& kpSize, uchar* imgPyramid_data)
{
    /*if (image.empty())
```

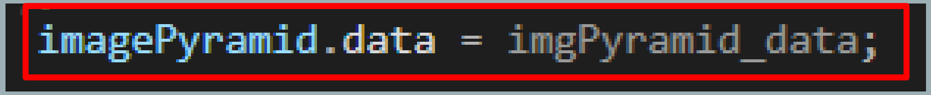


```
 imagePyramid.create();
 imagePyramid.data = imgPyramid_data;
/* imagePyramid.size[0] = 1850;
```

在 detect 與 compute中都會需要建立 imagePyramid，而兩者建立完成後一模一樣



```
void compute(Mat image,
    std::vector<KeyPoint>& keypoints,
    /*OutputArray*/ uchar* des_data, KeyPoint* kp, int& kpSize, uchar* imgPyramid_data)
{
    if (image.empty())
```



```
imagePyramid.data = imgPyramid_data;
```

Optimization：
在detect建立完後就直接傳給compute用