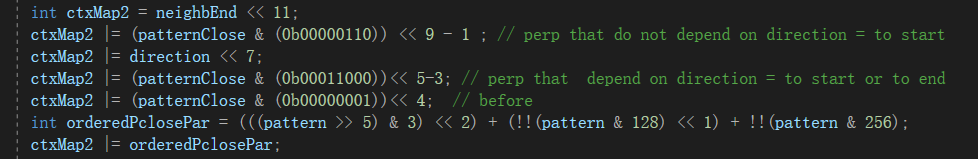
**次要信息上下文顺序优化——v2**

**标识信息现有上下文排列顺序：**



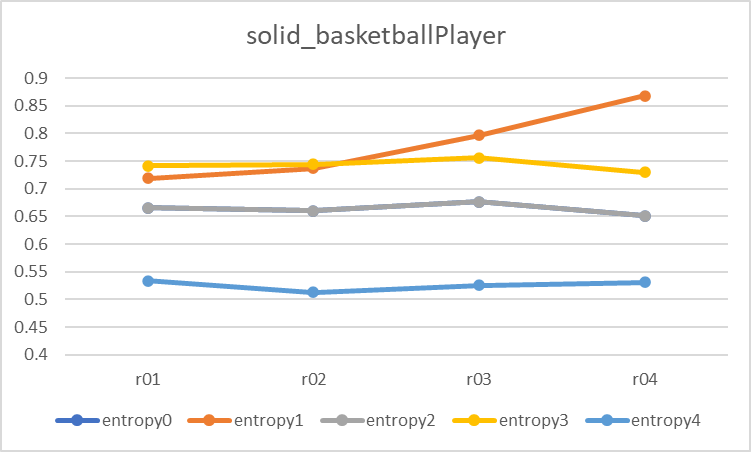
我们将上下文进行编号依次为 1、2、3、4、5、6

其物理含义分别为：

* ：表示包含当前待编码边终止点的 4 个节点的占据情况
* ：9 条边在 close 区间被占据的情况
* ：当前待编码边的朝向（0=X，1=Y，2=Z）
* ：9 条边的被占据的情况
* ： 不相邻的平行与垂直边

**测试过程与熵值对比结果**

1. 选取现有上下文顺序中排在最前面的neighbEnd作为第一个上下文，然后在此条件下其他上下文熵值有：

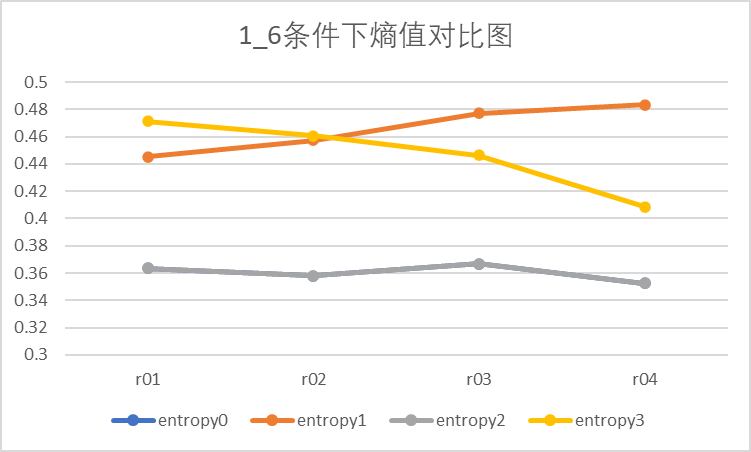


其中：

|  |
| --- |
| C++ entropy0：neighbEnd\*4+(patternClose & (0b00000110)>>1); entropy1：neighbEnd\*4+direction; entropy2：eighbEnd\*4+(patternClose & (0b00011000)>>3); entropy3：neighbEnd\*2+(patternClose & (0b00000001)); entropy4：neighbEnd\*16+orderedPclosePar; |

由于entropy4最小，我们选择的第二个上下文为orderedPclosePar

1. 在第一个与第二个上下文均确定后，在此条件下测试其他四个上下文的熵：

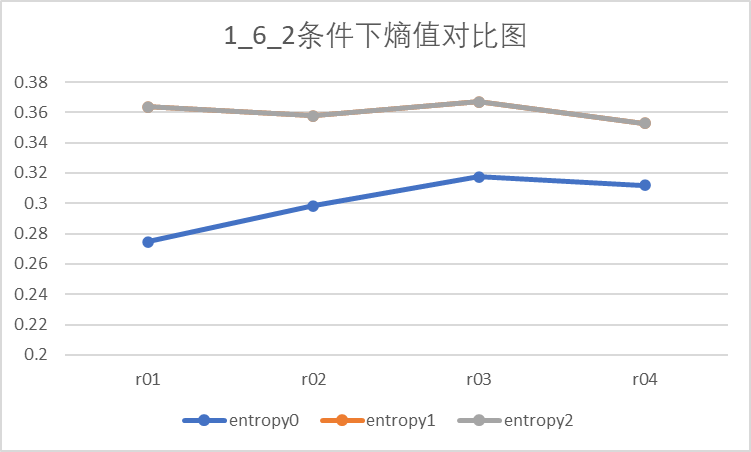


其中：

|  |
| --- |
| C++ entropy0：int ctx1\_6\_2 = (ctx1\_6 << 2) | (patternClose & (0b00000110) >> 1); entropy1：int ctx1\_6\_3 = (ctx1\_6 << 2) | direction; entropy2：int ctx1\_6\_4 = (ctx1\_6 << 2) | (patternClose & (0b00011000) >> 3); entropy3：int ctx1\_6\_5 = (ctx1\_6 << 1) | (patternClose & (0b00000001)); |

由于entropy0与entroy2相同且最小，我们选择的第三个上下文为patternClose & (0b00000110) >> 1

1. 在前三个上下文均确定后，在此条件下测试其他三个上下文的熵：

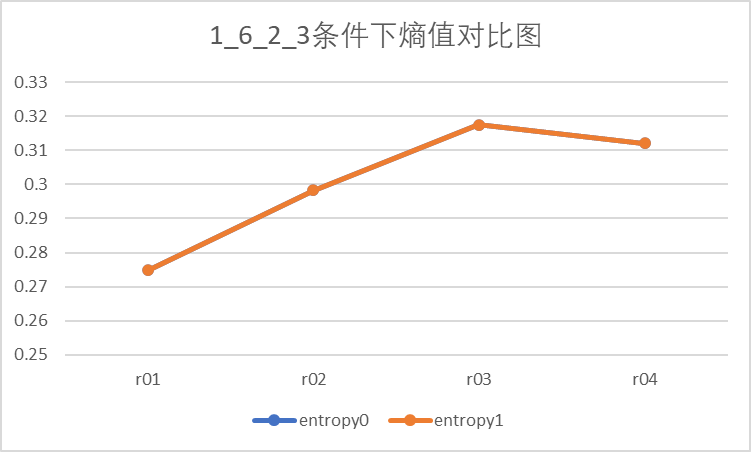


其中：

|  |
| --- |
| C++ entropy0：int ctx1\_6\_2\_3 = (ctx1\_6\_2 << 2) | direction; entropy1：int ctx1\_6\_2\_4 = (ctx1\_6\_2 << 2) | (patternClose & (0b00011000) >> 3); entropy2：int ctx1\_6\_2\_5 = (ctx1\_6\_2 << 1) | (patternClose & (0b00000001)); |

由于entropy0最小，我们选择的第四个上下文为direction

1. 在前四个上下文确定后，在此条件下测试其他两个上下文的熵：



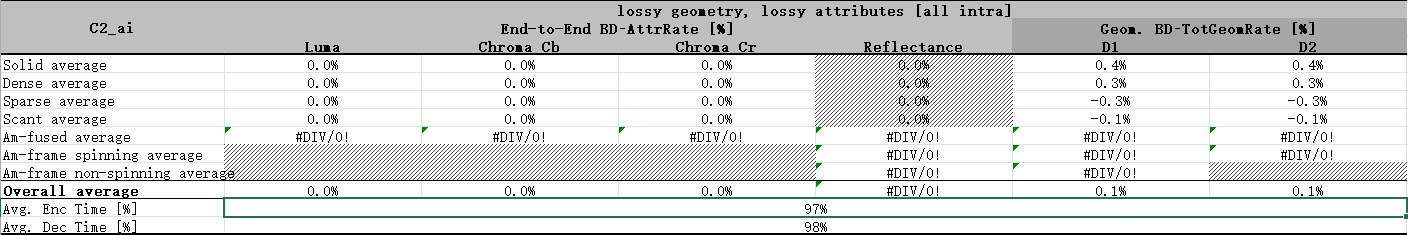
其中：

|  |
| --- |
| SQL entropy0：int ctx1\_6\_2\_3\_4 = (ctx1\_6\_2\_3 << 2) | (patternClose & (0b00011000) >> 3); entropy1：int ctx1\_6\_2\_3\_5 = (ctx1\_6\_2\_3 << 1) | (patternClose & (0b00000001)); |

剩下两个上下文熵值大小一致，根据其物理含义，一个是共线边，一个是其中两条相邻垂直边，选取(patternClose & (0b00011000) >> 3)作为下一个上下文，因此共线边patternClose & (0b00000001)成为最后一个上下文。

|  |
| --- |
| 综上，得到局部最优上下文顺序为：1-6-2-3-4-5 |

**性能测试结果**



在solid与dense类型序列有loss

在sparse与scant类型序列有增益

* 其中sparse类型中：

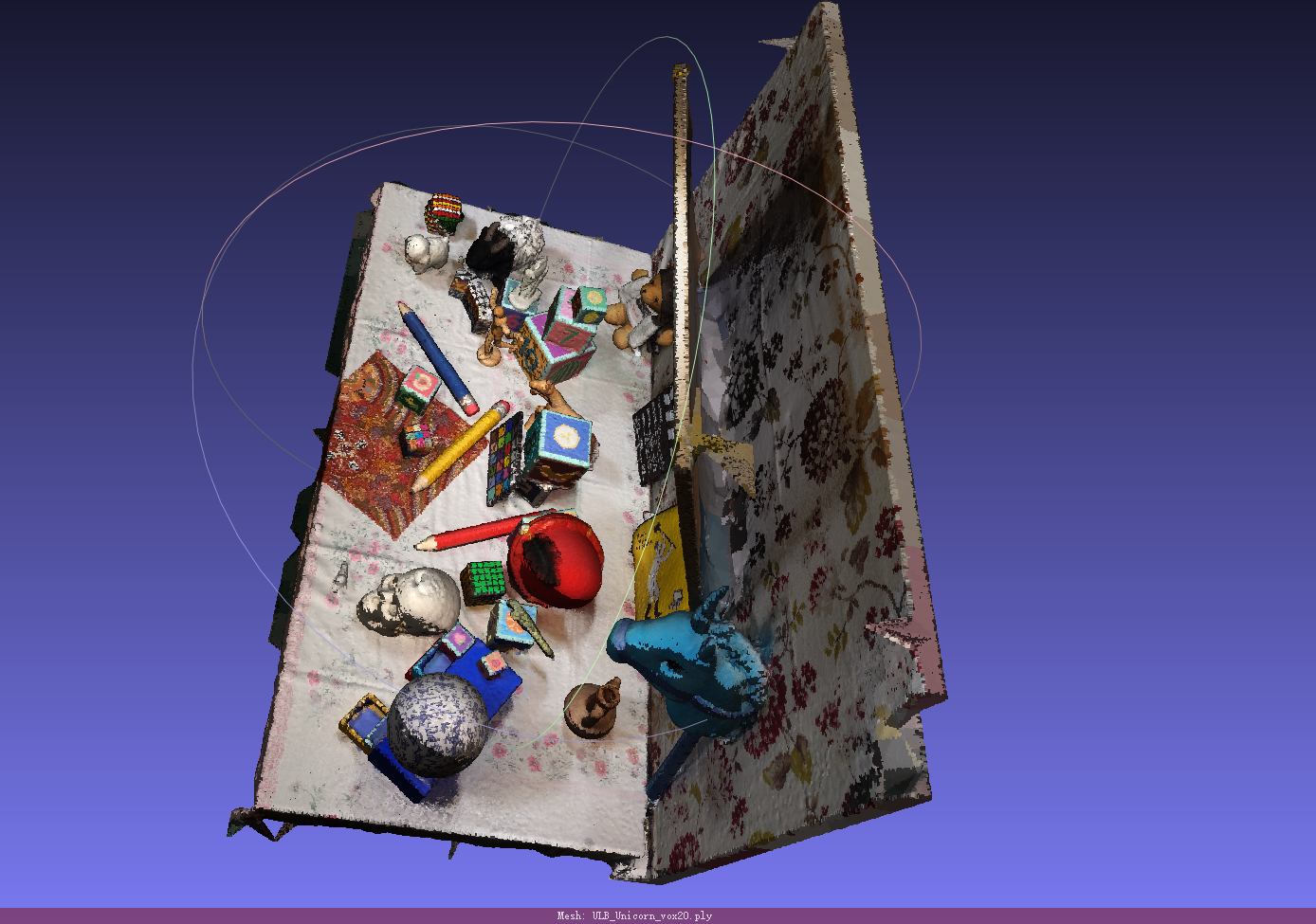
|  |  |
| --- | --- |
| egyptian\_mask\_vox12 | -0.2% |
| palazzo\_carignano\_dense\_vox14 | -0.4% |
| stanford\_area\_2\_vox16 | -1.2% |
| stanford\_area\_4\_vox16 | -1.4% |
| ulb\_unicorn\_hires\_vox15 | -0.2% |
| ulb\_unicorn\_vox13 | -0.6% |

* scant类型中：

|  |  |
| --- | --- |
| egyptian\_mask\_vox20 | -0.2% |
| landscape\_00014\_vox20 | -0.1% |
| palazzo\_carignano\_dense\_vox20 | -0.4% |
| stanford\_area\_2\_vox20 | -1.2% |
| stanford\_area\_4\_vox20 | -1.4% |
| ulb\_unicorn\_hires\_vox20 | -0.2% |
| ulb\_unicorn\_vox20 | -0.8% |

增益较大的stanford\_area、ulb\_unicorn序列，都是有较多平坦区域的点云序列。

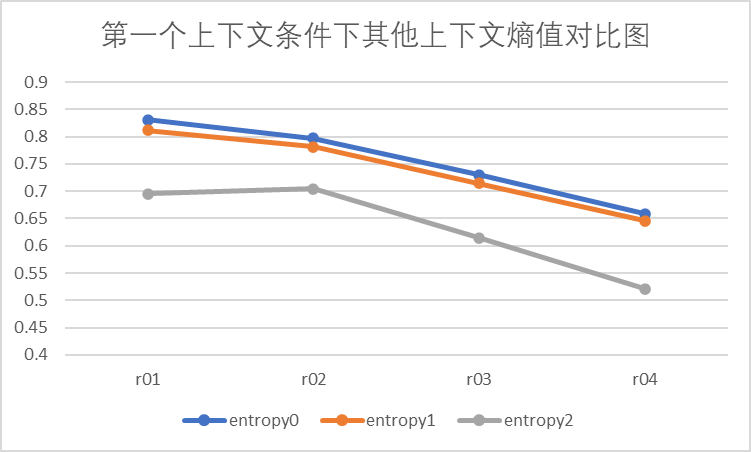
ulb\_unicorn:



**位置信息第一bit位现有上下文排列结果**

**测试过程与熵值结果对比**

1. 选取现有上下文顺序中排在最前面的missedCloseStart作为第一个上下文，然后在此条件下其他上下文熵值有：

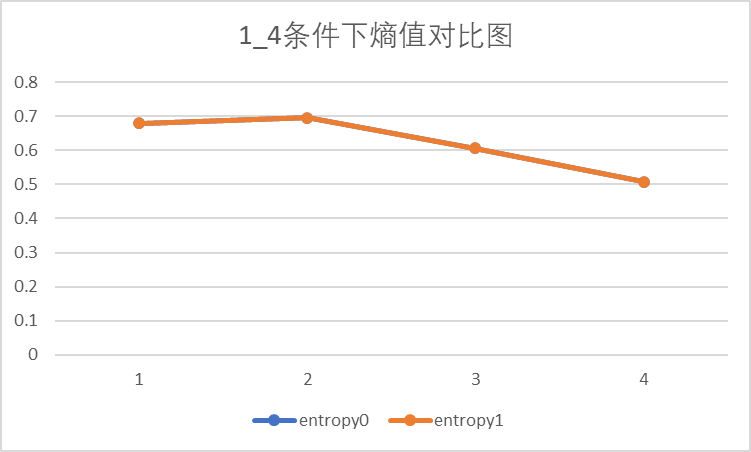


其中：

|  |
| --- |
| C++ int ctx1\_2 = (missedCloseStart << 1) | (patternClosest & 1); int ctx1\_3 =(missedCloseStart << 2) | direction; int ctx1\_4 = (missedCloseStart << 5) | (patternClose & (0b00011111)); |

由于entropy2熵值最小，选取patterClose&（0b00011111）作为第二个上下文

1. 在第一和第二上下文条件下，剩下两个上下文熵值有：



其中：

|  |
| --- |
| C++ int ctx1\_4\_2 = (ctx1\_4 << 1) | (patternClosest & 1); int ctx1\_4\_3 = (ctx1\_4 << 2) |direction; |

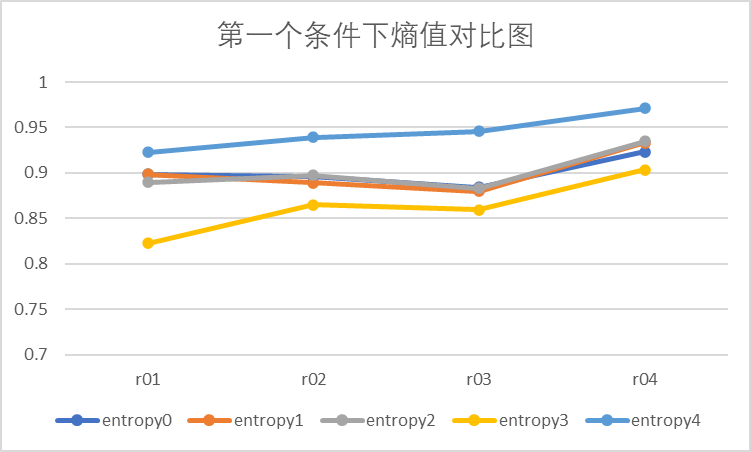
由图可知，剩下两个上下文在1\_4条件下，熵值一致，由于patterClose&（0b00011111）已经使用，那么从物理意义上来看，(patternClosest & 1)的优先级应该大于direction

|  |
| --- |
| 因此编码位置信息的高位bit时，采取的局部最优顺序为：1-4-2-3 |

**位置信息第二bit位现有上下文排列结果**

**测试过程与熵值结果对比**

1. 选取现有上下文顺序中排在最前面的missedCloseStart作为第一个上下文，其他五个上下文熵值有：

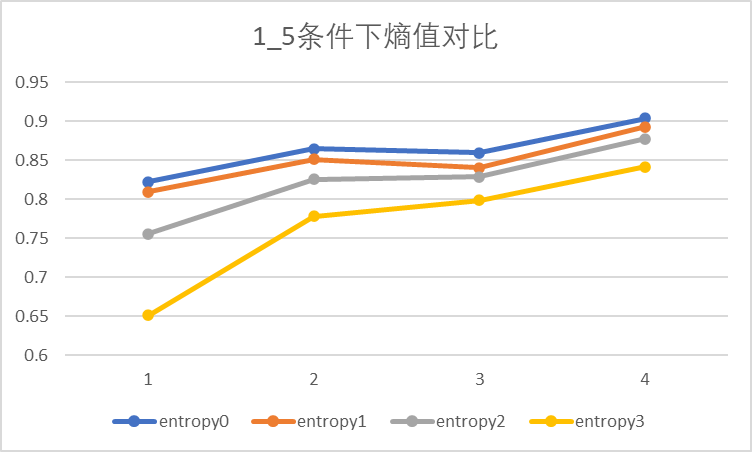


其中：

|  |
| --- |
| C++ int ctx1\_2 = (missedCloseStart << 1) | (patternClose & 1); int ctx1\_3 = (missedCloseStart << 1) | (patternClosest & 1); int ctx1\_4 = (missedCloseStart << 2) | direction; int ctx1\_5 = (missedCloseStart << 4) | (patternClose & (0b00011111)>>1); int ctx1\_6 = (missedCloseStart << 1) | orderedPclosePar; |

由图可知，entropy3最小，因此第二个上下文应该为(patternClose & (0b00011111)>>1

1. 在第一个与第二个上下文均确定以后，其他四个上下文的熵有：

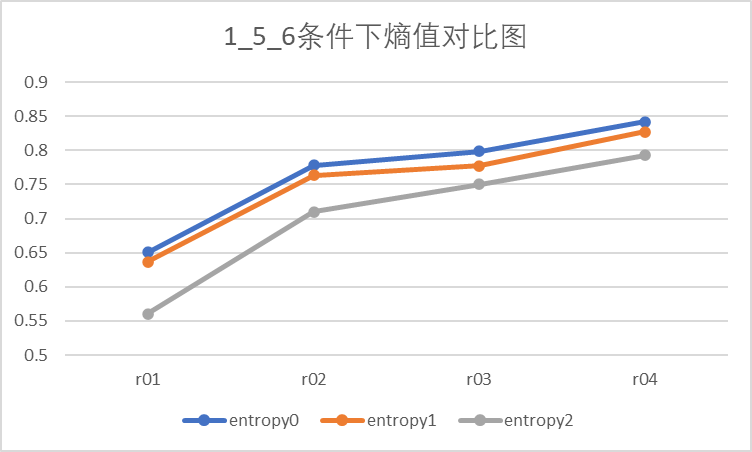


其中：

|  |
| --- |
| C++ int ctx1\_5\_2 = (ctx1\_5 << 1) | (patternClose & 1); int ctx1\_5\_3 = (ctx1\_5 << 1) | (patternClosest & 1); int ctx1\_5\_4 = (ctx1\_5 << 2) | direction; int ctx1\_5\_6 = (ctx1\_5 << 4) | orderedPclosePar; |

由图可知entropy3对应的上下文orderedPclosePar明显小于其他上下文熵值，因此选取orderedPclosePar作为第三个上下文。

1. 在前三个上下文确定好后，测剩下三个上下文的熵值，有：

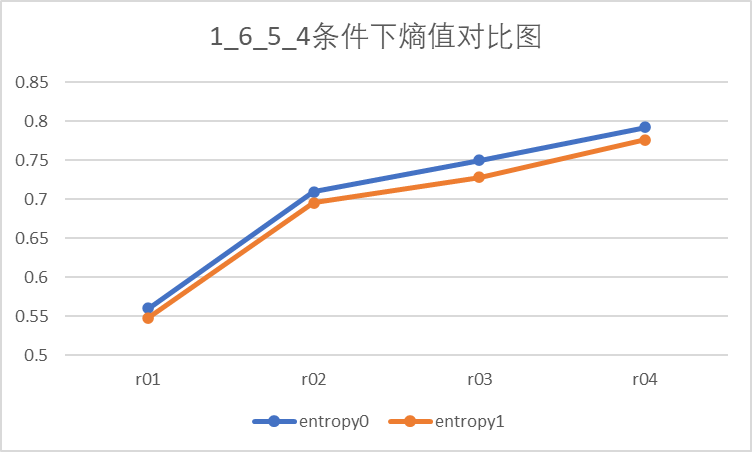


其中：

|  |
| --- |
| C++ int ctx1\_5\_6\_2 = (ctx1\_5\_6 << 1) | (patternClose & 1); int ctx1\_5\_6\_3 = (ctx1\_5\_6 << 1) | (patternClosest & 1); int ctx1\_5\_6\_4 = (ctx1\_5\_6 << 2) | direction; |

由图可知entropy2对应的上下文direction条件熵值小于其他两个上下文，因此选取direction为下一个上下文。

1. 确定前四个上下文顺序后，剩下两个上下文熵值有：



其中：

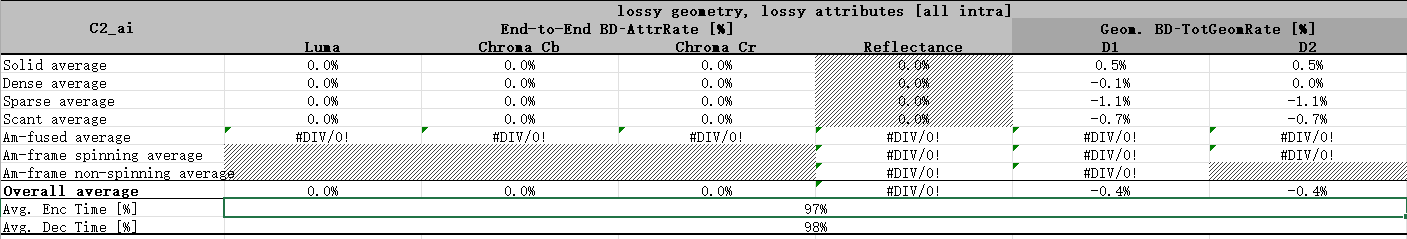
|  |
| --- |
| C++ int ctx1\_5\_6\_4\_2 = (ctx1\_5\_6\_4 << 1) | (patternClose & 1); int ctx1\_5\_6\_4\_3 = (ctx1\_5\_6\_4 << 1) | (patternClosest & 1); |

由图可知，在1\_6\_5\_4条件下，entropy1在各个码率点均小于entropy0，因此下一个上下文为patternClosest & 1，故剩下patternClose & 1作为最后一个上下文。

|  |
| --- |
| 综上，编码位置信息时，低bit位上下文顺序为：1-6-5-4-3-2 |

**性能测试结果**

将编码2bit位置信息用到的上下文顺序分别进行调整以后得到如下性能表：



其中，与修改编码Flag信息时用到的上下文顺序类似，个别序列增益很大：

* dense类型中

|  |  |  |
| --- | --- | --- |
| landscape\_00014\_vox14 | -3.5% | -3.4% |

* sparse类型中

|  |  |  |
| --- | --- | --- |
| stanford\_area\_2\_vox16 | -4.8% | -4.9% |
| stanford\_area\_4\_vox16 | -5.5% | -5.4% |

* scant类型中

|  |  |  |
| --- | --- | --- |
| landscape\_00014\_vox20 | -3.5% | -3.5% |
| stanford\_area\_2\_vox20 | -4.9% | -4.9% |
| stanford\_area\_4\_vox20 | -5.5% | -5.4% |

* Stanford:

