ICS 35.040
CCS L 71

团 体 标 准

T/AI XXX.XX—XXXX

信息技术 视觉特征编码 第 6 部分：结构点序列

Information Technology – Visual Feature Coding

Part 6: Structure Point Sequence

点击此处添加与国际标准一致性程度的标识

(征求意见稿)

(在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上)

XXXX - XX - XX 发布　　XXXX - XX - XX 实施

中关村视听产业技术创新联盟　发布

目　　次

前　言

本文件按照 GB/T 1.1-2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

本文件是T/AI XX.XX-XXXX《信息技术 视觉特征编码》的第6部分。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由新一代人工智能产业技术创新战略联盟AI标准工作组提出。

本文件由中关村视听产业技术创新联盟归口。

本部分起草单位：上海交通大学，鹏城实验室，北京大学，博云视觉（北京）科技有限公司，青岛海信网络科技股份有限公司，青岛新一代人工智能技术研究院

本部分起草人：林巍峣，刘鸣洲，陈一航，段凌宇，陈杰，高雪松，张四海，王雯雯，熊红凯，赵海英，崔晓冉。

引　言

《信息技术 视觉特征编码》拟由6个部分构成。

——第1部分：系统；

——第2部分：手工设计特征；

——第3部分：深度学习特征；

——第4部分：深度特征图；

——第5部分：语义分割图；

——第6部分：结构点序列。

本文件的发布机构提请注意，声明符合本文件时，可能涉及到如下3项专利的使用。专利名称如下：

CN111641830A，CN103517073A，CN106295561A ；

本文件的发布机构对于该专利的真实性、有效性和范围无任何立场。

该专利持有人已向本文件的发布机构保证，他愿意同任何申请人在合理且无歧视的条款和条件下，就专利授权许可进行谈判。该专利持有人的声明已在本文件的发布机构备案，相关信息可以通过以下联系方式获得：

联 系 人： 林巍峣

通讯地址： 上海市闵行区东川路800号上海交通大学闵行校区

邮政编码：200240

电子邮件：hellomikelin@gmail.com

电话：

传真：

网址：https://weiyaolin.github.io

请注意除上述专利外，本文件的某些内容仍可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

信息技术 视觉特征编码
第 6 部分：结构点序列

1 范围

本标准规范了视频中的多种结构点序列的表征格式, 设计了对于结构点序列进行无损编码与解码的工具。

本部分适用于视频数据中的运动目标行为识别, 运动目标行为分析, 汽车辅助驾驶等应用。

2 规范性引用文件

本文件不包含规范性引用文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1

结构体 structure body

一段视频帧中用于被提取特征的实体对象（如人脸、汽车、人体、物体等）。

3.2

结构点 structure point

记录结构体特征信息的坐标点（如人脸特征点、汽车3D识别框角点、人体骨架特征点、物体定位点等）。

3.3

结构点序列 structure point sequence

描述一帧中某一结构体所有结构点坐标的序列。

3.4

中心结构点 central structure point

在结构体中最靠近其几何中心位置的一个结构点。

3.5

父结构点 parent structure point

存在于父子结构点关系对中。编码时位置信息被用于参考的结构点为父结构点。在后文的图示中，父子结构点使用箭头连接，箭头起点为父结构点，箭头终点为子结构点。

3.6

子结构点 child structure point

存在于父子结构点关系对中。编码时参考父结构点位置信息的结构点为子结构点。在后文的图示中，父子结构点使用箭头连接，箭头起点为父结构点，箭头终点为子结构点。

3.7

检测 detection

对于结构体中的结构点进行获取的过程。

3.8

跟踪 track

对视频帧中的结构体进行跟随的过程。

3.9

像素 pixel

原始图像或转换图像的最小单元，每个像素包含空间坐标信息及其亮度和色度值。

3.10

模式 model

对某一结构体中某个结构点采用的编码方式。

3.11

关键帧 intra frame

用于进行帧内空间自差分编码的帧，该类型的帧在编解码时不参考其他帧的信息。关键帧的出现频率可由关键帧间隔来指定，关键帧间隔的定义见章节6.2。

3.12

非关键帧 inter frame

用于进行帧间差分编码的帧，该类型的帧在编解码时参考已解码帧的信息。关键帧以外的帧是非关键帧。

3.13

参考帧 reference frame

若当前帧是非关键帧，那么前一帧为其参考帧。

3.14

前序参考帧 former reference frame

若当前帧与前一帧都是非关键帧，那么前一帧的参考帧为当前帧的前序参考帧。

4 缩略语

下列缩略语适用于本文件。

2D：二维

3D：三维

ID：视频帧特定结构体的编号

MV：运动矢量

5 约定

5.1 概述

本部分中使用的数学运算符和优先级参照C语言。但对整型除法和算术移位操作进行了特定定义。除特别说明外，约定编号和计数从0开始。

5.2 算术运算符

算术运算符定义见表1。

表 1 算术运算符定义

算术运算符	定义
+	加法运算
-	减法运算（二元运算符）或取反（一元前缀运算符）
×	乘法运算

a^b	幂运算，表示a的b次幂。也可表示上标
/	整除运算，沿向0的取值方向截断。例如，7/4和-7/-4截断至1，-7/4和7/-4截断至-1
÷	除法运算，不做截断或四舍五入
$\frac{a}{b}$	除法运算，不做截断或四舍五入
$\sum_{i=a}^{b} f(i)$	自变量i取由a到b（含b）的所有整数值时，函数 $f(i)$ 的累加和
a % b	模运算，a除以b的余数，其中a与b都是正整数
Sqrt	开平方运算
算术运算符	定义
Arctg	反正切运算
Arcsin	反正弦运算
Arccos	反余弦运算
Abs	取绝对值
Max	取较大值
Min	取较小值
tan	正切运算

5.3 逻辑运算符

逻辑运算符定义见表2。

表 2 逻辑运算符定义

逻辑运算符	定义
a&&b	a和b之间的与逻辑运算
a ‖ b	a和b之间的或逻辑运算
!	逻辑非运算

5.4 关系运算符

关系运算符定义见表3。

表 3 关系运算符定义

关系运算符	定义
>	大于
>=	大于或等于
<	小于
<=	小于或等于
==	等于

!=	不等于

5.5 位运算符

位运算符定义见表4。

表 4 位运算符定义

位运算符	定义
&	与运算
|	或运算
~	取反运算
a>>*b*	将*a*以2的补码整数表示的形式向右移*b*位。仅当*b*取正数时定义此运算
a<<*b*	将*a*以2的补码整数表示的形式向左移*b*位。仅当*b*取正数时定义此运算

5.6 赋值

赋值运算定义见表5。

表 5 赋值运算定义

赋值运算	定义
=	赋值运算符
++	递增，*x*++相当于*x*= *x*+ 1。当用于数组下标时，在自加运算前先求变量值
--	递减，*x*--相当于*x*=*x* – 1。当用于数组下标时，在自减运算前先求变量值
+=	自加指定值，例如*x* += 3相当于*x* = x + 3，*x* += (-3)相当于*x* = *x* + (-3)
-=	自减指定值，例如*x*-= 3相当于*x* = *x*- 3,，*x*-= (-3)相当于*x* = *x*- (-3)

5.7 位流语法、解析过程和解码过程的描述方法

位流语法描述方法类似C语言。位流的语法元素使用粗体字表示，每个语法元素通过名字（用下划线分割的英文字母组，所有字母都是小写）、语法和语义来描述。语法表和正文中语法元素的值用常规字体表示。

某些情况下，可在语法表中应用从语法元素导出的其他变量值，这样的变量在语法表或正文中用不带下划线的小写字母和大写字母混合命名。大写字母开头的变量用于解码当前以及相关的语法结构，也可用于解码后续的语法结构。小写字母开头的变量只在它们所在的小节内使用。

语法元素值的助记符和变量值的助记符与它们的值之间的关系在正文中说明。在某些情况下，二者等同使用。助记符由一个或多个使用下划线分隔的字母组表示，每个字母组以大写字母开始，也可包括多个大写字母。

位串的长度是4的整数倍时，可使用十六进制符号表示。十六进制的前缀是“0x”，例如“0x1a”表示位串“0001 1010”。

条件语句中0表示FALSE，非0表示TRUE。

语法表描述了所有符合本部分的位流语法的超集，附加的语法限制在相关条中说明。

描述语法的伪代码例子见表6。当语法元素出现时，表示从位流中读一个数据单元。

表 6 语法描述的伪代码

伪代码	描述符
/*语句是一个语法元素的描述符，或者说明语法元素的存在、类型和数值，下面给出两个例子。*/	
syntax_element	ue(v)
conditioning statement	
/*花括号括起来的语句组是复合语句，在功能上视作单个语句。*/	
{	
statement	
…	
}	
/*“while”语句测试condition是否为TRUE，如果为TRUE，则重复执行循环体，直到condition不为TRUE。*/	
while (condition)	
statement	
/*“do … while”语句先执行循环体一次，然后测试condition是否为TRUE，如果为TRUE，则重复执行循环体，直到condition不为TRUE。*/	
do	
statement	
while (condition)	
/*“if … else”语句首先测试condition，如果为TRUE，则执行primarY语句，否则执行alternative语句。如果alternative语句不需要执行，结构的“else”部分和相关的alternative语句可忽略。*/	
if (condition)	
primarY statement	
else	
alternative statement	
/*“for”语句首先执行initial语句，然后测试condition，如果conditon为TRUE，则重复执行primarY语句和subsequent语句直到condition不为TRUE。*/	
for (initial statement; condition; subsequent statement)	
primarY statement	

解析过程和解码过程用文字和类似 C 语言的伪代码描述。

5.8 函数

5.8.1 概述

以下函数用于语法描述。假定解码器中存在一个位流指针，这个指针指向位流中要读取的下一个二进制位的位置。函数由函数名及左右圆括号内的参数构成。函数也可没有参数。

StructureSequenceExtension()

结构点序列编码的入口函数。具体过程的定义见章节 6.1。

InitEncodeOrder()

从结构点的空间依赖关系数组中解析结构点编解码顺序数组。具体过程的定义见章节 6.1。结构点的空间依赖关系数组见章节 7.1.3。

WriteBoneExist()

编码当前结构体中每个结构点的存在性。具体过程的定义见章节 6.1。

SpatialEncode()

空间自差分模式编码，具体过程的定义见章节 6.1。

BestModeChoose()

根据 A 矩阵的数值选取当前结构点最优的编码模式，对于待编码的结构点 j，如果其 A 矩阵的数值全为 0 则采用基于运动矢量的帧间差分模式，否则根据 A 矩阵的最大值所在索引相应地采用基于运动矢量的帧间差分模式，基于运动矢量的相对帧间差分模式，基于线性预测的帧间差分模式或基于中值预测的帧间差分模式。A 矩阵的定义见章节 7.3。帧间编码模式的原理见章节 7.2.2.2 至章节 7.2.2.5，具体过程的定义见章节 6.1。

MVEncode()

基于运动矢量的帧间差分模式编码，具体过程的定义见第章节 6.1。

MVREncode()

基于运动矢量的相对帧间差分模式编码，具体过程的定义见章节 6.1。

LinearEncode()

基于线性预测的帧间差分模式编码，具体过程的定义见章节 6.1。

MedianEncode()

基于中值预测的帧间差分模式编码，具体过程的定义见章节 6.1。

Mid()

选取输入的三个数据中，数值居中的一个。

5.9 描述符

描述符表示不同语法元素的解析过程，见表7。

表 7 描述符

描述符	说明
ue(v)	无符号哥伦布编码
se(v)	有符号哥伦布编码

fx(v)	定长编码

5.10 保留、禁止和标记位

本部分定义的位流语法中，某些语法元素的值被标注为“保留”（reserved）或“禁止”（forbidden）。

“保留”定义了一些特定语法元素值用于将来对本部分的扩展。这些值不应出现在符合本部分的位流中。

“禁止”定义了一些特定语法元素值，这些值不应出现在符合本部分的位流中。

“标记位”（marker_bit）指该位的值应为‘1’。

位流中的“保留位”（reserved_bits）表明保留了一些语法单元用于将来对本部分的扩展，解码处理应忽略这些位。“保留位”不应出现从任意字节对齐位置开始的21个以上连续的‘0’。

6 语法和语义

6.1 结构点序列语法

结构点序列编码语法见表8。此表包含了结构点序列在编码时的整体框架，其调用的函数及其编码的语义。

表 8 结构点序列编码定义

结构点序列编码的入口函数	描述符
StructureSequenceExtension( ){	
version_ID	ue(v)
model_ID	ue(4)
frame_skip	ue(v)
intra_frame_ratio	ue(v)
key_point_num	ue(v)
key_point_dim	ue(v)
spatial_reference_order	se(v)
encode_order = InitEncodeOrder(spatial_reference_order)	
for(frame_num=0;frame_num<total_frame_num;frame_num+=frame_skip+1){	
if(frame_num % intra_frame_ratio == 0{ /*关键帧，用空间自差分编码*/	
for(pos_info_buffer=0; pos_info_buffer<total_pos_in_frame; pos_info_buffer++){	
WriteBoneExist( )	
SpatialEncode( )	
}	
}	
else{ /*非关键帧*/	
for(pos_info_buffer=0; pos_info_buffer<total_pos_in_frame; pos_info_buffer++){	
WriteBoneExist( )	

pos_info_buffer.new_rect	fx(1)
if(pos_info_buffer.new_rect==1)	
SpatialEncode( )	
else{	
pos_info_buffer.not_move	fx(1)
if(pos_info_buffer.not_move==1)	
continue	
for(point;point<key_point_num;point++){	
point_to_be_encode = encode_order[point]	
encode_mode = BestModeChoose( )	
if(encode_mode == $(0001)_2$)	
MVEncode( )	
if(encode_mode == $(0010)_2$)	
MVREncode( )	
if(encode_mode == $(0100)_2$)	
LinearEncode( )	
if(encode_mode == $(1000)_2$)	
MedianEncode( )	
}	
}	
}	
}	
}	
}	

结构点编码顺序数组解析函数如下表所示。此表通过输入的结构点空间依赖关系数组spatial_reference_order[]中的标识符号-1与-2来解析得到结构点的编码顺序数组encode_order[]。其关系见章节7.1.3，解析过程见表9。

表 9 结构点编码顺序数组解析函数

结构点编码顺序数组解析函数	
InitEncodeOrder( ){	
while (spatial_reference_order[point_parent + 1] != -2) {	
if (spatial_reference_order[point_parent] == -1) {	
point_parent++;	
continue;	
while (spatial_reference_order[point_child + 1] != -2) {	
point_child++;	
if (spatial_reference_order[point_child] == -1)	
if (flag == 0) {	
flag = 1;	
continue;	
}	

else	
break;	
encode_order[j++] = spatial_reference_order[point_parent];	
point_parent++;	
}	
}	

空间自差分模式编码定义见表 10。对于中心结构点，直接编码其位置坐标 point_location；对于非中心结构点，编码其相对于其父结构点的位置残差值 res。

表 10 空间自差分模式编码定义

空间自差分模式编码定义	
SpatialEncode(){	
for(point=0;point<key_point_num;point++) {	
point_to_be_encode = encode_order[point]	
if(point_to_be_encode == encode_order[0])	
point_location	se(v)
else{	
res = point_to_be_encode - parent_structure_point	se(v)
}	
}	
}	

基于 MV 的帧间差分模式编码定义见表 11。对于中心结构点，编码其相对于参考帧的运动矢量 MV_central；对于非中心结构点，先借助 MV_central 得到其预测位置，再编码残差值。

表 11 基于 MV 的帧间差分模式编码定义

基于 MV 的帧间差分模式编码定义	
MVEncode(){	
if(point_to_be_encode == encode_order[0])	
MV_central	se(v)
else{	
pred_MV = point_to_be_encode_ref + MV_central	
res = point_to_be_encode – pred_MV	se(v)
}	
}	

基于 MV 的帧间差分模式编码定义见表 12。对于中心结构点，编码其相对于参考帧的运动矢量 MV_central；对于非中心结构点，先后借助 MV_central 和其父结构点残差 res_parent 来预测其位置，再编码残差值。

表 12 基于 MV 的相对帧间差分模式编码定义

基于 MV 的相对帧间差分模式编码定义	
MVREncode(){	
if(point_to_be_encode == encode_order[0])	
MV_central	se(v)
else{	
pred_MV = point_to_be_encode_ref + MV_central	
pred_MVR = pred_MV + res_parent	
res = point_to_be_encode – pred_MVR	se(v)
}	
}	

基于线性预测的帧间差分模式编码定义见表13。结构点的预测值由其在前序参考帧和参考帧的位置做线性延伸得到。

表 13 基于线性预测的帧间差分模式编码定义

基于线性预测的帧间差分模式编码定义	
LinearEncode(){	
MV_ref = point_to_be_encode_ref - point_to_be_encode_ref_ref	
pred_li = point_to_be_encode_ref + MV_ref	
res = point_to_be_encode – pred_li	se(v)
}	

基于中值预测的帧间差分模式编码定义见表 14。在该模式中，采用前三种帧间差分模式预测值的中值作为最终预测值。

表 14 基于中值预测的帧间差分模式编码定义

基于中值预测的帧间差分模式编码定义	
MedianEncode(){	
pred_me = Mid(pred_MV, pred_MVR, pred_li)	
res = point_to_be_encode – pred_me	se(v)
}	

结构点存在性编码定义见表 15。如当前结构体所有结构点都存在, 则 all_point_exist_flag 置 1；否则逐结构点编码其存在性。

表 15 结构点存在性编码定义

结构点存在性编码定义	
WriteBoneExist(){	
all_point_exist_flag	fx(1)
if(all_point_exist_flag == 0)	
point_exist_info	fx(1)
}	

6.2 结构点序列语义

结构点序列编码版本号 version_ID

无符号整数。在当前版本中取值为 1。

模式启用参数 model_ID

无符号整数。表示编码结构点序列时所启用的编码模式，编码时会从启用的编码模式中选择。具体取值见表 16。如果要采用多种模式混合的方式编码，则将每个模式对应的值相加即可。具体过程的定义见章节 7.4.2。

表 16 结构点序列编码模式

model_ID的值	结构点序列编码模式
$(0001)_2$	仅启用基于MV的帧间差分模式
$(0010)_2$	仅启用基于 MV 的相对帧间差分模式
$(0100)_2$	仅启用基于线性预测的帧间差分模式
$(1000)_2$	仅启用基于中值预测的帧间差分模式

跳帧个数 frame_skip

无符号整数。控制编码器跳帧的个数。在默认设置 0 下，编码所有帧的数据；设置为 1 时，隔一帧的数据进行编码；设置为 2 时，隔两帧的数据进行编码；以此类推。

关键帧间隔 intra_frame_ratio

无符号整数。标识结构点序列编码时两个关键帧的距离。

结构点个数 key_point_num

无符号整数。标识单个结构体中的结构点个数。

结构点维度 key_point_dim

无符号整数。标识结构点的坐标维度数。

空间依赖关系数组 spatial_reference_order[]

无符号整数型数组。标识单个结构体中相邻结构点之间的父子关系，用于进一步决定不同结构点的编码顺序数组，具体见章节 6.3.3 节。

编码顺序数组 encode_order[]

无符号整数型数组。根据空间依赖关系数组得出，用于从循环序号中获取要编码的结构点编号。

循环序号 point

循环中的临时变量。

视频帧号 frame_num

当前执行的视频帧号，用于判断是否为关键帧。

执行步数 total_frame_num

需要处理的视频帧总数。

结构体个数 total_pos_in_frame

表示当前视频帧中的结构体总数。

结构体 pos_info_buffer

表示当前视频帧中用于编码特征的结构体的缓存变量，其包含了结构体的多种信息。

结构体无位移标识 not_move

1 位无符号整数。如为 1 则表示结构体在当前帧相比参考帧没有发生移动，否则发生了移动。

新结构体标识 new_rect

1 位无符号整数。如为 1 则表示当前结构体 ID 是新出现的，否则不是新出现的。

残差 res

结构点坐标的真实值与其预测值之差。

结构点坐标 point_location

表示该结构点的坐标。

中心结构点运动矢量 MV_central

中心结构点由参考帧到当前帧的坐标偏移量, 用于基于 MV 的帧间差分模式和基于 MV 的相对帧间差分模式中。

当前结构点前序运动矢量 MV_ref

当前结构点由前序参考帧到参考帧的坐标偏移量。

待编码的结构点 point_to_be_encode

通过循环序号从结构点编码顺序数组映射得到的要编码的结构点。

参考点 point_to_be_encode_ref

待编码结构点在参考帧的对应点。

前序参考点 point_to_be_encode_ref_ref

待编码结构点在前序参考帧的对应点。

父结构点 parent_structure_point

编码时位置信息被用于参考的结构点。

位置预测值（从运动矢量） pred_MV

待编码结构点在基于 MV 的帧间差分模式下的预测值。

位置预测值（从相对运动矢量） pred_MVR

待编码结构点在基于 MV 的相对帧间差分模式下的预测值。

线性预测值 pred_li

待编码结构点在基于线性预测的帧间差分模式下的预测值。

中值预测值 pred_me

待编码结构点在基于中值预测的帧间差分模式下的预测值。

父结构点残差 res_parent

待编码结构点的父结构点的预测残差。

结构点存在标志 all_point_exist_flag

1 位无符号整数。是待编码的结构体中所有结构点是否都存在的标志。

结构点存在性信息 point_exist_info

由 0 和 1 构成的数组，表征待编码结构体中的结构点的存在性。1 表示对应索引的结构点存在，0 表示对应索引的结构点不存在。

7 结构点序列编码

7.1 结构点序列

7.1.1 结构点序列示例

本部分适用于结构点序列的编码。结构点是指在视觉任务中对目标事物具有特征标识作用的点，由这些点的存在性信息及坐标位置信息构成的序列即结构点序列。结构点序列在应

用中出现得十分广泛。见图1，在视频目标跟踪中，使用了2D检测框来标识车辆，此时该检测框的四个角点即可被视为结构点；在人群流量统计中，人物的2D检测框角点和骨骼点可以被视为结构点；在人脸识别中，人物的面部关键点可以被表示成结构点；同样的，运动目标行为分析中，的人物骨骼点也可以被视为结构点。

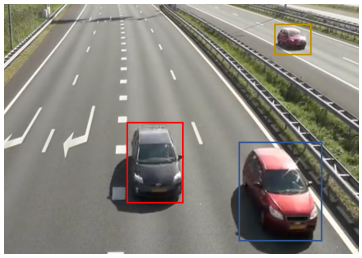
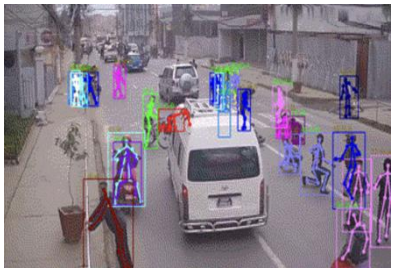
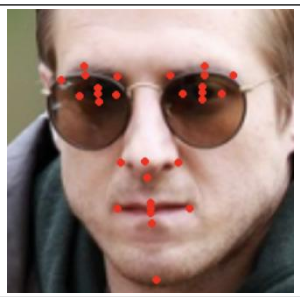
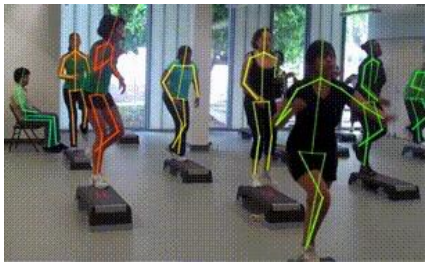
<table>
<tr><td>视频目标跟踪[5]</td><td>人群流量统计[6]</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>人脸识别[7]</td><td>运动目标行为分析[8]</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

图 1 常见的结构点序列

对于结构点序列，应首先规定该序列的以下几个特征值：

1) 当前视频帧数 total_frame_num：标识当前用于编码的视频帧帧数。

2) 当前帧中结构体的个数 total_pos_in_frame：标识当前帧中的结构体总数。

3) 结构点个数 key_point_num：标识结构体中需要编码结构点个数。

4) 结构点维度 key_point_dim：标识结构点坐标维度。

5) 结构体中结构点之间的空间依赖关系数组 spatial_reference_order[]：标识结构体中不同结构点之间的父子关系，用于决定不同结构点的编码顺序，获得方式见章节7.1.3。

6) 结构体的缺失标志数组 point_exist_info[]：用于表示某个结构体中的结构点的缺失情况，其长度和结构点数量一致，用“1”表示对应的结构点存在，用“0”表示对应的结构点不存在。

7.1.2 结构点序列的输入格式

为了能够让代码成功解析输入的结构点序列并进行编码，规定结构点序列的输入格式建表17。若需要编码有*N*帧的视频中的结构点序列，其输入是由*N*行组成的。对于每一行（帧）结构点序列，第一列为本帧的视频帧号*n*（*n*=0,1,…,*N*-1）；第二列为当前视频帧中的结构体个数，之后为本视频帧中各个结构体的信息，由ID值、缺失标志数组及其结构点位置坐标组成。

表 17 结构体序列的输入格式

视频帧号	本帧中结构体个数	首个结构体ID	首个结构体的缺失标志数组	首个结构体的结构点坐标	……	最后一个结构体ID	最后一个结构体的缺失标志数组	最后一个结构体的结构点坐标
0	10	0	0 1 … 1	X_1 Y_1…X_{13} Y_{13}	……	9	1 1,… 1	X_0 Y_0…X_{13} Y_{13}
1	10	0	1 1,… 1	X_0 Y_0…X_{13} Y_{13}	……	9	1 1,… 1	X_0 Y_0…X_{13} Y_{13}
2	11	0	1 1,… 1	X_0 Y_0…X_{13} Y_{13}	……	10	1 1,… 1	X_0 Y_0…X_{13} Y_{13}
…	…	…			……	…		
N	9	2	0 1 … 1	X_1 Y_1…X_{13} Y_{13}	……	10	1 1 … 0	X_0 Y_0…X_{12} Y_{12}

需要注意的是：为了便于区分，结构点序列中的每一个结构体均被赋予一个 ID 值。该值在该结构体第一次在视频中出现时即被确定，且在整个视频序列的各帧中均保持不变，即当结构体在视频序列中消失后重新出现时，也应当保持原有的 ID 值。一帧中不允许出现两个相同 ID 的结构体。

在某些应用场景中，会出现结构体的某些部分被物体遮挡，无法获取该部分关键点的坐标的情况。因此，每个结构体应赋予一个容量为 key_point_num 的缺失点标志数组。若结构体的某个结构点可以获取，则该标志数组中对应索引的位置为 1，否则取 0;缺失的结构点坐标省略即可。

见图 2，现有一个长度为两帧的视频，需要编码人体骨架特征点。在第一帧中，共出现两个结构体，其中 ID 为 10 的结构体出现结构点遮挡（被遮挡结构点：4)；在第二帧中，共出现一个结构体，且所有结构点均未被遮挡。其结构点序列应该为：0 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 X_0 Y_0 X_1 Y_1 X_2 Y_2 X_3 Y_3 X_4 Y_4 X_5 Y_5 X_6 Y_6 X_7 Y_7 X_8 Y_8 X_9 Y_9 X_{10} Y_{10} X_{11} Y_{11} X_{12} Y_{12} X_{13} Y_{13} 10 1 1 1 1 0 1 1 1 1 1 1 1 1 1 X_0 Y_0 X_1 Y_1 X_2 Y_2 X_3 Y_3 X_5 Y_5 X_6 Y_6 X_7 Y_7 X_8 Y_8 X_9 Y_9

X_{10} Y_{10} X_{11} Y_{11} X_{12} Y_{12} X_{13} Y_{13} 1 1 10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 X_0 Y_0 X_1 Y_1 X_2 Y_2 X_3 Y_3 X_4 Y_4 X_5 Y_5 X_6 Y_6 X_7 Y_7 X_8 Y_8 X_9 Y_9 X_{10} Y_{10} X_{11} Y_{11} X_{12} Y_{12} X_{13} Y_{13}

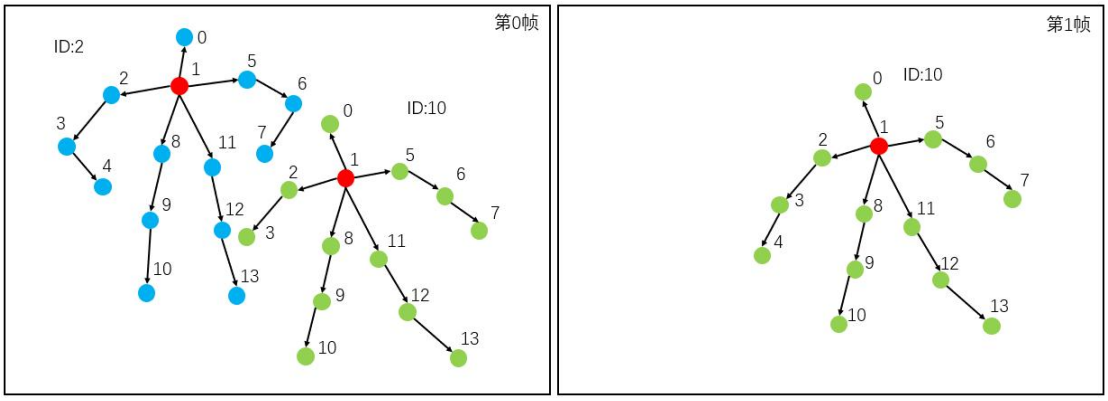


图 2 视频帧的结构点序列示例

7.1.3 结构点空间依赖关系数组

结构点空间依赖关系数组 spatial_reference_order[]是手动确定的，用于标识单个结构体中不同结构点之间的父子依赖关系，并用于进一步地决定说结构点的编码顺序。建议的确定方式如下（此处需要注意区分，结构点是本技术中的概念，结点是树中的概念）：

1) 将结构体中的中心结构点视为树的根结点，其子结构点作为树的子结点。其他结构点在树中的父子结点关系以相同方式确定。此时将此结构体的结构点序列视作一棵树。
2) 向结构点空间依赖关系数组中添加中心结构点序号，作为树的根结点。
3) 向结构点空间依赖关系数组中添加-1 作为标识符。
4) 将中心结构点视作父结点。
5) 向结构点空间依赖关系数组中按树的层次遍历顺序添加父结点的全部子结点序号。这里树的层次遍历顺序即数据结构中树结构的层次遍历顺序。
6) 向结构点空间依赖关系数组中添加-1 作为标识符，并将树的下一结点更新为父结点，下一结点的更新方向为树的层次遍历顺序。
7) 循环第 5-6 步，直到父节点为树的第一个最深层叶子结点为止。
8) 向结构点空间依赖关系数组中添加-2 作为标识符，表示结束。

此处以人体骨架特征点序列为例，具体地阐述结构点空间依赖关系数组。常见的人体骨架见图 3 (a)，由 N=14 个结构点组成，每个结构点是 2D 坐标点，中心结构点一般选取人体胸口的结构点（即 M=1）。见图 3(a)，中心结构点是 1，以其为根结点建立树，见图 3(b)。

a) 向结构点空间依赖关系数组中添加 1，添加-1。

b) 再添加 1 的子结点{0,2,8,11,5}，添加-1。

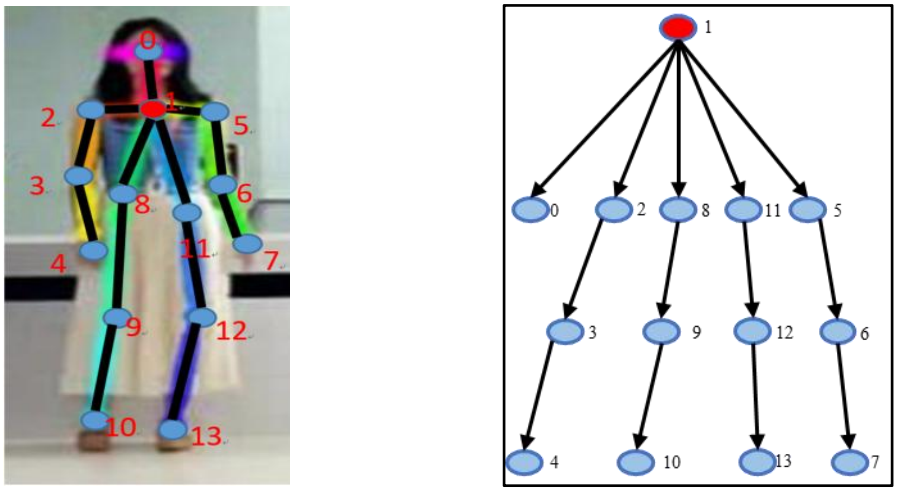
c) 父结点设置为 0，添加 0 的子结点（此处 0 没有子结点），添加-1。

d) 父结点设置为 2，添加 2 的子结点{3}，添加-1。

e) 参照步骤 4) 遍历并不断更新父结点，添加子结点，添加-1。

f) 当父结点遍历到 4 时，因为 4 为树的第一个最深层叶子结点，故向结构点空间依赖关系数组中添加-2，表示结束。

见图 3(a)，最终得到结构点的空间依赖关系数组为 spatial_reference_order[] =[1,-1,0,2,8,11,5,-1,-1,3,-1,9,-1,12,-1,6,-1,4,-1,10,-1,13,-1,7,-1,-2]。



(a) (b)

图 3 人体骨架格式：(a) 人体骨架结构点结构体，(b) 以树表示的人体骨架结构体

见图 3(b)，空间依赖关系数组的生成并不强制遵循树结构，但是更优的空间依赖关系数组可以实现更好的去冗余，以达到高效压缩的目的。理论上，在上述的标识符规则下，能够包含结构体中的所有结构点的树生成的数组均可以用作空间依赖关系数组。见图 3，最简单直接的情形，空间依赖关系数组可以按照结构点序号顺序生成为 spatial_reference_order[] =[0,-1,1,-1,2,-1,3,-1,4,-1,5,-1,6,-1,7,-1,8,-1,9,-1,10,-1,11,-1,12,-1,13,-1,-2]。

根据结构点空间依赖关系数组，可以进一步调用 InitEncodeOrder()函数得到不同结构点的编码顺序，InitEncodeOrder()函数的定义见章节 6.1。

7.2 基于多模式的编码方法

7.2.1 概述

在多模式的编码方法中，针对不同的结构体会选择不同的帧内或帧间编码，具体如下。

在关键帧中，所有结构体中的结构点均采用空间自差分模式进行编码；在非关键帧中，如果某个结构体在前一帧未出现，则其中的结构点也采用空间自差分模式进行编码。关于空间自差分编码的的具体内容见章节7.2.2.1节。

在非关键帧中，如果某个结构体在前序的帧中出现了，则对其中的结构点采用帧间编码。对于不同结构点，会根据A矩阵判断具体采用的帧间编码模式。关于帧间编码模式的具体内容见章节7.2.2.2至章节7.2.2.5，关于A矩阵的定义和计算过程见章节7.3。

本部分所采用的哥伦布编码原理参考[4]。

7.2.2 编码模式原理

7.2.2.1 帧内编码模式：空间自差分模式

该模式按照结构点的空间依赖关系数组 spatial_reference_order[]，见图 4 箭头所示的编码顺序，先利用哥伦布编码方法编码结构体的中心结构点（即图中 1 号红色点），再依次用哥伦布编码方法编码子结构点与父结构点的位置差分值。需要注意的是，父子结构点使用箭头连接，箭头起点为父结构点，箭头终点为子结构点。当父结构点为缺失点位时，编码子结构点与原点位置的差分值，原点位置即所有维度上的坐标值为 0，后同。

空间自差分编码模式利用了结构点的空间相关性，每帧可以进行独立解码，适用结构点序列中关键帧的压缩编码。

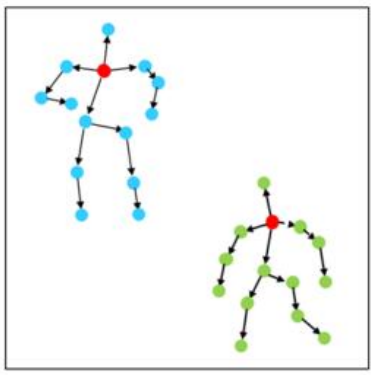


图 4 空间自差分编码

7.2.2.2 帧间编码模式：基于运动矢量的帧间差分模式

基于运动矢量的帧间差分模式是指首先用有符号哥伦布编码方法编码该结构体的中心结构点在第 t 帧（当前帧）的位置对比第 t-1 帧（参考帧）间的运动矢量(MV_central(t))，再

将 MV_central(t)应用到目标结构点以得到预测值, 再用熵编码方法编码目标结构点的预测值和真实值的残差。

所述的 MV_central(t)计算见式（1）（此处以 2D 结构点为例，后同）：

$$(MV\_central_x(t), MV\_central_y(t)) = (x_{ID}^c(t) - x_{ID}^c(t-1), y_{ID}^c(t) - y_{ID}^c(t-1)), \quad (1)$$

式中：

$MV\_central_x(t)$——该结构体的中心结构点在第 t 帧的位置对比第 t-1 帧的位置在横坐标上的 MV_central(t)分量；

$MV\_central_y(t)$——该结构体的中心结构点在第 t 帧的位置对比第 t-1 帧的位置在纵坐标上的 MV_central(t)分量；

$x_{ID}^c(t)$——第 t 帧该结构体中心结构点的横坐标值；

$y_{ID}^c(t)$——第 t 帧该结构体中心结构点的纵坐标值；

$x_{ID}^c(t-1)$——第 t-1 帧该结构体中心结构点的横坐标值；

$y_{ID}^c(t-1)$——第 t-1 帧该结构体中心结构点的纵坐标值；

ID 表示其所在结构体的编号；

c 表示此结构点为中心结构点。

以第 j 个结构点为例，首先利用第 t-1 帧该结构点的坐标位置和 MV_central(t)计算出其在第 t 帧位置的预测值，计算过程见式（2）：

$$(pred\_MVx_{ID}^j(t), pred\_MVy_{ID}^j(t)) = \{(x_{ID}^j(t-1) + MV\_central_x(t), y_{ID}^j(t-1) + MV\_central_y(t)) | j \in S_c\}. \quad (2)$$

此处规定 S_c 为当前结构体除去中心结构点后的其余结构点。预测值与第 t 帧的结构点坐标做差得到预测残差，见式（3）：

$$(resx_{ID}^j(t), resy_{ID}^j(t)) = \{(x_{ID}^j(t) - pred\_MVx_{ID}^j(t), y_{ID}^j(t) - pred\_MVy_{ID}^j(t)) | j \in S_c\}. \quad (3)$$

对此模式下的残差编解码则根据标志位使用哥伦布编解码方案，后续编码模式相同。当 t-1 帧中该结构点为缺失点位时，将原点坐标视为 t-1 帧中该结构点的坐标。

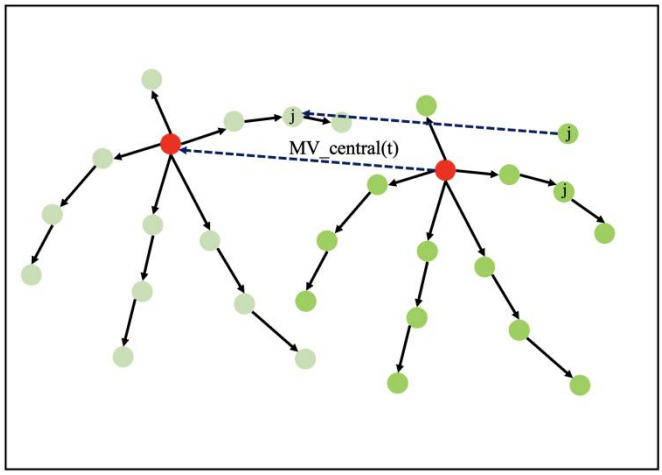


图 5 基于运动矢量的帧间差分模式

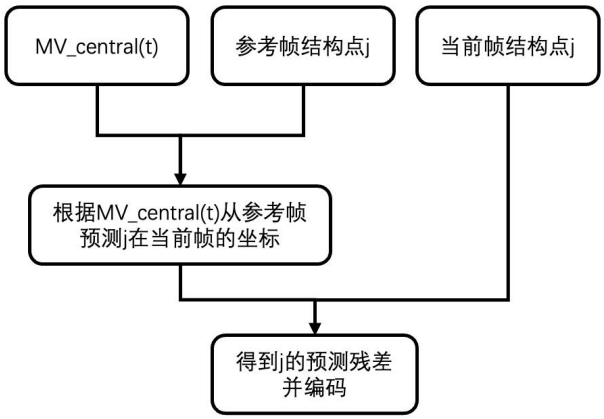


图 6 基于运动矢量的帧间差分模式的流程框图

7.2.2.3 帧间编码模式：基于运动矢量的相对帧间差分模式

基于运动矢量的相对帧间差分方法建立在基于运动矢量的帧间差分方法的基础上，充分利用结构体中不同结构点间的空间相关性。

该方法首先利用基于运动矢量的帧间差分模式得到 MV_central(t)和结构点 j 的预测值，再用其父结构点的运动残差应用到该预测值，以作为子结构点的最终预测值，再用熵编码方法编码各结构点的预测值和真实值的残差，见式（4）：

$$(pred\_MVRx_{ID}^{j}(t), pred\_MVRy_{ID}^{j}(t)) =$$

$$\{(pred\_MVx_{ID}^{j}(t) + res\_parentx_{ID}^{p(j)}(t), pred\_MVy_{ID}^{j}(t) + res\_parenty_{ID}^{p(j)}(t)) | j \in S_c\}, \quad (4)$$

式中：

p(j)——取 j 的父结构点。

预测值与第 t 帧的结构点坐标做差得到预测残差，见式（5）：

$$(resx_{ID}^{j}(t), resy_{ID}^{j}(t)) =$$

$$\{(x_{ID}^{j}(t) - pred\_MVRx_{ID}^{j}(t), y_{ID}^{j}(t) - pred\_MVRy_{ID}^{j}(t)) | j \in S_c\}. \quad (5)$$

当 t-1 帧中该结构体的某个结构点为缺失点位时，将原点坐标视为该点的坐标。

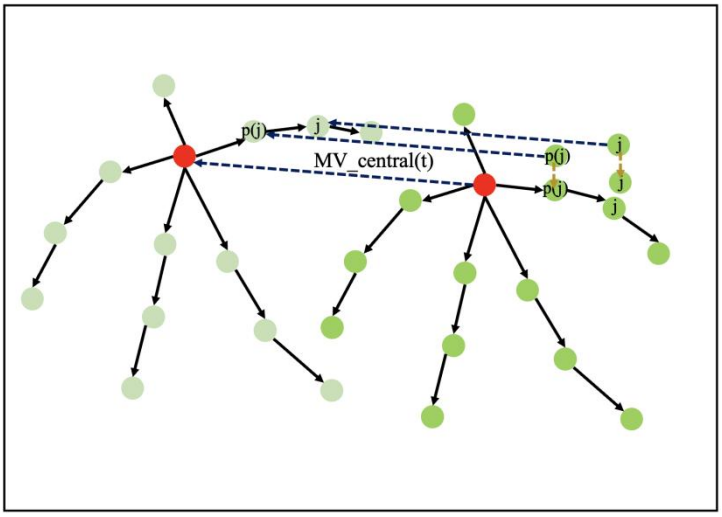


图 7 基于运动矢量的相对帧间差分模式

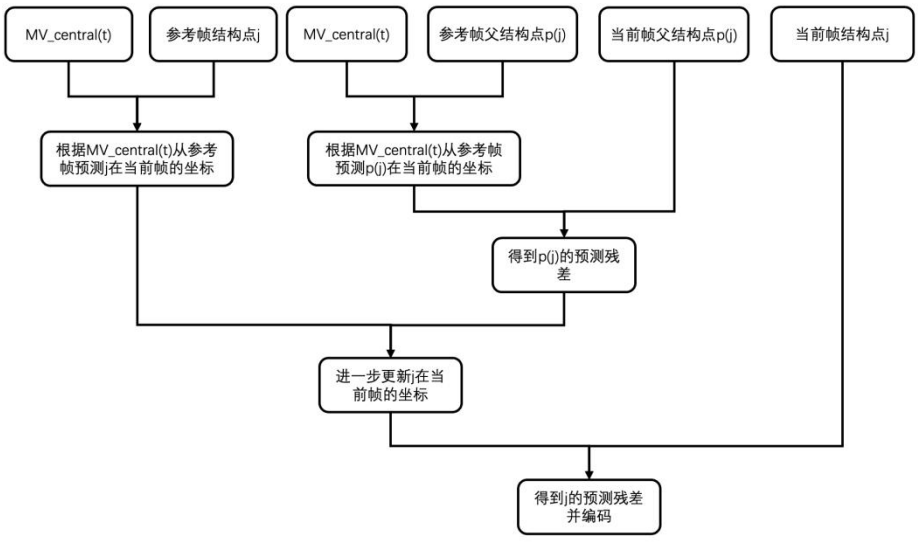


图 8 基于运动矢量的相对帧间差分模式的流程框图

7.2.2.4 帧间编码模式：基于线性预测的帧间差分模式

基于线性预测的帧间差分模式会根据第 t-2 帧（前序参考帧）和第 t-1 帧（参考帧）计算得到当前帧结构点 j 的运动矢量预测值，再用熵编码方法编码其预测值和真实值的残差。

预测过程见式（6）：

$$(MV\_ref x_{ID}^{j}(t), MV\_ref y_{ID}^{j}(t)) =$$

$$\{(x_{ID}^{j}(t-1) - x_{ID}^{j}(t-2), y_{ID}^{j}(t-1) - y_{ID}^{j}(t-2)) | j \in S\}, \quad (6)$$

此处规定 S 为当前结构体的所有结构点。将其应用到第 t-1 帧即可得到第 t 帧中结构点 j 的预测值，见式（7）：

$$(pred\_lix_{ID}^{j}(t), pred\_liy_{ID}^{j}(t)) =$$

$$\{(x_{ID}^{j}(t-1) + MV\_ref x_{ID}^{j}(t), y_{ID}^{j}(t-1) + MV\_ref y_{ID}^{j}(t)) | j \in S\}. \quad (7)$$

预测值与第 t 帧的结构点坐标做差得到预测残差，见式（8）：

$$(resx_{ID}^{j}(t), resy_{ID}^{j}(t)) = \{(x_{ID}^{j}(t) - pred\_lix_{ID}^{j}(t), y_{ID}^{j}(t) - pred\_liy_{ID}^{j}(t)) | j \in S\}.$$

(8)

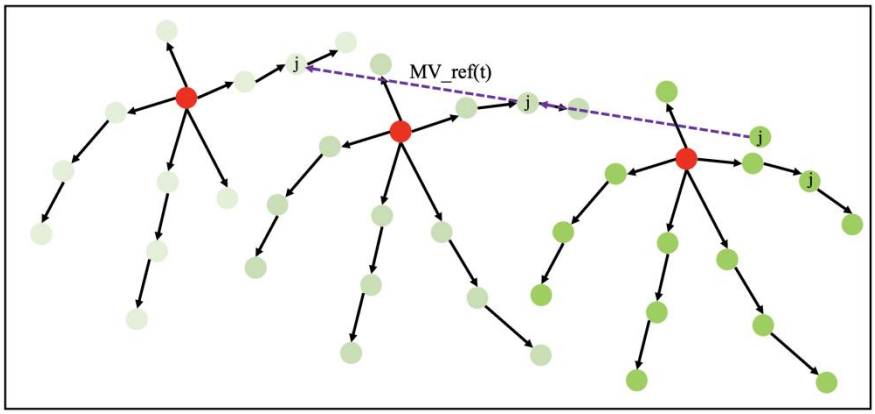


图 9 基于线性预测的帧间差分模式

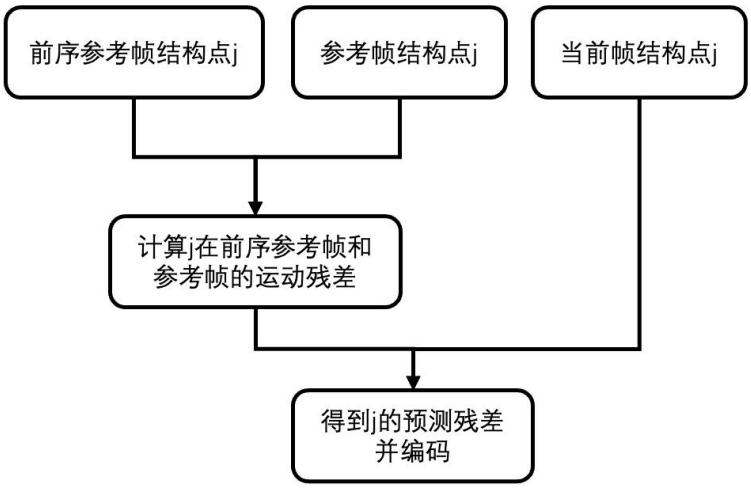


图 10 基于线性预测的帧间差分模式的流程框图

7.2.2.5 帧间编码模式：基于中值预测的帧间差分模式

基于中值的帧间差分模式首先分别通过 7.2.2.2 至 7.2.2.4 小节提出的三种模式计算出当前帧该结构体中每个结构点的坐标预测值，在每个维度上，取三者的中值作为预测值，再用熵编码方法编码各结构点的预测值和真实值的残差。见式（9）和（10）：

$$pred\_mex_{ID}^{j}(t) = Mid\{pred\_MVx_{ID}^{j}(t), pred\_MVRx_{ID}^{j}(t), pred\_lix_{ID}^{j}(t) | j \in S\}. \quad (9)$$

$$pred\_mey_{ID}^{j}(t) = Mid\{pred\_MVy_{ID}^{j}(t), pred\_MVRy_{ID}^{j}(t), pred\_liy_{ID}^{j}(t) | j \in S\}. \quad (10)$$

预测值与第 t 帧的结构点坐标做差得到预测残差，见式（11）：

$$(resx_{ID}^{j}(t), resy_{ID}^{j}(t)) =$$

$$\{(x_{ID}^{j}(t) - pred\_mex_{ID}^{j}(t), y_{ID}^{j}(t) - pred\_mey_{ID}^{j}(t)) | j \in S\}. \quad (11)$$

7.3 基于迭代预测的多个编码模式融合

基于多模式的编码方法的总体思想是以结构体的各个结构点为编码单位(即同一结构体的各个结构点可以采取不同的编码模式)，根据已编码信息(包括时间相关性和空间相关性)，在上述四种编码模式中，预测最佳模式对当前结构点进行编码。因为最佳模式通过预测而得，不使用标志位即可近似选取最优的编码模式，从而进一步提高了压缩率。

首先对于结构体中的每个结构点均定义二维模式选择矩阵$A_{ID}(t)[4][keypoint_num]$，其矩阵元素$a_{ID}(t)[i][j]$代表对于第 t+1 帧中编号为 ID 的结构体的 j 结构点使用第 i 个帧间编码模式的预估压缩效果。首先使用第 i 种间编码模式计算得到当前帧中结构点 j 的压缩占用比特，赋给$a_{ID}(t)[i][j]$：

1. 如果 j 结构点是中心结构点：$a_{ID}(t)[i][j]$由$a_{ID}(t)[i][j], a_{ID}(t-1)[i][j], \ldots, a_{ID}(t-t_0)[i][j]$加权计算得到。其中 t_0 最大取 4。如果由于先前某一帧 t_k 该结构体不存在导致$a_{ID}(t-t_k)[i][j]$不存在，则 $t_0=t_k-1$。

2. 如果 j 结构点不是中心结构点：在 1 的基础上，额外计入$a_{ID}(t)[i][p(j)]$的加权结

果。

$\{a_{ID}(t)[i][p(j)],\ a_{ID}(t)[i][j],\ a_{ID}(t-1)[i][j],\ a_{ID}(t-2)[i][j],\ a_{ID}(t-3)[i][j],$ $a_{ID}(t-4)[i][j]\}$对应的原始权重为$M=\{0.2667, 0.2667, 0.1778, 0.1333, 0.0889, 0.0667\}$。当某种情况中上述用于计算加权的 a 矩阵元素不足 6 个时, 需要根据原始权重重新计算其归一化权重。

举例如下：如 j 结构点不是中心结构点，且仅在参考帧中出现了该结构体，在前序参考帧中没有出现该结构体。则其计算方式见式（12）：

$$a_{ID}(t)[i][j]=\frac{m1}{s}*a_{ID}(t)[i][p(j)]+\frac{m2}{s}*a_{ID}(t)[i][j]+\frac{m3}{s}*a_{ID}(t-1)[i][j], \quad (12)$$

其中$m1=0.2667, m2=0.2667, m3=0.1778, s=m1+m2+m3$。

需要注意的是，对于某些帧，因为不满足该结构体必须出现在之前两帧的条件，线性预测法和中值预测法可能无法使用，此时对应行复制第 1 种编码模式（基于运动矢量的相对帧间差分模式）的行，即$a_{\mathrm{ID}}(t)[i][j]=k*a_{\mathrm{ID}}(t)[0][j]$,其中$k$为衰减系数；当 i=2 时，k=1.25；当 i=3 时，k=1.05。

对于一个视频中的结构点序列进行压缩编码时，首先应选取一定数目的关键帧，对于关键帧中的结构点均采用空间自差分模式编码保证独立可解，对于非关键帧中的结构点则采用跳过模式或基于多模式预测的压缩编码方法进行编码。

基于多模式的编码方法总结见图 11。

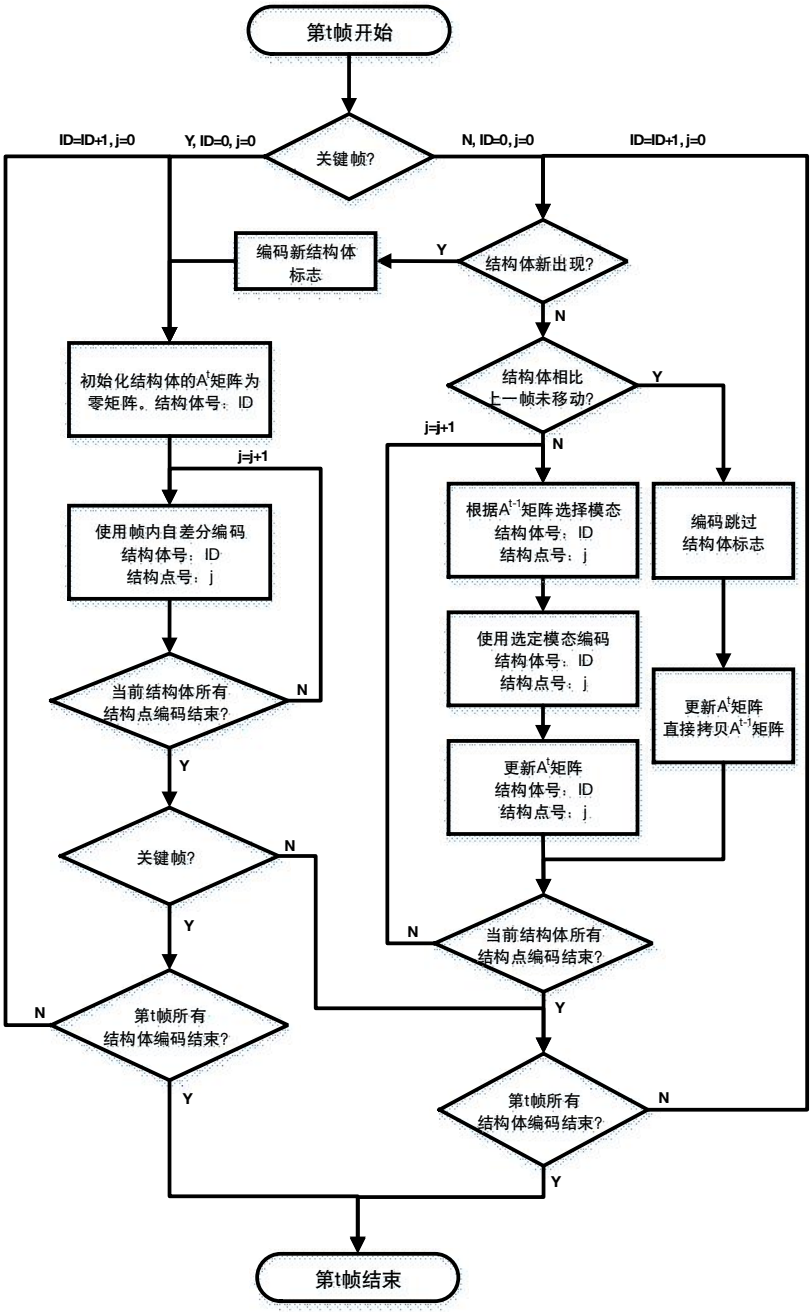


图 11 基于多模式的编码流程图

7.4 特殊处理

7.4.1 跳过模式

编码某个结构体时，倘若出现相对于参考帧中同一个 ID 的结构体，所有结构点的坐标均未改变的情况，则只在码流中写入一位 not_move 标识符，其值取 1，代表结构体位置没有改变，并跳过编码。

7.4.2 跳帧和特定编码模式选取

由于不同的应用场景对结构点序列的帧率要求不同, 因此在编码时规定 frame_skip 参数

控制编码器跳帧的个数。当该参数为 n 时, 表示每隔 n 帧抽出一帧进行编码, 如 frame_skip=2 时，只编码第 0,3,6,...帧。

对于不同的应用场景，可以灵活选取基于多模式的编码方法或者章节 7.2.3.2 至 7.2.3.5 节所述的某一个编码模式对非关键帧进行编码。规定 model_ID 参数设定编码方式(见表 16)，当该参数取 15 $(1111)_2$ 时，表示所有非关键帧均采用基于多模式的编码方法；该参数取 1$(0001)_2$,2 $(0010)_2$,4 $(0100)_2$,8 $(1000)_2$ 分别代表采用 7.2.2 节第 2、3、4、5 小节所述的单一编码模式对非关键帧进行编码。该参数取 0 到 15 间其他数值时，则代表若干种方法混合的方式进行编码：如果要采用第 7.2.2.2 和 7.2.2.3 两种方式编码，则 model_ID 是 1$(0001)_2$ 和 2 $(0010)_2$ 相加，为 3$(0011)_2$

7.5 输出码流格式

结构点序列的输出码流格式见图 12。

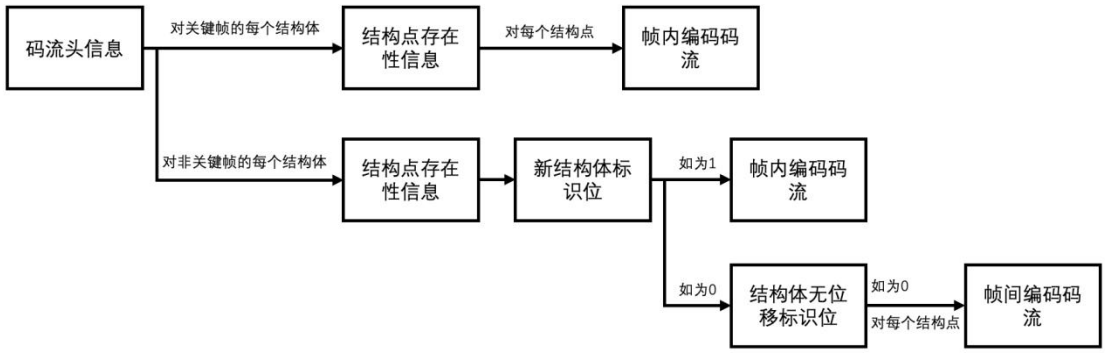


图 12 输出码流格式

7.6 含多种结构体的结构点序列编码

本技术支持对含有单种结构体的结构点序列进行编码。当一段视频具有多种结构体，并相应地生成了含有多种结构体的结构点序列时, 可以通过将不同种类的结构体剥离成含有单种结构体的结构点序列后，再各自编码的方式进行处理。

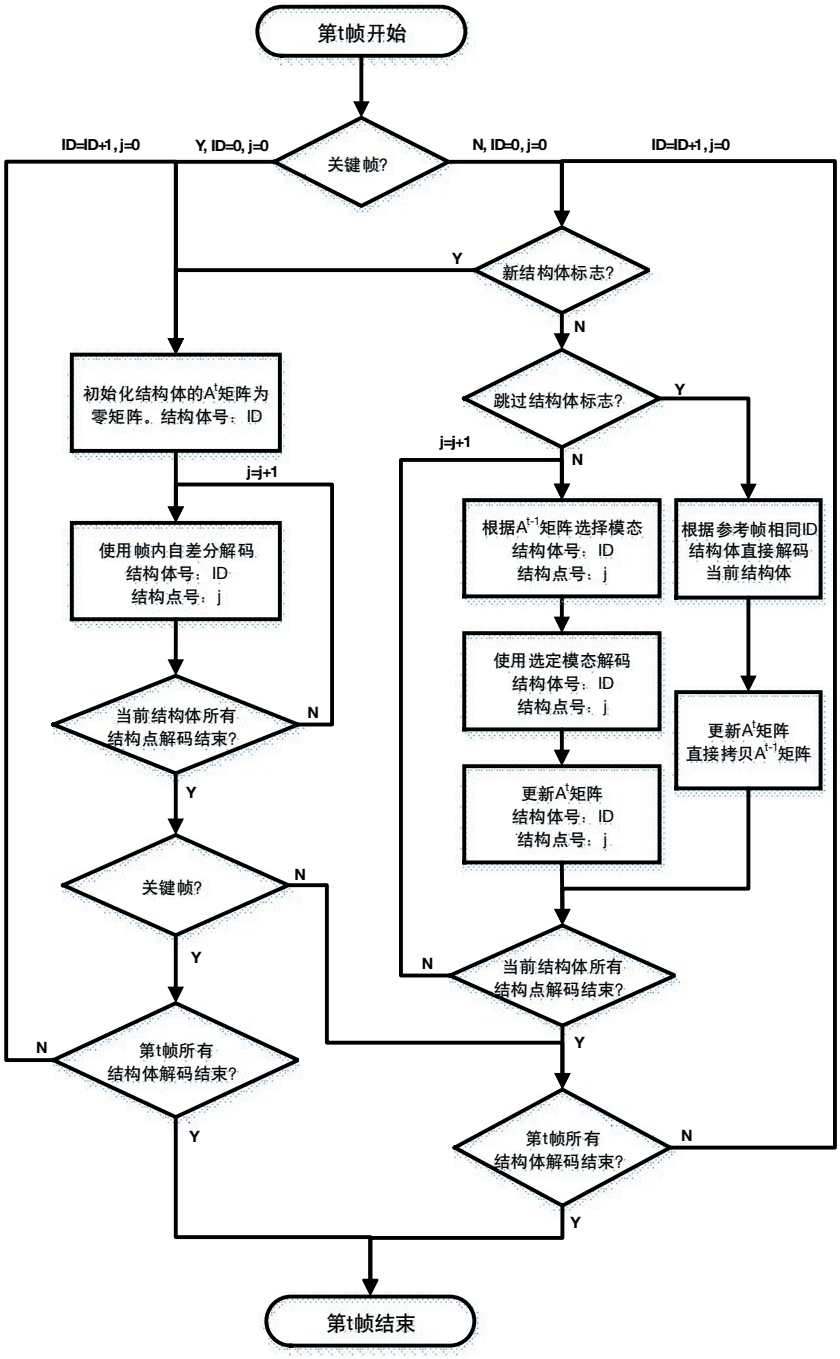
附 录 A
(资料性附录)
结构点获取

结构点序列作为一种数据结构，拥有很多应用场景。而在视频流中，结构点在空间上的获取往往还要伴随时域的跟踪匹配技术，以获取结构点的ID关系。

结构点序列数据可通过HIEVE(Human in events)[3]数据集获取。

附 录 B
（技术性附录）
结构点序列解码流程图

结构点序列解码流程见图 B.1。



图B.1 结构点序列解码流程图

附 录 C
（技术性附录）
技术参数说明

本技术涉及的参数见表C.1。

表 C.1 结构体序列的输入格式

参数名	参数含义	参数值
version_ID	结构点序列编码版本号	1
model_ID	模式启用参数	$(0001)_2$到$(1111)_2$的数，具体定义方式见表16与章节7.4.2。
frame_skip	跳帧个数	为0时，编码所有帧的数据，否则根据设定的值进行跳帧编码。
intra_frame_ratio	关键帧间隔	关键帧之间的距离。
key_point_num	结构点个数	单个结构体中的结构点个数。
key_point_dim	结构点维度	结构点的坐标维度数。
spatial_reference_order	空间依赖关系数组	标识单个结构体中相邻结构点之间的父子关系。
encode_order	编码顺序数组	从循环序号中获取要编码的结构点编号。
frame_num	视频帧号	当前执行的视频帧号。
total_frame_num	执行步数	需要处理的视频帧总数。
M	编码模式预测权重	{0.2667, 0.2667, 0.1778, 0.1333, 0.0889, 0.0667}

参 考 文 献

[1] Weiyao Lin, Xiaoyi He, Wenrui Dai, John See, Tushar Shinde, Hongkai Xiong, Lingyu Duan, "Key-Point Sequence Lossless Compression for Intelligent Video Analysis," IEEE MultiMedia, vol. 27, no. 3, pp. 12-22, 1 July-Sept. 2020.

[2] Weiyao Lin, Tushar Shankar Shinde, Wenrui Dai, Mingzhou Liu, Xiaoyi He, Anil Kumar Tiwari, Hongkai Xiong, "Adaptive Lossless Compression of Skeleton Sequences," Signal Processing: Image Communication, Vol. 80, 2020.

[3] Weiyao Lin, Huabin Liu, Shizhan Liu, Yuxi Li, Rui Qian, Tao Wang, Ning Xu, Hongkai Xiong, Guo-Jun Qi, Nicu Sebe, "Human in Events: A Large-Scale Benchmark for Human-Centric Video Analysis in Complex Events," arXiv preprint arXiv:2005.04490, 2020.

[4] https://en.wikipedia.org/wiki/Exponential-Golomb_coding

[5] https://thestocks.im

[6] http://humanineventsorg

[7] Wenyan Wu, Chen Qian, Shuo Yang, Quan Wang, Yici Cai, Qiang Zhou, "Look at Boundary: A Boundary-Aware Face Alignment Algorithm," Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 2129-2138, 2020.

[8] https://github.com/MVIG-SJTU/AlphaPose