多媒体基础大作业

信息 005 王靳朝 2206113602

1. 给出 MPEG-1 帧中 I 帧编码的流程，并给出其算法流程

MPEG-1 中 I 帧的编码的基本思路是及逆行帧内压缩和帧间压缩，另外可以通过时间相关性的统计分析，在相邻间隔的 1~2 帧中，像素点变化、亮度变化、色度插值变化均较小。主要包含以下步骤：

1.空间域预处理：

获取图像的原始数据，对图像进行必要的预处理。将用 RGB 颜色空间表示的图像，转化为 YCbCr 空间表示的图像，后将每个图象平分为 8*8 的像素块，此时可以进行色彩空间转换和亮度调整。

2.DCT 变换：

对每个块进行 DCT 变换，将图像从空间域转换到频域。得到图像的高频数据和低频数据。

3.量化：

由于人眼对高频区域并不敏感，因此对 DCT 变换后的系数进行量化，将高频的数据进行压缩以减少数据的精度。量化表中的值确定了每个频域系数的量化级别。

4.熵编码：

使用熵编码技术（如霍夫曼编码）对量化后的数据进行编码，以减少数据的冗余。生成熵编码表。

5.位流打包（Bitstream Packing）：

将经过熵编码的数据组织成位流。将各个数据单元进行打包，添加同步字和标识信息。

伪代码算法流程为：

```
# 空间域预处理
original_image = read_original_image()
processed_image = spatial_preprocessing(original_image)
# DCT 变换
blocks = divide_into_blocks(processed_image, block_size=8)
dct_coefficients = apply_dct(blocks)
# 量化
quantized_coefficients = quantize(dct_coefficients, quantization_table)
# 熵编码
encoded_data = entropy_encode(quantized_coefficients, huffman_table)
# 位流打包
bitstream = pack_bitstream(encoded_data, sync_code, frame_type)
# 输出位流
output(bitstream)
```

2. 给出运动估计和运动补偿的原理，结合 MPEG-1 中的 P 帧编码，描述其实现流程

运动估计是从视频序列中抽取运动信息的技术。运动估计的基本思想是将图像的序列的每一帧分成许多互不重叠的宏块，并认为宏块内所有像素的位移量都相同，然后对于每个宏块，再参考帧的某一给定的搜索范围内，根据一定的匹配准则找出与当前块最相似的块，即匹配块，匹配块与当前块的相对位移即为运动矢量。运动矢量和残差数据可以用于视频压缩和回复。运动估计的方法可以分为基于块的和基于参数的。及愉快的简化了运动矢量的估算，但编码码率略大于基于参数的方法。基于参数的方法需要对图像进行区域划分，然后用物体的运动模型来描述运动矢量，获得更加精准的图下个表示。

运动补偿利用先前的局部图像来预测、补偿当前的局部图像，从而减少视频序列的空余冗余。具体方法是描述前面一阵的每个小块怎样移动到当前帧的某个位置，也可以用来进行去交织的操作。

MPEG-1 中的 P 帧编码主要过程如下：

首先从前一关键帧或者 P 帧中选择一个参考帧。其次将参考帧与当前帧进行愚弄的那个估计，以确定当前帧中每个宏块的最佳匹配块。然后对于每个宏块计算残差，及当前帧的像素值与参考帧的像素值之间的差异。之后对残差进行变换和量化，并对量化后的数据进行熵编码。伪代码表示为：

```
# 运动估计
for each block in current_frame:
    motion_vector = motion_estimation(block, previous_frame)
# 运动补偿
predicted_frame = motion_compensation(previous_frame, motion_vector)
# 误差计算
residual_frame = current_frame - predicted_frame
# DCT 变换、量化和熵编码
dct_coefficients = apply_dct(residual_frame)
quantized_coefficients = quantize(dct_coefficients, quantization_table)
encoded_data = entropy_encode(quantized_coefficients, huffman_table)
# 位流打包
bitstream = pack_bitstream(encoded_data, motion_vector, frame_type)
# 输出位流
output(bitstream)
```

3. 给出 MPEG-4 中基于对象的编码流程

MPEG-4 的编码思想是在编码是将统一幅景物分成若干在时间和空间上相互联系的视频音频对象没分别编码后在经过复用传输到接收端，然后在对不同的对

象分别进行解码，从而组合成所需要的视频和音频。
MPEG-4 的编码流程如下：
VO 的形成：先要从原始视频流中分割出 VO，之后由编码控制机制为不同的 VO 以及各个 VO 的三类信息分配码率。
形状编码：对于任意形状的 VOP，需要进行形状编码。
运动纹理编码：对于前景对象和后景对象采用不同的编码策略，对于人们所关心的前景对象，则尽可能地保持对象的细节及平滑，而对不大关心的后景对象采用大压缩比的编码策略。
复合阶段：将各个 VO 的码流复合成一个位流。

4. 给出 H.264 I 帧编码的大致流程

H.264 也被称为高级视频编码，简称 AVC，该标准引入了一系列能够大规模提高压缩性能的技术，同时在高码率端和低码率端远超以前诸多标准。在其内部定义的三种帧，分别是 I 帧、B 帧和 P 帧。

I 帧它是一个全帧压缩编码帧，将全帧图像信息进行压缩编码及传输，解码时仅用 I 帧的数据就可重构完整图像。它的特点还有：
I 帧描述了图像背景和运动主体的详情；
I 帧不需要参考其他画面而生成；
I 帧是 P 帧和 B 帧的参考帧，其质量直接影响到同组中以后各帧的质量；
一般地，I 帧是图像组 GOP 的基础帧（第一帧），在一组中只有一个 I 帧；
I 帧所占数据的信息量比较大。

Ⅰ帧的编码流程为：首先对当前帧进行帧内预测，决定采所采用的帧内预测模式。其次当前像素值减去预测值，已得到残差。并对残差进行量化和编码。最后对重构图像进行滤波，得到的图像作为其他帧的参考帧。最后进行熵编码以及位流打包，伪代码表示为：

```
# 帧内预测
for each 16x16 macroblock in processed_image:
    intra_prediction(macroblock)
# 变换与量化
for each 16x16 macroblock:
    residual_block = macroblock - predicted_macroblock
    transformed_block = integer_dct(residual_block)
    quantized_block = quantize(transformed_block, quantization_matrix)
# 熵编码
encoded_data = entropy_encode(quantized_block, entropy_coding_method)
# 位流打包
bitstream = pack_bitstream(encoded_data, sync_code, frame_type)
```

```
# 输出位流
output(bitstream)
```

5. 对比分析 MPEG-1/4,H.264 编码的特点

从发布时间来看，MPEG-1 发布于 1993 年，是最早的视频压缩标准之一，MPEG-4 发布于 1998 年，为多媒体技术引入了更高级别的压缩，H.264 发布于 2003 年，更大提高了压缩比例。

MPEG-1/4 采用了混合编码，结合了运动补偿策略以及变换编码策略进行预测残差估计。MPEG-1 中引入的运动补偿与 B 帧，MPEG-4 引入了基于对象的编码方式。

H.264 最大的优势是具有很高的数据压缩比率，在同等图像质量的条件下，H.264 的压缩性能比 MPEG-2 高 50%，比 MPEG-4 高 30%。因为他将重心聚焦于降低 I 帧的码率，因此 H.264 能够在低码率的情况下提供高质量的视频图像，能够在较低的带宽上提供高质量的图像传输是 H.264 的亮点。