[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04N 7/24 (2006.01)
G06T 9/00 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200610089252.1

[43] 公开日 2008 年 2 月 13 日

[11] 公开号 CN 101123723A

[22] 申请日 2006.8.11
[21] 申请号 200610089252.1
[71] 申请人 北京大学
地址 100871 北京市海淀区颐和园路 5 号
[72] 发明人 周秉锋 韩 博

[74] 专利代理机构 北京君尚知识产权代理事务所
代理人 贾晓玲

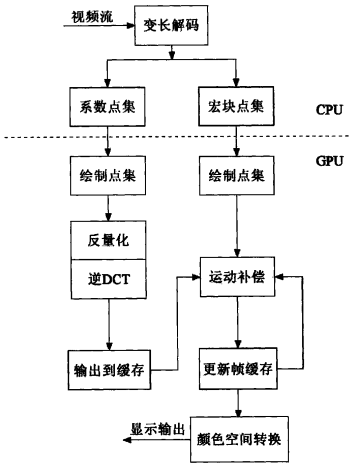
权利要求书 2 页 说明书 10 页 附图 9 页

[54] 发明名称

基于图形处理器的数字视频解码方法

[57] 摘要

本发明提出了一种基于 GPU 的压缩视频解码方法。该方法利用点图元而不是矩形来表示视频块，将除变长解码外的所有解码环节成功映射到 GPU 上，利用 CPU 将视频数据组织成点集并通过绘制点集的方式完成解码过程。本发明结合了 CPU 和 GPU 各自的优势，使两者并行工作加速视频解码过程，同时具有硬件解码的高性能和软件解码的灵活性，可以处理多样的视频压缩格式和标准，能用来替代配备 GPU 的个人计算机，游戏主机，手持移动设备等上的专用解码硬件，提高硬件资源的利用率，降低成本。



1、一种基于图形处理器的数字视频解码方法，其步骤包括:

1）CPU 变长解码得到宏块和系数块，并用图形绘制中的基本图元“点”表示，生成视频块对应的点集;

2）CPU 将视频块对应的点集以批处理方式分批送入 GPU;

3）绘制宏块点集和 DCT 系数点集，GPU 执行相应的顶点和像素处理程序完成解码过程。

2、如权利要求 1 所述的一种基于图形处理器的数字视频解码方法，其特征在于：步骤 1）将宏块表示为点图元的方法为，将宏块的解码所需信息存储在点图元的属性中，上述解码所需信息为宏块的位置，类型，运动向量，上述点图元的属性为位置，法线，纹理坐标等向量属性。

3、如权利要求 1 或 2 所述的一种基于图形处理器的数字视频解码方法，其特征在于：步骤 1）将系数块表示为点图元的方法为，将系数块中的 DCT 系数存放在点图元的属性中， DCT 系数利用 CPU 预处理生成规整的系数存放结构。

4、如权利要求 1 所述的基于图形处理器的数字视频解码方法，其特征在于：步骤 1）CPU 在生成宏块和系数块对应的点图元时，根据宏块和系数块的不同类型和操作过程，预先划分点图元到不同的点集。

5、如权利要求 1 所述的一种基于图形处理器的数字视频解码方法，其特征在于：步骤 3）解码过程中的反量化操作在 GPU 顶点处理器中完成，量化矩阵通过 Uniform 参数装入常量寄存器，结合点属性中存储的系数索引和量化参数，利用向量乘法完成反量化。

6、如权利要求 1 所述的一种基于图形处理器的数字视频解码方法，其特征在于：步骤 3）解码过程中的逆 DCT 过程是通过 DCT 系数和对应基图像的线性组合来实现的，基图像生成纹理，并以纹理的形式存放在 GPU 中。

7、如权利要求 6 所述的一种基于图形处理器的数字视频解码方法，其特征在于：从基图像生成纹理的过程同权利要求 3 所述的系数点的生成过程相对应，每个四元组系数对应的基图像分别存储在同一个纹理块的 RGBA 通道中。

8、如权利要求 1 所述的一种基于图形处理器的数字视频解码方法，其特征在于：步骤 3）解码过程中的逆 DCT 操作过程依次包含以下步骤:

1）顶点处理程序计算基图像的纹理坐标;

2）按照设定的点图元大小光栅化为像素块;

3）像素处理程序采样基图像，并与继承的系数属性做点乘运算;

4）激活 GPU 混合功能并设置为加法操作，累加不同点图元中的计算结果，输出到预

测残差图像纹理。

9、如权利要求 1 所述的一种基于图形处理器的数字视频解码方法，其特征在于：步骤 3）解码过程中的运动补偿过程依次包含以下步骤：

1）顶点处理程序根据预测精度处理运动向量，设置对应的小数部分；

2）光栅化宏块点图元为像素块；

3）像素处理程序根据运动向量计算参考块在参考帧中的纹理坐标，采样参考帧和权利要求 8 步骤 4）输出的预测残差图像纹理，累加结果并做饱和化操作。

10、如权利要求 9 所述的一种基于图形处理器的数字视频解码方法，其特征在于：所述采样参考帧过程是利用了 GPU 纹理单元的双线性过滤功能来完成子像素精度运动补偿所需要的插值运算。

基于图形处理器的数字视频解码方法

技术领域

本发明属于计算机数字视频压缩领域，具体涉及一种利用图形处理器（Graphics Processing Unit — GPU）来完成视频解码的方法。

背景技术

数字视频已经广泛应用于人们的日常生活中，涉及数字电视，个人计算机，手持移动设备以及娱乐，教育等各个领域。对于广大用户来讲，最基本的要求就是对视频内容的高质量实时播放(解码)。但是视频压缩标准为了取得高的压缩率和好的图像质量，需要采用高计算复杂度的视频压缩技术，这直接导致其解码过程需要消耗大量的计算资源。

常见的视频压缩标准大多以 16×16 大小的宏块为基本的处理单元，参考图 1，为完成解码过程，需对每个宏块依次完成以下处理环节：变长解码，反量化，逆离散余弦变换（Inverse Discrete Cosine Transform — IDCT），运动补偿和颜色空间转换。变长解码完成视频比特流的解析，恢复视频的熵编码信息，例如每个宏块的参数，系数和运动向量等，该过程为严格串行的位操作。随后的反量化和 IDCT 作用在构成宏块的每个系数块上，处理稀疏的 DCT 系数，用来恢复原始的像素块，该变换过程计算复杂度高。运动补偿是减少视频序列中时间冗余的有效方法，以宏块为基本的单位。该过程在编码阶段的基本原理是在参考帧中搜索一个与当前图像中宏块最相似的图像块，即预测块，搜索结果用运动向量来表示，计算当前宏块和预测块之间的差值，然后将此差值和运动向量加以编码。运动补偿就是通过差值和运动向量恢复出编码图像的过程。由于好的预测往往带来更好的编码效率，因此常见的视频编码系统都采用了双向预测(B 帧)和子象素精度运动向量等技术来提高运动估计的准确度。虽然提高了预测精度和压缩率，但是却进一步增加了运算的复杂度。最后的颜色空间转换过程针对图像中的每个像素进行颜色向量（RGB）与变换矩形的乘法运算，为典型的计算密集型过程。由此可见，视频的解码过程是由多个耗时的处理环节所共同构成的复杂系统。

面对高质量和高分辨率视频以及新一代压缩标准(例如 H.264)所引入的复杂压

缩技术，目前计算机系统中单纯利用 CPU 的软件解码器甚至无法满足实时解码视频的需求。因此，需要其他的子系统来分担部分解码任务从而缓解 CPU 的压力。近十年来专用的视频解码硬件被引入到计算机系统中，以独立板卡或集成于图形硬件内的形式出现。微软的 DirectX Video Acceleration（DXVA）规范的推广使后者成为目前的主流。但是这种专用解码硬件往往只能针对某种特定的视频压缩标准（大部分为 MPEG-2）因而具备非常有限的扩展性和可编程性，缺少足够的灵活性来应对目前多样的视频压缩格式。虽然目前图形卡上已经开始集成可编程的视频处理硬件，例如 Nvidia 的 PureVideo 和 ATI 的 Avivo 等技术，但是它们需要额外的硬件开销和更高的成本，而且目前缺少有效的高级语言和应用程序接口来方便的控制这些底层硬件资源。

另一方面，随着三维图形应用的发展和推广，图形硬件已经演化成为一种兼具高性能和灵活性的图形处理器，也就是 GPU，目前主要的可编程部分包括顶点处理器（Vertex Processor）和像素处理器（Fragment Processor）。这两部分处理单元结合光栅和合成器件构成 GPU 的流水线处理结构。图形处理器大规模并行所带来的高性能，成熟的高级着色语言所带来的可编程性以及高精度数据类型的支持（32 位浮点数）使得 GPU 成为计算机系统中除 CPU 外的一个极具吸引力的协处理器，并且可被用于解决图形领域之外的很多通用计算问题（GPGPU），例如数值计算，信号处理，流体模拟等。从体系结构的角度来考虑，GPU 是以向量操作为基础的高度并行的流处理器，该结构同一些成功的专用多媒体和视频处理器具有很大的相似形。这些都为在 GPU 上实现高效的视频解码提供了强有力的支持。

但是 GPU 从设计到发展都是为了加速图形计算，处理的数据都是相对规整的顶点和像素，所以并不能直接用于相对复杂和多分支的视频解码过程。除了最后的颜色空间转换环节外，GPGPU 领域常用的纹理方法并不适用于该解码过程。主要原因在于目前的大多数视频压缩标准都基于宏块/系数块这种组织结构；每个宏块或系数块都有自己特有的参数和属性，彼此有所不同，不便于用规整的单一纹理来表示。一些以纹理表示法为基础的前人工作，例如利用 GPU 的 DCT/IDCT 变换，性能同 CPU 相比并没有优势，而且还存在相当可观的数据传输开销。文献“Accelerate video decoding with generic GPU”(Shen G. 等，IEEE Transaction on Circuits and Systems for Video Technology, May 2005)利用小矩形来表示宏块，从而完成解码中的

运动补偿过程，虽然有效但仍然存在数据冗余等问题。这些方法没有充分利用 GPU 的计算资源导致性能偏低，并不适用于实用的视频解码系统。

发明内容

本发明的目的是解决目前软硬件解码方案在性能或灵活性上的不足，提出了一种基于 GPU 的压缩视频解码方法。该方法兼具硬件的高性能和软件的灵活性，适用于多种视频压缩标准，能用来替代配备 GPU 的个人计算机，游戏主机，手持移动设备等上的专用解码硬件，提高硬件资源的利用率，降低成本。

本发明的上述目的是通过如下的技术方案予以实现的:

一种基于图形处理器的数字视频解码方法，其步骤包括:

1）CPU 变长解码得到宏块和系数块，并用图形绘制中的基本图元“点”表示，分别生成宏块对应的宏块点集和系数块对应的 DCT 系数点集;

2）CPU 将宏块点集和 DCT 系数点集以批处理方式分批送入 GPU;

3）绘制宏块点集和 DCT 系数点集，GPU 执行相应的顶点和像素处理程序完成视频解码过程。

本发明将构成视频的基本单位－“宏块和系数块”用图形绘制中的基本图元－“点”来的表示，从而将传统的视频解码过程映射为点集的绘制过程，继而充分发挥出 GPU 流水线处理和大规模并行处理的优势，取得更高的解码性能。在绘制点集的过程,通过顶点和像素程序来控制 GPU 上可编程的顶点处理器和像素处理器来完成解码过程中的主要环节：反量化，IDCT，运动补偿和颜色空间转换，并进一步利用 GPU 上的合成单元(Blending)和纹理过滤单元来分担部分计算任务。该技术方案具体包括以下几个方面:

1）利用点图元而不是矩形来表示视频块信息。工作原理在于利用点的属性（四维向量）如位置，法线和纹理坐标等存储视频中的宏块和系数块的类型，位置，参数，系数等信息。其中宏块和系数块分别对应两类不同的点集：宏块点集和 DCT 系数点集,分别用于运动补偿和 IDCT。其中 DCT 系数点集的生成过程利用了 Zigzag 扫描来减少点集中点的个数。考虑到 GPU 低效的分支处理能力和不同类型的宏块或系数块对应的不同操作过程，在生成 DCT 系数点集和宏块点集的过程中，利用 CPU 对两类点集做了进一步的细分，对应同类操作的块被分到一类子集中，例如宏

块中所有的非预测宏块（Intra）聚为一类，所有的前向预测宏块（forward）聚为另一类。

2）解码过程中的反量化和 IDCT 过程通过一次绘制 1）中创建的 DCT 系数点集来完成。其中反量化完全由 GPU 的顶点处理器完成，而 IDCT 主要在像素处理器中完成，两者构成流水线结构提高了执行效率。反量化中的量化参数和 DCT 系数通过点图元的属性送入顶点处理器而量化矩形通过 uniform 参数预先设置放入顶点处理器的常量寄存器。IDCT 过程通过在像素处理单元中线性组合 DCT 系数和对应的基图像来完成，基图像经预处理后，以纹理的形式存放在 GPU 的显存中。对于分布于多个点中的属于同一个系数块的 DCT 系数，利用 GPU 中的混合器件（blending）累加多个点图元中的结果到 IDCT 输出缓存中（残差图像纹理）。

3）运动补偿过程通过绘制 1)中创建的宏块点集来完成，在像素处理单元中采样参考图像纹理和步骤 2）中输出的 IDCT 输出纹理，累加采样结果并做饱和化运算，完成运动补偿过程。对于子像素精度的运动补偿，利用 GPU 纹理单元的双线性过滤硬件来实现子像素的插值运算。

本发明的优点可以总结为以下几个方面：

1）该方法结合了 CPU 和 GPU 各自的优势，使两者并行工作加速视频解码过程，同时具有硬件解码的高性能和软件解码的灵活性，可以处理多样的视频压缩格式和标准。

2）对比专用的视频硬件，该解决方案基于上层的图形 API (如 OpenGL)和高级着色语言（如 CG 和 GLSL）可以做到平台无关和系统无关，独立于底层的具体硬件实现，适用于各种配有 GPU 的系统中，如个人计算机，游戏主机，手机和 PDA 等。GPU 的演化速度快，性能增长幅度远远超过摩尔定律，不断增加新的功能和特性带来更灵活的可编程性，从长远看比 CPU 软解码和专用硬件更具潜力。

3）该方法用点来表示宏块和系数块，实现简单，控制灵活。对比纹理表示法，基于点的方法仅传输非零系数；对比矩形表示法，消除了矩形中四个顶点的大量冗余数据，从而减少了传输开销，降低了带宽需求。同时，点方法控制灵活，方便剔除未编码块（non-coded block），而且系数块所对应的 DCT 系数点图元生成过程自动剔除了零系数，减少了不必要的计算。基于点的表示方式，也方便利用 GPU 处理流水线中的顶点处理器和光栅化硬件，充分挖掘 GPU 的计算资源。另一方面，

利用 CPU 划分为不同点集的方法，消除了 GPU 分支处理的瓶颈，提高了性能。

附图说明

下面是对本发明附图的简要说明:

图 1 是典型的视频解码过程的主要环节示意图。

图 2 是本发明对应的硬件系统结构图。

图 3 是数字视频的宏块/块结构的示意图。

图 4 是本发明通过绘制点集来利用 GPU 进行视频解码的整体流程图。

图 5 是本发明中从视频的系数块产生 DCT 系数点图元的示意图。

图 6 是 DCT 基图像形成纹理的示意图。

图 7a 是 IDCT 过程的输出缓存的结构示意图。

图 7b 是运动补偿过程的帧缓存的结构示意图。

图 8a 是子像素精度运动补偿插值过程的示意图。

图 8b 是纹理过滤单元双线性插值的示意图。

图 9 是绘制 DCT 系数点集完成反量化和 IDCT 的过程示意图。

图 10 是绘制宏块点集完成运动补偿的过程示意图。

具体实施方式

下面参照本发明的附图，更详细地描述本发明的最佳实施例。

图 2 说明了本发明所对应的硬件系统的结构图。本发明需要 CPU 和 GPU 合作完成整个解码过程，两者可以并行执行，进一步提高效率。CPU 和 GPU 间通过系统总线相连接，例如 PCIE 或 AGP。总线带宽是个有限资源，数据传输开销是影响整体性能的重要因素。本发明对比现有方法的一个重要优势就在于避免了无用或冗余数据，明显降低了数据传输量。CPU 将视频中宏块和系数块的解码所需信息封装进绘制用的点集中，以顶点数组或其他形式暂存于系统内存中，然后通过系统总线传入 GPU。GPU 是本发明中解码任务的主要执行单元，完成主要的解码任务，要求具有可编程性的顶点和像素处理器以及一定容量的显存用于存放计算数据和中间结果。

本发明提出了一种用点图元表示视频中宏块和系数块，通过图形硬件 GPU 绘制

对应宏块和系数块所对应的点集来实现视频解码的方法。本发明的处理流程如图 4 所示。下面结合附图详细的说明本发明实现视频解码的具体步骤:

1）CPU 变长解码生成视频中宏块和系数块对应的点集。

首先 CPU 完成变长解码得到视频中宏块和系数块的信息，然后将视频信息封装进点图元的属性中，并根据宏块和系数块的不同类型或处理过程分类到不同的点集中，处理完所有的视频块后，对应的点集以批处理的方式（如顶点数组）分批送入 GPU，提高 GPU 并行和流水线执行的效率。

点集被划分为两个大类：DCT 系数点集和宏块点集。这一划分的主要依据是目前压缩视频基于块的构成结构，如图 3 所示，其中的宏块是运动补偿的基本单位，而构成宏块的系数块是反量化和 IDCT 的基本单位。两大类点集都可以依据块的类型和特性进一步划分为新的子集。例如 DCT 系数点集可以根据 DCT 编码方式的不同进一步划分为场 DCT 编码点集和帧 DCT 编码点集；宏块点集可以根据宏块类型细分为非预测宏块（Intra）集，单向预测宏块集和双向预测宏块集等。这些不同类型的视频块往往对应不同的解码处理过程，预先利用 CPU 分类到子集并分别送入 GPU 执行，可以避免 GPU 上耗时的分支操作，提高整体的解码效率。

宏块和系数块中的信息封装进点图元的过程有所不同，但是基本思想都是利用点图元的多个向量属性如位置，法线，颜色，纹理坐标等来存储视频块中的类型，参数，系数等有用信息。

宏块中包含的主要信息为宏块的位置，类型（intra，inter）和运动向量，可以直接放入点图元的向量属性中，从而将宏块转化为点图元。

系数块的主要信息为 DCT 系数。得益于 DCT 的能量汇聚特性和量化过程，一个 8×8 系数块中的 64 个 DCT 系数仅有少量非零值分布在低频部分。虽然能够将少量系数直接放入点图元的属性中，但是不同系数块内的系数分布无规律，不利于形成适合 GPU 处理的规整点集，所以需要对每个系数块中的系数重新组织生成规整的结构。我们利用 DCT 系数的 Zigzag 存储形式来生成对应的系数点图元，如图 5 所示。Zigzag 扫描将二维转化为一维形式将非零系数尽可能的集中起来。以 Zigzag 的一维系数数组为基础，每四个系数为一个组对应点图元中的一个四维属性。为保证点的规整性，每一个或特定多个四维属性放入一个点中，同时装入该组系数在此一维数组中的索引（系数索引），连同系数块的位置，类型，量化参数信息，形成

一个 DCT 系数点图元。该方法的直接结果就是对每个视频块可能产生多个点图元。我们随后利用 IDCT 过程将分散到不同点中的结果累加起来。

上述点图元的生成方法作用于每帧图像中的所有的宏块和系数块，产生的点集以顶点数组(Vertex Array)的形式存储在系统内存中，然后利用图形 API 绘制点集，数据以批处理的方式送入 GPU 完成随后的解码过程。

2）初始化图形 API 绘制环境。

a) 调用 API 函数设置点图元光栅化后的大小（例如 OpenGL 中的 *glPointSize*）。绘制 DCT 系数点集时大小设置为 8 并激活点精灵方式的纹理生成(Point Sprite ARB Extension)，绘制宏块点集时设置为 16。对于可变大小的块结构，可以将块的大小存放在点的属性中，改变 GPU 的顶点处理器中的 PSIZE 寄存器来实现不同的光栅化大小。

b) 在 GPU 上分配好离屏缓存(off-screen buffer)空间，存放中间输出结果。我们分配了一个 IDCT 输出缓存和三个帧缓存。为了保证 IDCT 运算的精度，IDCT 的输出缓存为单通道的 16 位浮点数格式（fp16），亮度和色度分量如图 7a 所示。由于运动补偿过程需要保留参考帧，三个帧缓存分别用于保存前向参考帧，后向参考帧和当前帧，帧缓存的结构为 8 位 RGB 三通道的 Unsigned Byte 类型，结构如图 7b 所示，亮度分量保存在 R 通道中，两个色度分量经过插值后分别保存在 G 和 B 通道中。利用 GPU 的“渲染到纹理”功能，如 OpenGL 的 render to texture extension 或 FBO，这些缓存可以在渲染完成后直接作为纹理供采样和访问。对于 IDCT 输出纹理设置纹理过滤模式为“Nearest”；对于用于运动预测的帧缓存设置纹理过滤为“Bilinear”以便纹理采样时自动激活纹理过滤功能用于子像素精度运动补偿; 同时设置纹理寻址模式为“Clamp”用于“非受限的运动向量”所需的对图象边缘的像素填充（padding）。

c) 处理 DCT 基图像，合成供 GPU 采样用的基图像纹理。IDCT 变换可以看作是 DCT 系数和其对应基图像的线性组合，如下列公式所示:

$$x = \sum_{u=0}^{N}\sum_{v=0}^{N} X(u,v)[T(u)^T T(v)]$$

其中的 x 表示 IDCT 后的像素块，$X(u,v)$ 表示 DCT 系数块中 (u,v) 处的系数，T 表示 DCT 变换矩阵，$T(u)$ 是该矩阵的第 u 行，系数 (u,v) 所应对的基图像通过列向

量$T(u)^T$和行向量$T(v)$的外积生成。上述公式的计算过程为标量和矩阵的乘法运算和矩阵的线性组合运算。该过程的主要优势在于每个系数的计算相对独立，而且可以直接剔除零值系数减少计算量。

基图像纹理生成过程如图 7 所示。按照 Zigzag 扫描的次序，每四个系数对应的基图像存放在一个 8×8 纹理块的 RGBA 通道中，为了保证 IDCT 运算的精度，每个颜色通道的数据精度为 16 位。这样最终可以得到一个 32×32 大小，16 位浮点数精度 RGBA 格式的纹理。

d) 装入对应于绘制 DCT 系数点集的顶点处理程序(Vertex Program)和像素处理程序(Fragment Program)。通过 Uniform 参数将量化矩阵装入顶点处理程序，用于反量化。

3）完成准备工作 2）后，开始绘制步骤 1）生成的 DCT 系数点集，在绘制过程中 GPU 完成反量化和 IDCT 过程，如图 9 所示。

a) 顶点处理器实现反量化。反量化过程本质上是量化步长和系数的乘法运算。操作过程如下:

$$X_{iq}(u,v) = qp \times QM(u,v) \times X_q(u,v)$$

其中的$X_q(u,v)$和$X_{iq}(u,v)$分别表示反量化前后的 DCT 系数，qp所表示的量化参数已经通过步骤 1）中系数点图元的生成过程放入点的属性中，$QM(u,v)$表示量化矩阵的对应项，整个量化矩阵已经在步骤 2) d)中装入常量寄存器，对应的项（entry）可以通过步骤 1）中引入的系数索引获取。由于系数以向量形式存放，顶点处理程序中一次向量乘法可以完成四个系数的反量化过程。

顶点处理程序还可以根据系数索引计算出系数对应的基图像的纹理坐标，并传递到随后的光栅化阶段。

b) 光删化阶段根据步骤 2) a)中设定的点的大小和顶点处理器输出的位置，将点图元转化为相应位置上指定大小的像素块。同时像素块所覆盖的每个像素都继承了点图元在顶点处理阶段的输出属性。对于系数点集，激活步骤 2) a)点精灵纹理生成后，每个像素会生成对应的块内纹理坐标，范围(0,0) - (1,1)。

c) 像素处理器结合 a)中输出的基图像纹理坐标和 b)中形成的块内纹理坐标，可以精确采样每个像素点对应的基图像纹理值。考虑步骤 2）c) 中的 IDCT 计算公式。

此时已经将标量与矩阵间的乘法运算转换为了像素间的直接运算形式。由于系数和基图像纹理值都以 RGBA 四维向量的形式存在，像素处理程序中的一次向量点乘操作就可以完成四个系数的乘法和累加，然后输出结果到缓存中。

d) 激活 GPU 硬件的混合功能（Blending），并设置为 Add 运算。由于步骤 1)中每个系数块可能生成的多个系数点图元，通过该步骤每个点图元输出的运算结果可以最终在输出缓存中累加，从而完成步骤 2）c)中 IDCT 计算公式中对所有系数的线性累加。

到此，DCT 系数点集的绘制完成，视频中系数块反量化和 IDCT 后的结果保存在 IDCT 的输出缓存中，作为残差图像纹理用于随后的运动补偿过程。

4）装入用于运动补偿的顶点和像素处理程序，设置宏块点的大小（16），绘制宏块点集完成运动补偿过程，如图 10 所示。

a) 顶点处理程序主要用于对运动向量进行预处理。根据运动向量的像素精度产生对应的小数部分，以便在纹理采样时利用纹理的双线性过滤硬件自动完成像素的插值。例如对于半像素精度，小数部分为 0.5。图 8a 和图 8b 简单说明了像素插值和纹理双线性过滤过程。

b) 光栅化产生宏块大小的像素块，每个像素都继承 a)中输出的运动向量。

c) 像素处理程序中，首先利用 WPOS 寄存器得到每个像素的位置，然后利用运动向量对该位置进行偏移得出对应参考块的纹理坐标。像素处理程序采样参考帧纹理和 IDCT 输出的残差图像纹理，累加采样值并做饱和化处理，输出结果到帧缓存中。

5）帧缓存中的图像如果需要输出到显示设备上，需要进行颜色空间转化。实现过程为绘制一个图像大小的矩形，利用像素处理程序采样步骤 4) c)输出的帧缓存，并对每个像素做颜色变化，并将结果输出显示。最终完成整个解码过程。

上述步骤给出了利用 GPU 完成视频解码的全部过程，本发明中 CPU 仅用于生成和组织绘制用的点集，其他所有解码环节都在 GPU 上完成，最大限度降低了 CPU 的计算负担；通过将视频中宏块和系数块表示为点图元，整个解码过程高效的映射为点图元的绘制过程，充分发挥了 GPU 上的计算资源，借助于 GPU 硬件的并行计算和流水线处理的加速功能，本发明显著提高了视频解码的效率。

尽管为说明目的公开了本发明的具体实施例和附图，其目的在于帮助理解本发

明的内容并据以实施，但是本领域的技术人员可以理解：在不脱离本发明及所附的权利要求的精神和范围内，各种替换、变化和修改都是可能的。因此，本发明不应局限于最佳实施例和附图所公开的内容。

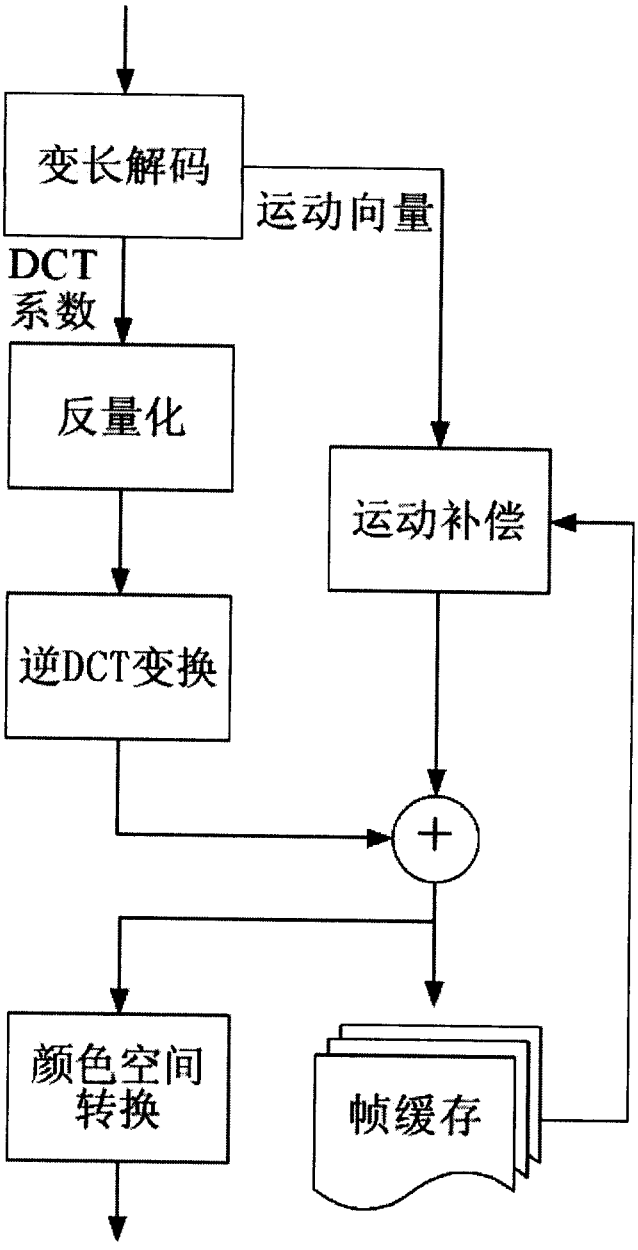


图 1

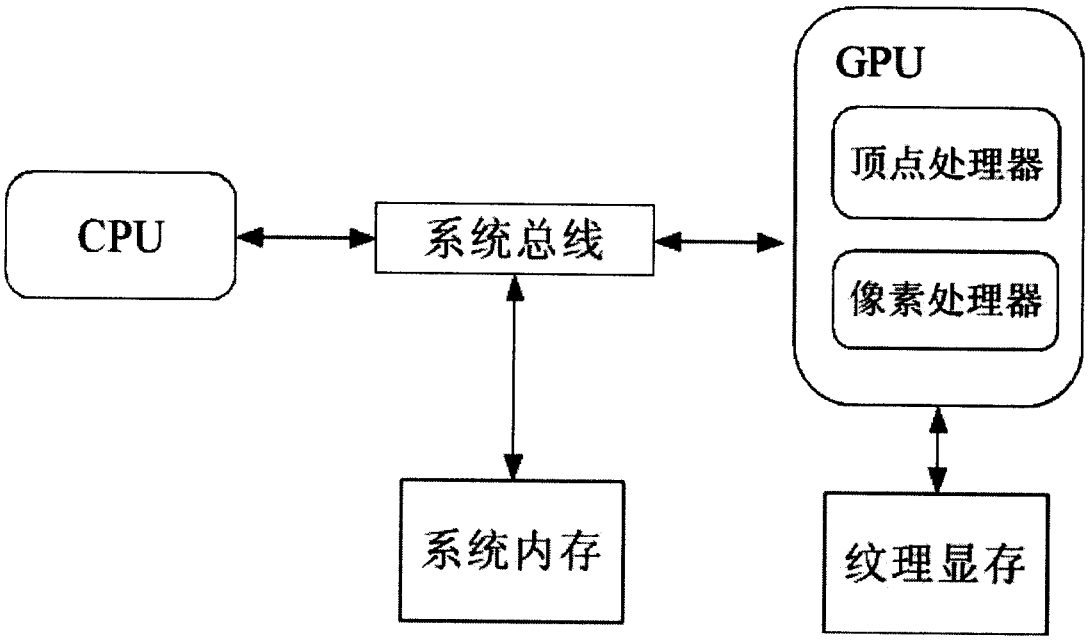


图 2

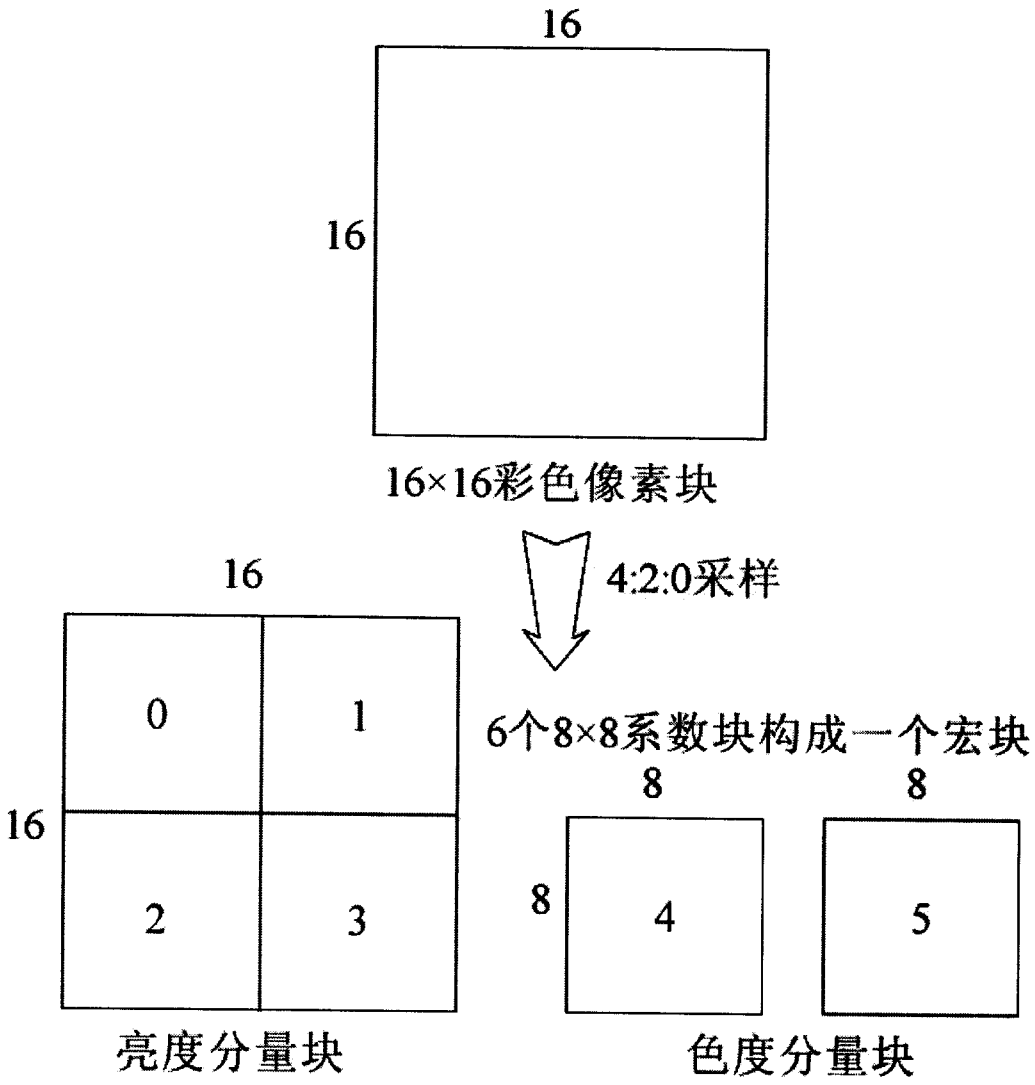

16
16
16×16彩色像素块
4:2:0采样
16
16
0
1
2
3
亮度分量块
6个8×8系数块构成一个宏块
8
8
8
4
5
色度分量块

图 3

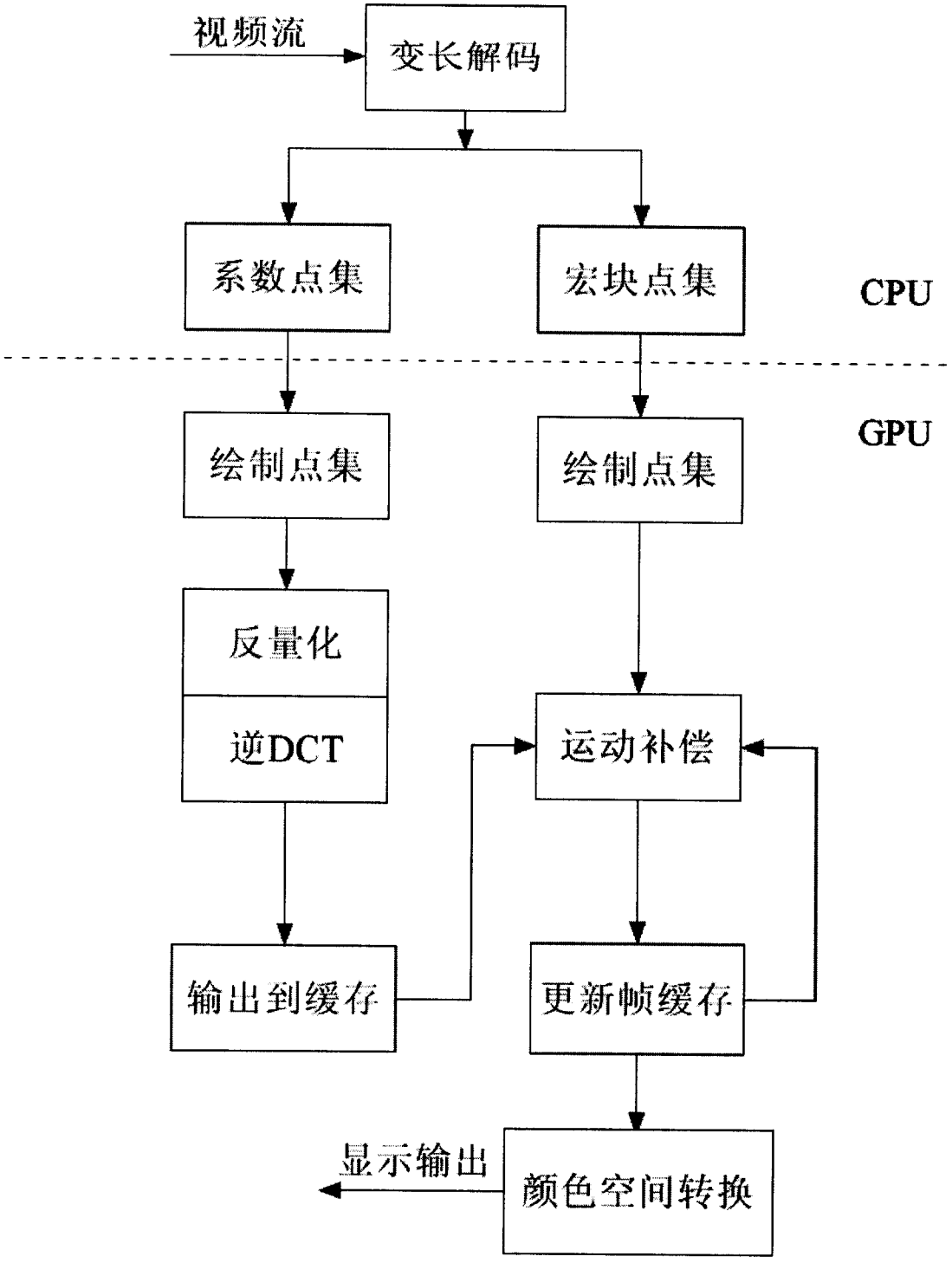

视频流
变长解码
系数点集
宏块点集
CPU
GPU
绘制点集
绘制点集
反量化
逆DCT
运动补偿
输出到缓存
更新帧缓存
显示输出
颜色空间转换

图 4

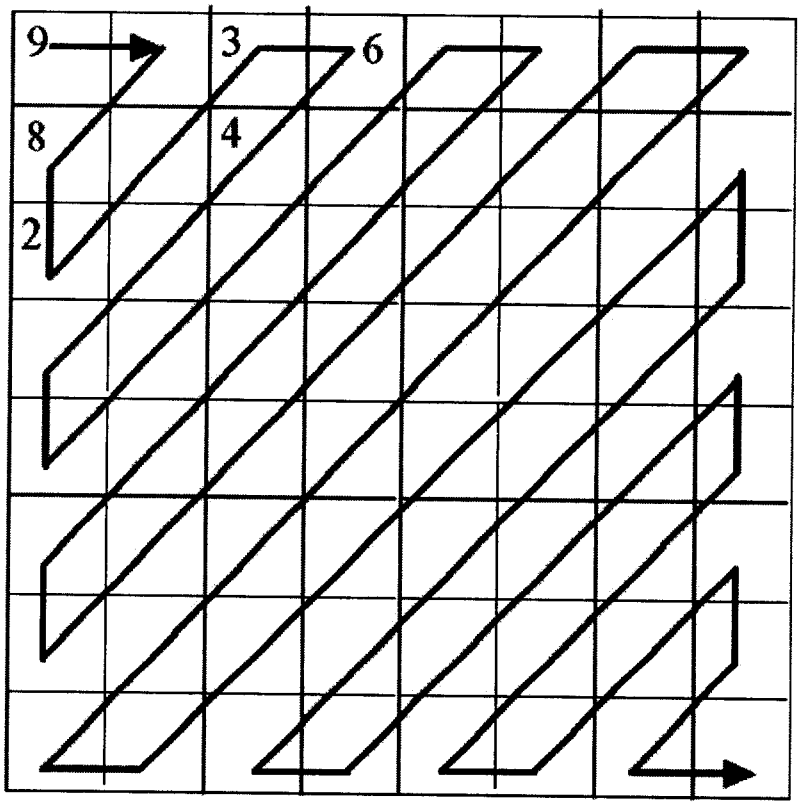
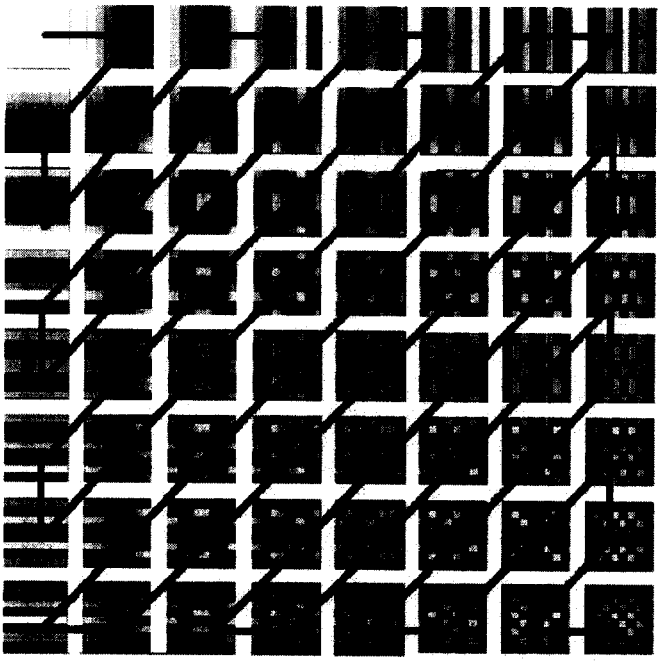


图 5



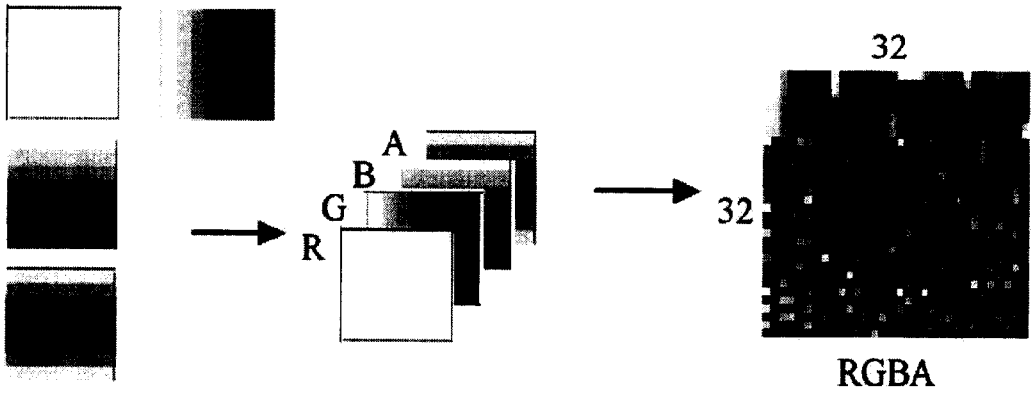


图 6

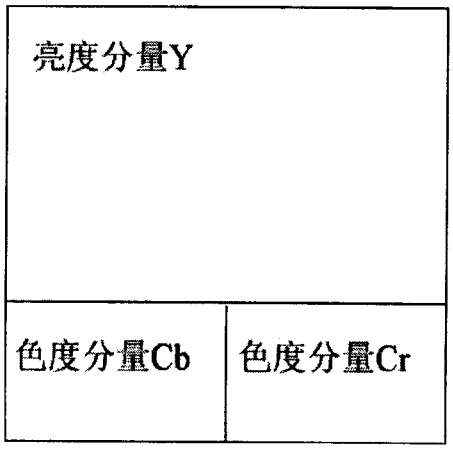


图 7a

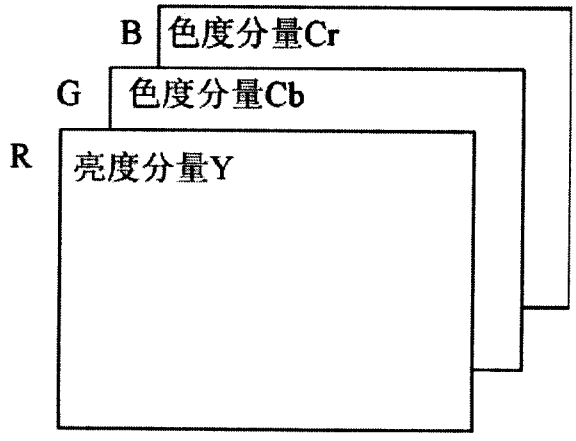
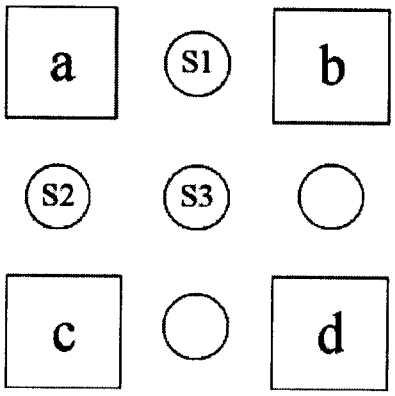


图 7b



s1=(a+b)/2
s2=(a+c)/2
s3=(a+b+c+d)/4

图 8a

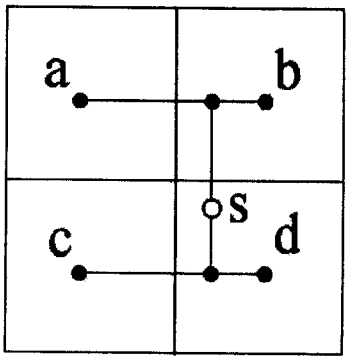


图 8b

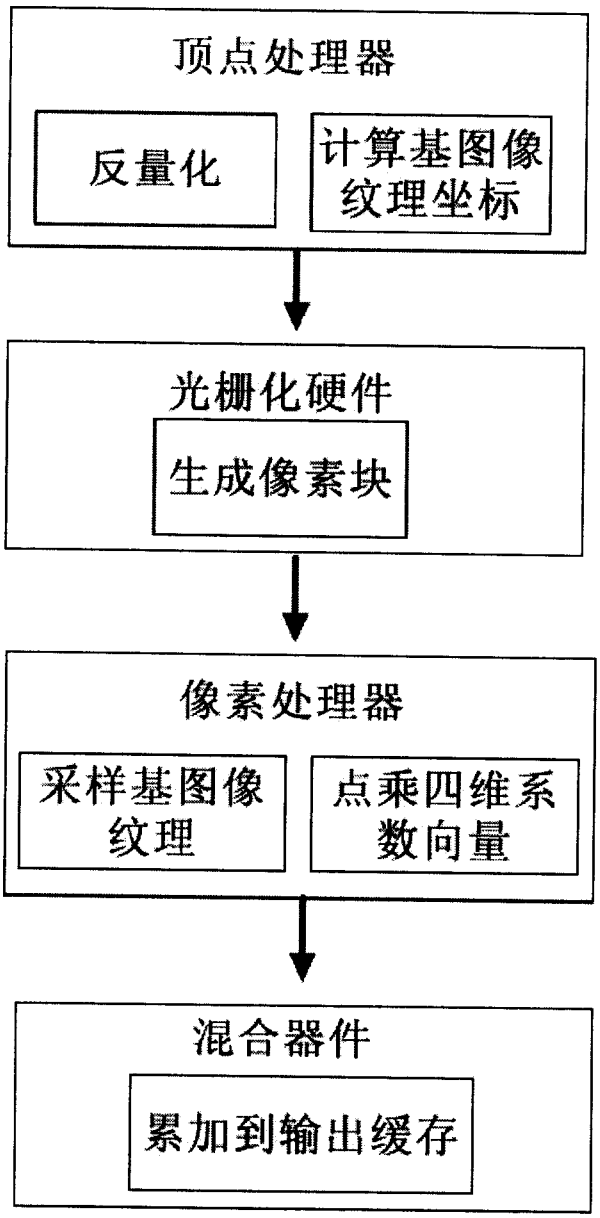

顶点处理器
反量化
计算基图像纹理坐标
光栅化硬件
生成像素块
像素处理器
采样基图像纹理
点乘四维系数向量
混合器件
累加到输出缓存

图 9

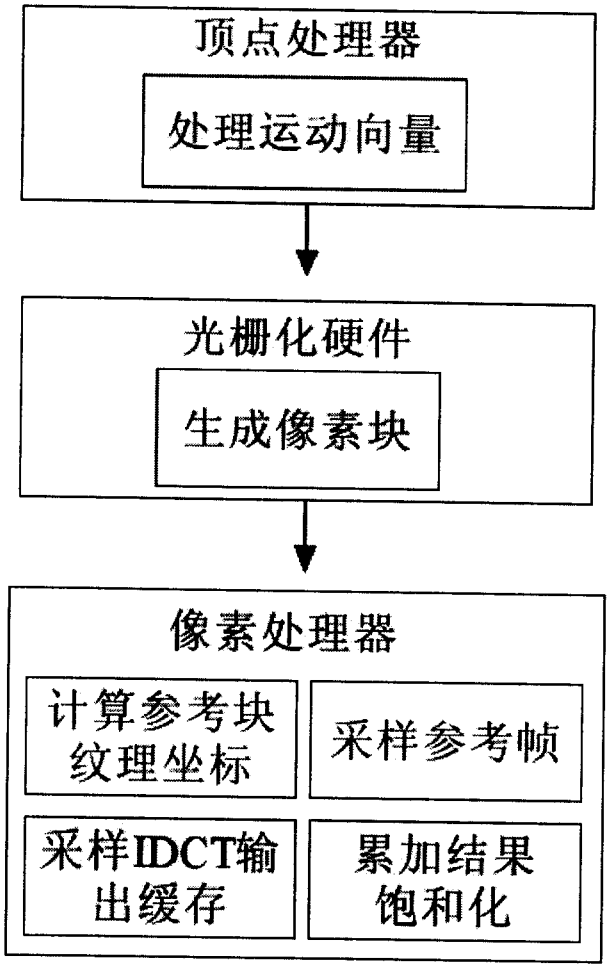

顶点处理器
处理运动向量
光栅化硬件
生成像素块
像素处理器
计算参考块纹理坐标
采样参考帧
采样IDCT输出缓存
累加结果饱和化

图 10