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：

```
int ctxMap2 = neighbEnd << 11;
ctxMap2 |= (patternClose & (0b00000110)) << 9 - 1 ; // perp that do not depend on direction = to start
ctxMap2 |= direction << 7;
ctxMap2 |= (patternClose & (0b00011000))<< 5-3; // perp that  depend on direction = to start or to end
ctxMap2 |= (patternClose & (0b00000001))<< 4;  // before
int orderedPclosePar = (((pattern >> 5) & 3) << 2) + (!!(pattern & 128) << 1) + !!(pattern & 256);
ctxMap2 |= orderedPclosePar;
```

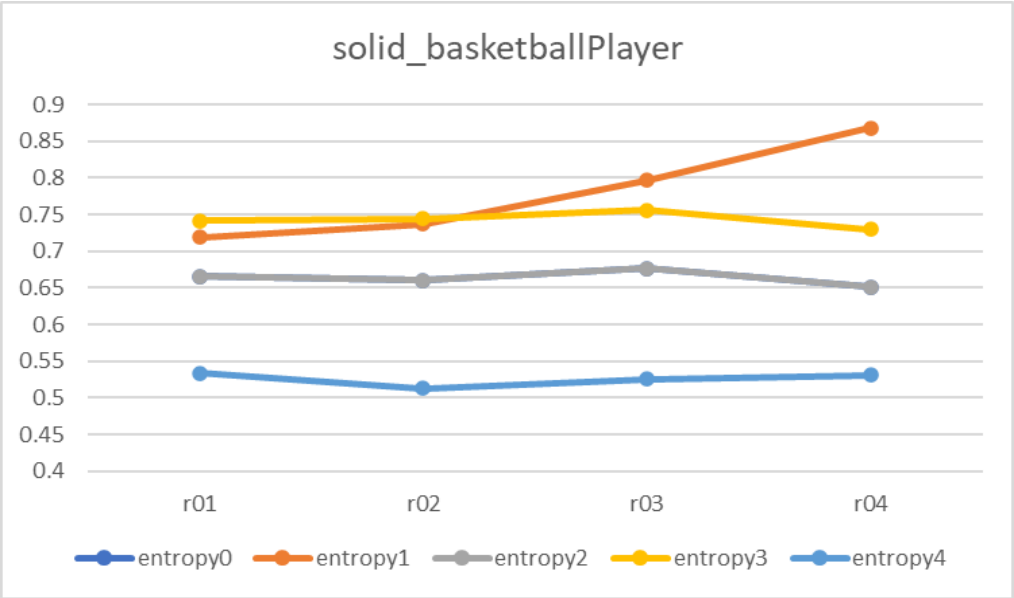
我们将上下文进行编号依次为 1、2、3、4、5、6

其物理含义分别为：

- $neighbEnd$：表示包含当前待编码边终止点的 4 个节点的占据情况
- $patternClose$：9 条边在 close 区间被占据的情况
- $direction$：当前待编码边的朝向（0=X，1=Y，2=Z）
- $pattern$：9 条边的被占据的情况
- $orderedPclosePar$： 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：

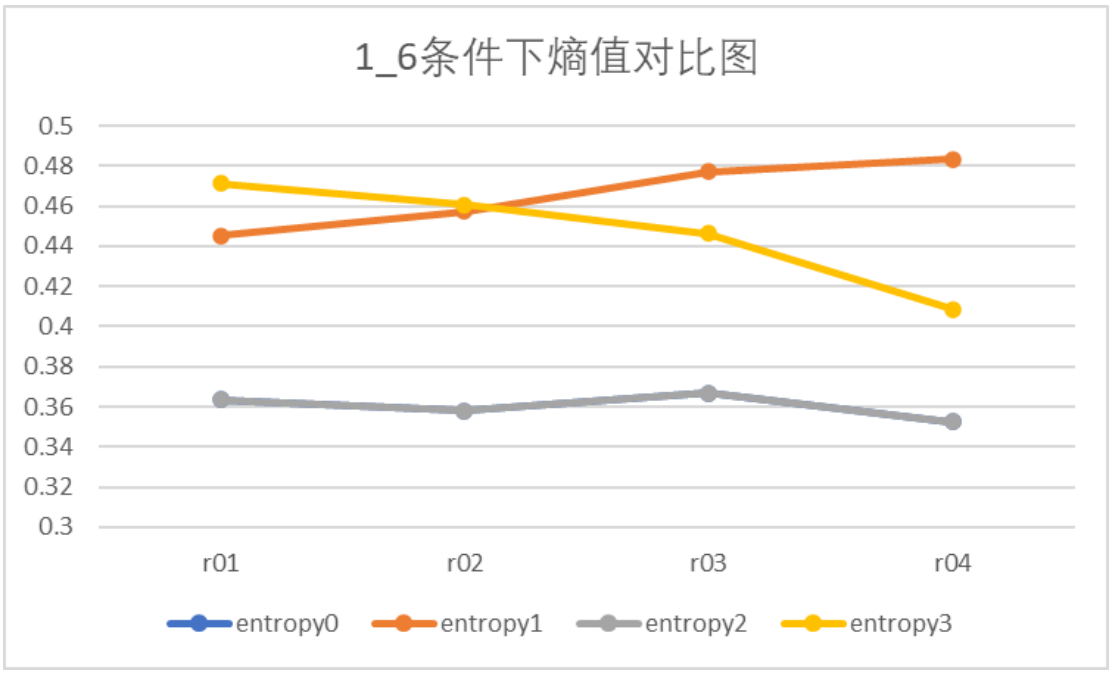


其中：

```
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

2. 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：

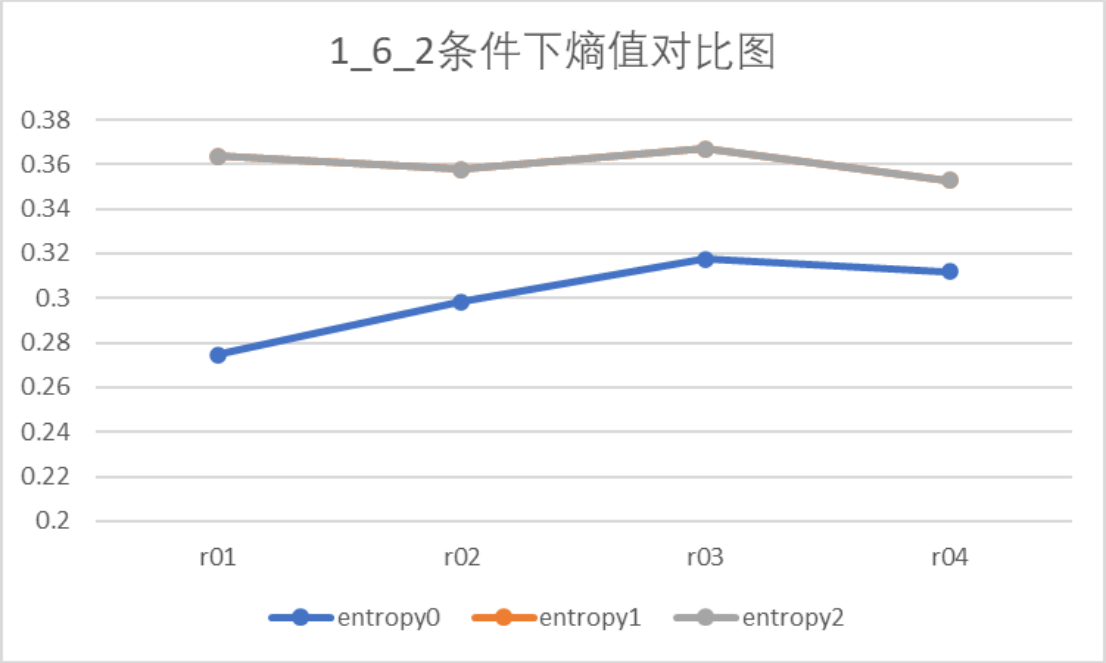


其中：

```
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

3. 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：

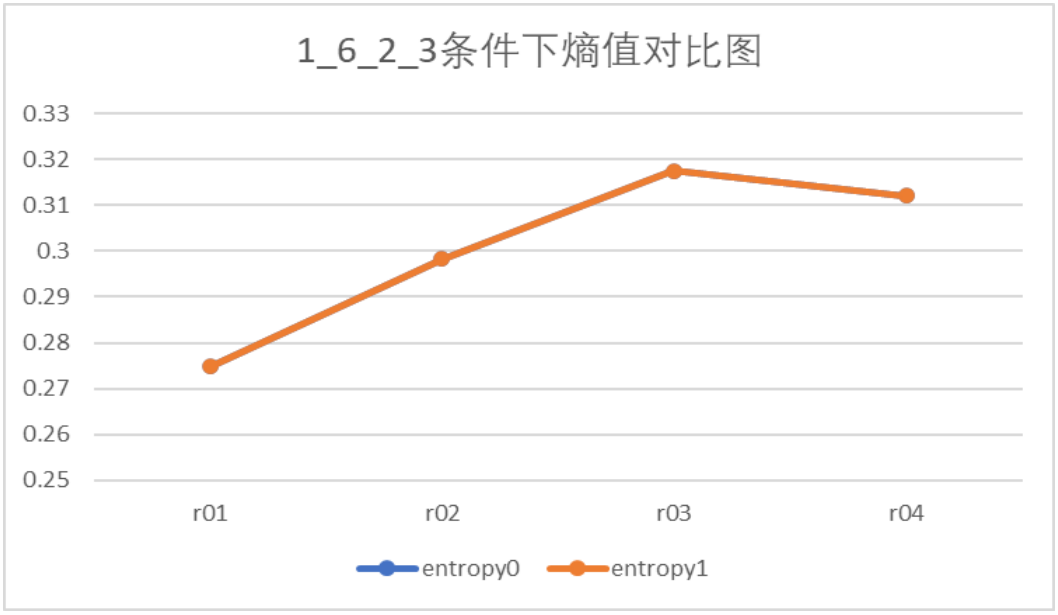


其中：

```
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

4. 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

C2_ai	lossy geometry, lossy attributes [all intra]					
	End-to-End BD-AttrRate [%]				Geom. BD-TotGeomRate [%]	
	Luma	Chroma Cb	Chroma Cr	Reflectance	D1	D2
Solid average	0.0%	0.0%	0.0%	0.0%	0.4%	0.4%
Dense average	0.0%	0.0%	0.0%	0.0%	0.3%	0.3%
Sparse average	0.0%	0.0%	0.0%	0.0%	-0.3%	-0.3%
Scant average	0.0%	0.0%	0.0%	0.0%	-0.1%	-0.1%
Am-fused average	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
Am-frame spinning average				#DIV/0!	#DIV/0!	#DIV/0!
Am-frame non-spinning average				#DIV/0!	#DIV/0!	
Overall average	0.0%	0.0%	0.0%	#DIV/0!	0.1%	0.1%
Avg. Enc Time [%]	97%					
Avg. Dec Time [%]	98%					

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%

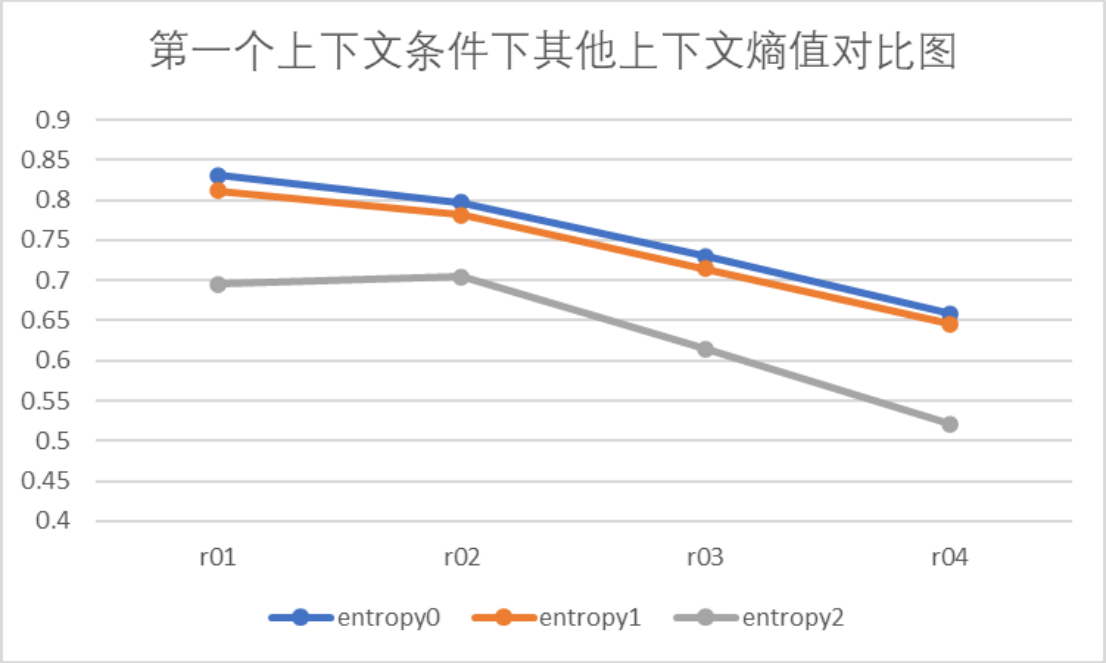
增益较大的 stanford_area、ulb_unicorn 序列，都是有较多平坦区域的点云序列。

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：

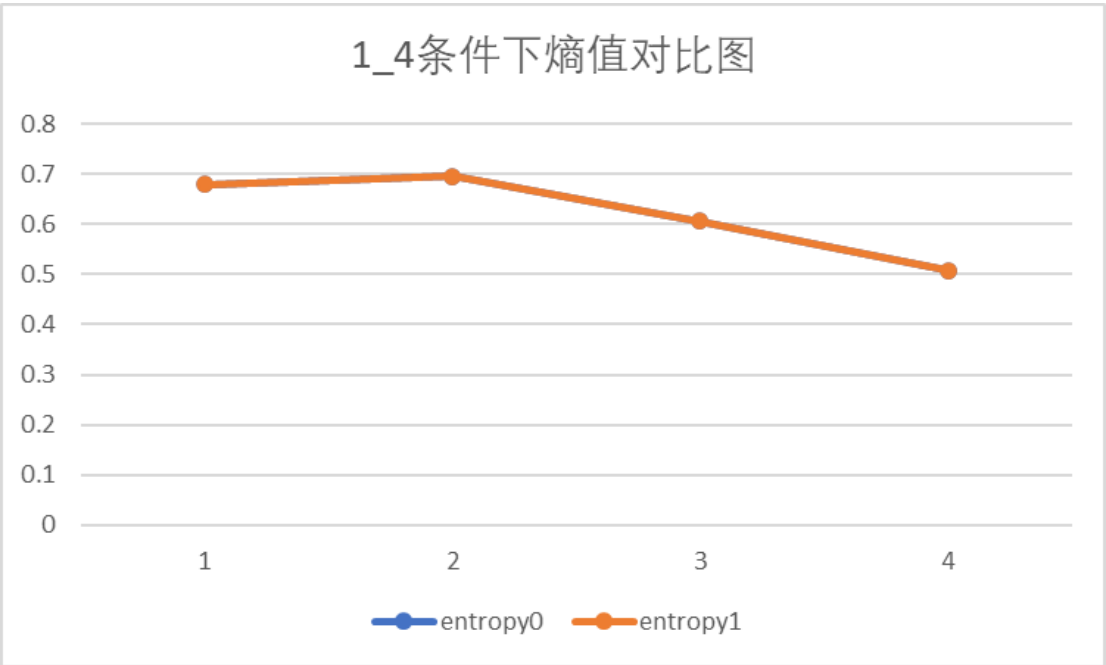


其中：

```
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

2. 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;
```

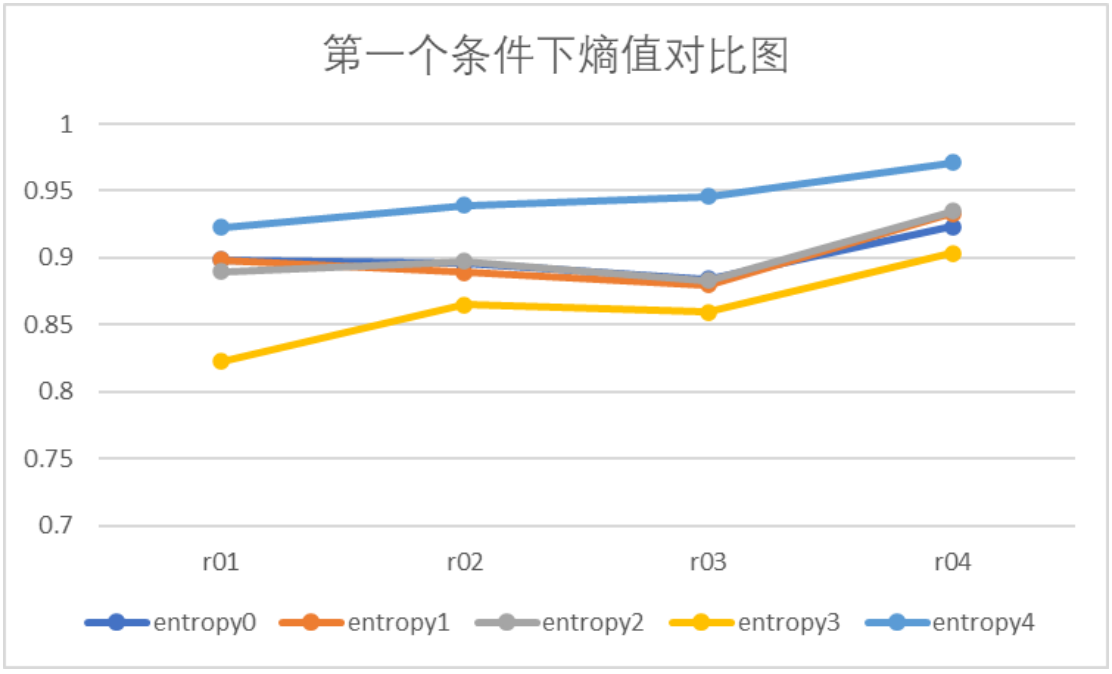
由图可知，剩下两个上下文在 1_4 条件下，熵值一致，由于 patterClose&（0b00011111）已经使用，那么从物理意义上来看，(patternClosest & 1)的优先级应该大于 direction

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：



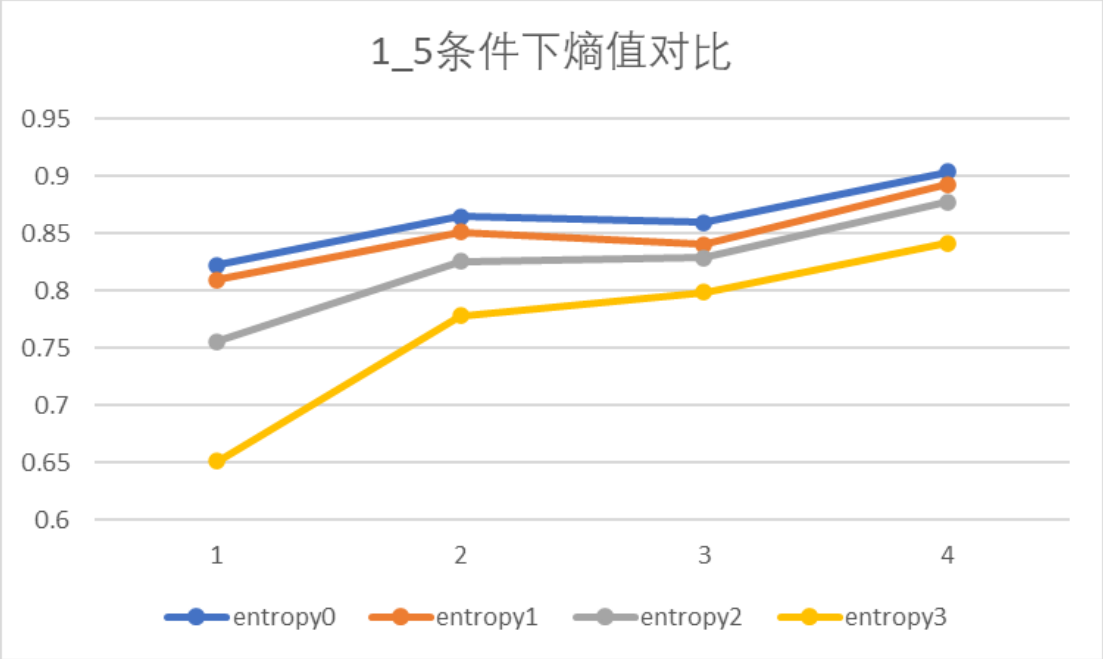
其中：

```
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

2. 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：

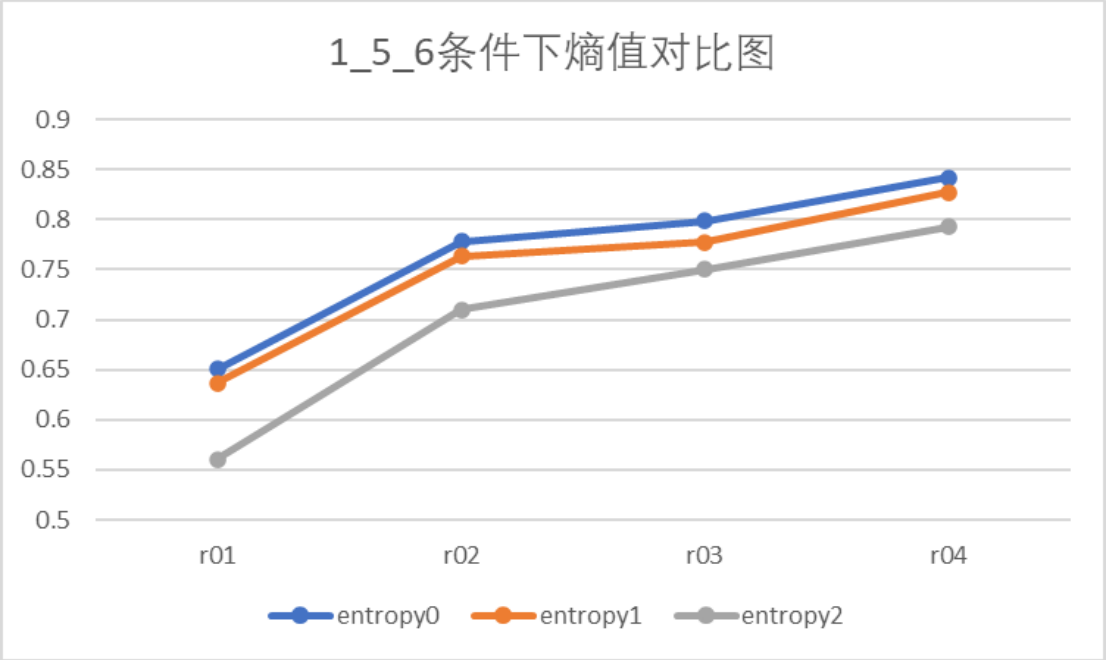


其中：

```
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。

3. 在前三个上下文确定好后，测剩下三个上下文的熵值，有：

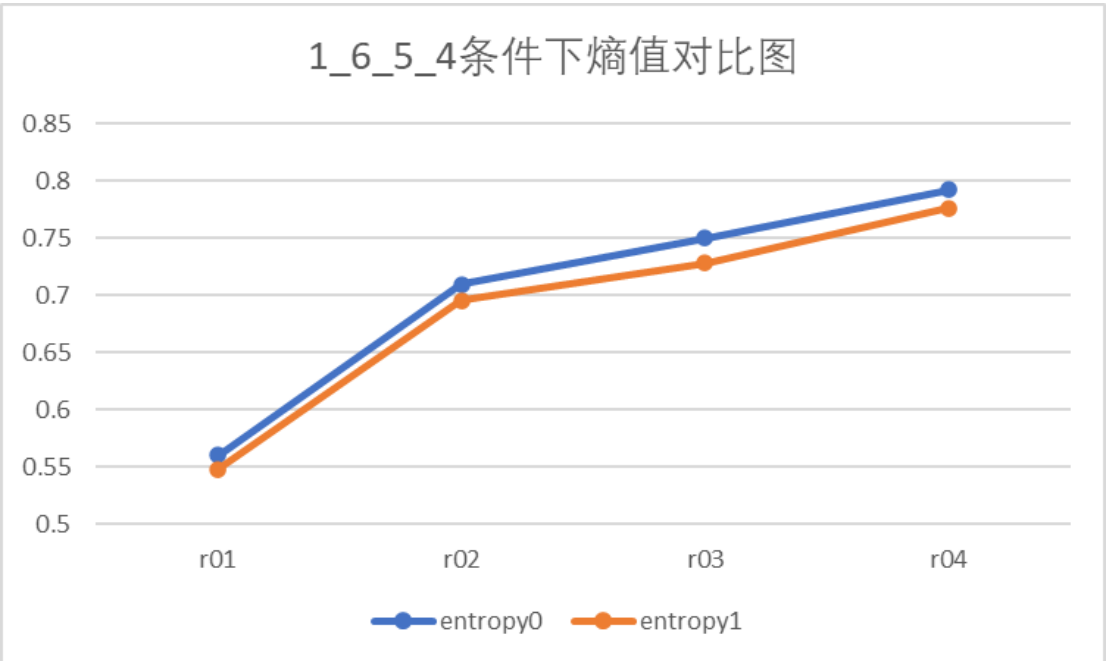


其中：

```
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。

4. 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：

C2_ai	lossy geometry, lossy attributes [all intra]					
	End-to-End BD-AttrRate [%]				Geom. BD-TotGeomRate [%]	
	Luma	Chroma Cb	Chroma Cr	Reflectance	D1	D2
Solid average	0.0%	0.0%	0.0%	0.0%	0.5%	0.5%
Dense average	0.0%	0.0%	0.0%	0.0%	-0.1%	0.0%
Sparse average	0.0%	0.0%	0.0%	0.0%	-1.1%	-1.1%
Scant average	0.0%	0.0%	0.0%	0.0%	-0.7%	-0.7%
Am-fused average	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
Am-frame spinning average				#DIV/0!	#DIV/0!	#DIV/0!
Am-frame non-spinning average				#DIV/0!	#DIV/0!	
Overall average	0.0%	0.0%	0.0%	#DIV/0!	-0.4%	-0.4%
Avg. Enc Time [%]	97%					
Avg. Dec Time [%]	98%					

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: