(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 108765282 B
(45)授权公告日 2020.10.09

(21)申请号 201810398028.3

(22)申请日 2018.04.28

(65)同一申请的已公布的文献号
申请公布号 CN 108765282 A

(43)申请公布日 2018.11.06

(73)专利权人 北京大学
地址 100871 北京市海淀区颐和园路5号

(72)发明人 罗国杰 何卓论 黄瀚贤 柏园超 贾惠柱 姜明

(74)专利代理机构 北京万象新悦知识产权代理有限公司 11360
代理人 黄凤茹

(51)Int.Cl.
G06T 3/40(2006.01)
G06T 5/10(2006.01)
G06T 7/11(2017.01)
G06N 3/08(2006.01)

(56)对比文件
CN 107967669 A,2018.04.27
US 2017339431 A1,2017.11.23
CN 106886978 A,2017.06.23
Chang J W ET AL.Optimizing FPGA-based convolutional neural networks accelerator for image super-resolution.《2018 23rd Asia and South Pacific Design Automation Conference》.2018,第343-348页.

审查员 徐灿

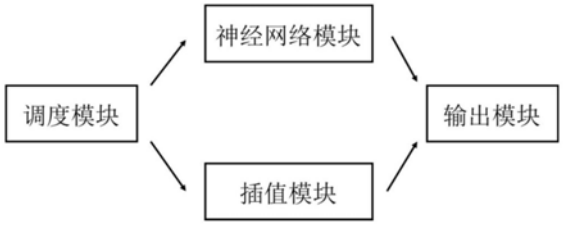
权利要求书2页 说明书11页 附图2页

(54)发明名称

基于FPGA的实时超分辨方法及系统

(57)摘要

本发明公布了一种基于FPGA的实时超分辨方法及系统,涉及图像处理技术领域;将较低分辨率的媒体中的每帧图像拆分成子图像进行超分辨处理,并根据当前帧子图像的特征值分配处理模块:若特征值较高,则由神经网络模块计算;若特征值较低,则由插值模块计算。输出缓冲区将计算得到的高分辨率子图像输出并重组成高分辨率图像,用户便可以观看到实时的超高清晰度画面。



CN 108765282 B

1.一种基于FPGA的实时超分辨方法,包括如下步骤:

1)将当前帧图像按步长拆分成子图像,通过测量函数计算每个子图像的特征值;每个子图像的特征值由全变分TV值判定,采用全变分TV分类方法作为掩蔽测量M;具体将当前帧图像裁剪成N×N像素的子图像,对于每个子图像,通过测量函数M:$R^{N\times N}$→R计算特征值;

2)设定特征值阈值,将当前帧子图像进行分配,通过计算得到高分辨率图像:

若特征值高于设定的阈值,则通过神经网络进行计算;否则采用插值方法进行计算;

3)将经过步骤2)由神经网络或由插值方法计算得到的高分辨率图像输出,并根据步骤1)的拆分顺序重新组合成高分辨率图像;

4)上述步骤1)~3)中的全部或部分在FPGA上实现:通过FPGA综合工具以流水线的形式配置成FPGA的计算电路,每一步骤为流水线的一个阶段;或将其中一个或多个步骤通过FPGA综合工具配置成FPGA的计算电路,进行局部的加速;

由此实现基于FPGA的实时超分辨。

2.如权利要求1所述基于FPGA的实时超分辨方法,其特征是,步骤1)具体采用具有各向异性的全变分TV值;具体将N×N子图像视为Z中的二维矩阵,其中Z是欧几里得空间$R^{N\times N}$;引入离散梯度算子∇:Z→Z×Z定义离散的TV;如果x∈Z,∇x是Z×Z中的一个向量,由式1表示:

$$(\nabla x)_{i,j}=((\nabla x)^{V}_{i,j},\ (\nabla x)^{H}_{i,j}) \qquad (式\ 1)$$

其中:

$$(\nabla x)^{V}_{i,j}=\begin{cases} x_{i+1,j}-x_{i,j} & if\ \ i<N \\ 0 & if\ \ i=N \end{cases}$$

$$(\nabla x)^{H}_{i,j}=\begin{cases} x_{i,j+1}-x_{i,j} & if\ \ j<N \\ 0 & if\ \ j=N \end{cases}$$

其中i,j=1,2,...,N;

总的TV值被定义为式2:

$$J(x)=\sum_{1\le i,j\le N}||(\nabla x)_{i,j}||_1 \qquad (式\ 2)$$

其中,$||y||_1=|y_1|+|y_2|$,对于y=(y1,y2)∈R^2;

具体采用微体系结构将访存与计算解耦;访存包括TV计算中所有变量值的获取和更新;计算包括变量值之间的运算;微体系结构包括配备内存控制器和数据互连的缓冲系统;缓冲系统彼此独立;在每个缓冲系统中,FIFO提供的存储与常规数据复用缓冲区相同,FIFO之间的数据路径分离器和滤波器用作内存控制器和数据互连;每个缓冲系统接收一个数据流,无需额外的外部存储器访问;

在计算开始之前,控制器首先读入数据并将FIFO填充多个时钟周期;然后在每个时钟周期中,滤波器将所需数据发送到计算内核,内核消耗所有数据以生成一个输出,控制器将所有缓冲数据向前移动;直到迭代域结束。

3.如权利要求1所述基于FPGA的实时超分辨方法,其特征是,步骤2)神经网络采用沙漏形卷积神经网络FSRCNN-s;

将FSRCNN-s的卷积层表示为Conv(c_i,f_i,n_i),将反卷积层表示为DeConv(c_i,f_i,n_i),其中变量c_i,f_i和n_i分别代表信道数量,滤波器尺寸和滤波器数量;将FSRCNN-s分解为多个阶段/层;

所述卷积层和反卷积层可在FPGA上以流水线的形式统一实现,每个层为流水线一个阶

段;或将一个或多个卷积层和反卷积层在FPGA上单独实现。

4.如权利要求3所述基于FPGA的实时超分辨方法,其特征是,将FSRCNN-s分解为五个阶段/层,包括:

a1)特征提取Conv(1,5,32):使用大小为5×5的滤波器从原始LR图像中提取32个特征映射;

a2)缩小Conv(32,1,5):使用大小为1×1的滤波器将LR特征维度从32减少到5;

a3)映射Conv(5,3,5):使用大小为3×3的滤波器非线性地将LR特征映射到HR特征上;

a4)扩展Conv(5,1,32):使用大小为1×1的滤波器将HR特征维度从5扩展到32;

a5)反卷积DeConv(32,9,1):使用大小为9×9的滤波器上采样并聚合先前的特征。

5.如权利要求4所述基于FPGA的实时超分辨方法,其特征是,FSRCNN-s在每个卷积层之后使用参数整形线性单元PReLU;激活函数定义为式3:

$f(x_i)=\max(x_i,0)+a_i\min(x_i,0)$ (式3)

其中,xi是第i个信道上激活f的输入信号;a_i是负数部分的系数,在PReLU中可以学习;

FSRCNN-s采用均方误差MSE作为成本函数,优化目标表示为式4:

$$min_\theta \frac{1}{n}\sum_{i=1}^{n}||F(Y_s^i;\theta)-X^i||_2^2 \quad (式4)$$

其中,Y_s^i和X^i是训练数据中的第i个LR和HR子图像对,并且$F(Y_s^i;\theta)$是具有参数θ的Y_s^i的网络输出;所有参数均使用随机梯度下降与标准反向传播进行优化。

6.如权利要求1所述基于FPGA的实时超分辨方法,其特征是,步骤2)所述神经网络采用流水线架构。

7.如权利要求1所述基于FPGA的实时超分辨方法,其特征是,步骤2)所述插值方法具有较低的运算代价;所述插值方法采用双线性插值方法。

8.一种实现权利要求1～7任一项所述基于FPGA的实时超分辨方法的基于FPGA的实时超分辨系统,包括:输入模块、调度模块、计算模块、输出缓冲区;其中,计算模块为可独立工作的模块,包括神经网络模块和/或插值模块;各模块具体为:

调度模块用于决定当前帧子图的分配;若当前帧子图的特征值高于设定的阈值,则通过神经网络进行计算;否则采用插值方法进行计算;

神经网络模块用于计算具有高特征值的图像;

插值模块用于计算具有低特征值的图像;

输出缓冲区用于将计算得到的高分辨率图像输出。

9.如权利要求8所述基于FPGA的实时超分辨系统,其特征是,所述实时超分辨系统的核心运算部分包括神经网络模块、插值模块和其他能实现超分辨的模块中的两个或多个,并根据当前处理图像的特征值选取并调用其中一个模块进行计算;所述神经网络模块为可产生较高分辨率内容的流水线架构;所述插值模块具有较低的运算代价。

基于FPGA的实时超分辨方法及系统

技术领域

[0001] 本发明涉及图像处理技术领域，具体涉及一种基于现场可编程门阵列(FPGA)的实时超分辨方法及系统。

背景技术

[0002] 超高清(UHD)技术正在显著地改变着媒体行业，目前高分辨显示器占据了市场主导地位。然而，超高清媒体资源供应有限，并且由于网络带宽不足导致访问困难，用户 实际体验仍然一般。因此，将1920×1080的传统高清(2K FHD)视频高效地升级到4K 超高清分辨率(3940×2160)意义重大。从低分辨率输入估算细粒度分辨率图像/视频的 技术通常被称为超分辨，在图像处理和计算机视觉领域，这个基本问题非常具有吸引力。

[0003] 基于现场可编程门阵列(FPGA)的神经网络加速器由于具有比GPU更高的能效， 和比专用集成电路(ASIC)更短的开发周期而逐渐受到广泛关注。由于卷积运算经常占 据神经网络总操作的很大比例，以往的大部分工作都集中在优化卷积上。许多加速器专 注于提高计算效率，通过提高并行度，计算序列(流水线)和基于循环展开和循环优化 等技术实现的计算-通信平衡。人们还在努力减少计算需求，通过频域加速，二值化/变 形网络和网络压缩，还提出了硬件抽象和端到端自动化框架等方法。

[0004] 已有的基于迭代反投影算法的实时超分辨率技术结合并修改了基于模型的超分辨 率算法，假设帧之间具有相同的模糊效果(用于计算效率)，以及使用L1最小化的迭 代算法。该方法使用固定点精度，并为实时提出了高度流水线的架构。

[0005] 目前，超分辨最直接的实现方法是插值算法，包括最近邻、双线性、双三次以及Lanczos算法。这些算法通常运行速度快且易于实现，但也不可避免地会产生模糊的结 果。基于模型的方法旨在根据观测模型和先验(正则化)来恢复高分辨率场景，现有技 术大多假定已知的模糊核和噪声水平，但实际情况中模糊核和噪声水平可以是任意的， 假定已知给实际应用造成了困难，效果不佳。基于神经网络的超分辨往往运算代价大， 在对视频实时处理时往往会面临输出帧率不足的问题。

发明内容

[0006] 为了克服上述现有技术的不足，本发明提供了一种基于FPGA的实时超分辨方法及系统，将较低分辨率的媒体中的每帧图像拆分成子图像，并根据其特征值，将当前帧子 图像分配：若特征值较高，则由神经网络模块计算；若特征值较低，则由插值模块计算。输出缓冲区将计算得到的高分辨率图像输出并重组成高分辨率图像，用户便可以观看到 实时的超高分辨率画面。

[0007] 本发明的技术方案是：

[0008] 一种基于FPGA的实时超分辨方法，可全部或部分在FPGA上实现，包括如下步骤：

[0009] 1)首先将当前帧图像按步长拆分成子图像，通过测量函数计算每个的特征值；

[0010] 优选地，特征值由全变分(TV)值判定。

[0011] 具体地，将当前帧图像裁剪成N×N像素的子图像，对于每个子图像，通过测量函数$M:R^{N\times N}\rightarrow R$计算其特征值；

[0012] 采用全变分(Total Variation,TV)分类方法作为掩蔽测量M。为了简化计算，采用具有各向异性的TV值。将N×N子图像视为Z中的二维矩阵，其中Z是欧几里得空间$R^{N\times N}$。为了定义离散的TV，引入了离散(线性)梯度算子$\nabla: Z\rightarrow Z\times Z$。如果$x\in Z$，$\nabla x$是Z×Z 中的一个向量，它由式1给出：

[0013] $$(\nabla x)_{i,j}=((\nabla x)^{V}_{i,j},(\nabla x)^{H}_{i,j}) \qquad (式\ 1)$$

[0014] 其中：

[0015] $$(\nabla x)^{V}_{i,j}=\begin{cases}x_{i+1,j}-x_{i,j} & if\ i<N\\ 0 & if\ i=N\end{cases}$$

[0016] $$(\nabla x)^{H}_{i,j}=\begin{cases}x_{i,j+1}-x_{i,j} & if\ j<N\\ 0 & if\ j=N\end{cases}$$

[0017] 其中i,j＝1,2,…,N.

[0018] 则总的TV值J(x)被定义为式2：

[0019] $$J(x)=\sum_{i\le i,j\le N}||(\nabla x)_{i,j}||_1 \qquad (式\ 2)$$

[0020] 其中||y||＝|y_1|+|y_2|，对于y＝(y1,y2)$\in R^2$。

[0021] 计算得到的TV值，与提前设定的阈值比较，用于处理当前子图像模块的分配。

[0022] 本发明采用微体系结构将访存与计算解耦，访存包括TV计算公式中所有变量值的获取和更新，计算包括变量值之间的运算(做差、取范数等)。微体系结构主要包含配 备内存控制器和数据互连的缓冲系统。在不同的数组中没有数据重用的机会，所以缓冲 系统彼此独立。在每个缓冲系统中，FIFO提供的存储与常规数据复用缓冲区相同，而 FIFO之间的数据路径分离器和滤波器则用作内存控制器和数据互连。每个缓冲系统接收 一个数据流而无需额外的外部存储器访问。在计算开始之前，控制器首先读入数据并将FIFO填充N个周期。然后在每个时钟周期中，滤波器将所需数据发送到计算内核，内核 消耗所有数据以生成一个输出，控制器将所有缓冲数据向前移动。以这种方式，缓冲系 统继续进行，直到迭代域结束。

[0023] 2)并将当前帧子图像进行分配：设定阈值，若特征值高于设定的阈值，则由神经网络模块计算；若特征值低于设定的阈值，则由插值模块计算；得到高分辨率图像；

[0024] 具体实施时，神经网络计算或插值计算可采用多种实现方式；

[0025] 3)输出缓冲区将经过步骤2)由神经网络模块或由插值模块计算得到的高分辨率图 像输出并根据拆分顺序重新组合成高分辨率图像；

[0026] 通过上述步骤，实现基于FPGA的实时超分辨。

[0027] 优选地，步骤1)～3)全部在FPGA上实现，或选其中FPGA可容纳部分在FPGA 上实现。步骤1)～3)可通过FPGA综合工具以流水线的形式配置成FPGA的计算电路， 每一步骤为流水线的一个阶段；这样充分利用了FPGA的可重构性和并行性，将串行的 算法流水化，可以增加系统的吞吐率和处理延迟；或者可以只将算法中的一个或几个步 骤通过FPGA综合工具配置成FPGA的计算电路，进行局部的加速。

[0028] 优选地，步骤2)采用沙漏形卷积神经网络，即FSRCNN-s。该神经网络可以学习原

始LR和目标HR图像之间的端到端映射，无需预处理。FSRCNN-s具有如下特征：将 卷积层表示为Conv(c_i,f_i,n_i)，将反卷积层表示为DeConv(c_i,f_i,n_i)，其中变量c_i,f_i和 n_i分别代表信道数量，滤波器尺寸和滤波器数量。FSRCNN-s可以分解为以下五个阶段 (层)。

[0029] 1)特征提取Conv(1,5,32)使用大小为5×5的滤波器从原始LR图像中提取32个特征映射。

[0030] 2)缩小Conv(32,1,5)使用大小为1×1的滤波器将LR特征维度从32减少到5。

[0031] 3)映射Conv(5,3,5)使用大小为3×3的滤波器非线性地将LR特征映射到HR特征上。

[0032] 4)扩展Conv(5,1,32)使用大小为1×1的滤波器将HR特征维度从5扩展到32。

[0033] 5)反卷积DeConv(32,9,1)使用大小为9×9的滤波器上采样并聚合先前的特征。

[0034] 基于FPGA的重构性，这些卷积层和反卷积层既可以在FPGA上以流水线的形式统一实现(每个层为流水线一个阶段)，也可以将一个或几个层在FPGA上单独实现，因此基 于FPGA的系统较串行实现可以获得很高的加速比，且具有很高的灵活性。

[0035] FSRCNN-s在每个卷积层之后使用参数整形线性单元(PReLU)。激活函数定义为式3：

[0036] $f(x_i)=\max(x_i,0)+a_i\min(x_i,0)$ (式3)

[0037] 其中，xi是第i个信道上激活f的输入信号，a_i是负数部分的系数。与将参数a_i固 定为零的ReLU不同，该参数在PReLU中可以学习。

[0038] FSRCNN-s采用均方误差(MSE)作为成本函数。优化目标表示为式4：

[0039] $$min_\theta \frac{1}{n}\sum_{i=1}^{n}||F(Y_s^i;\theta)-X^i||_2^2 \qquad (式4)$$

[0040] 其中，Y_s^i和X^i是训练数据中的第i个LR和HR子图像对，并且$F(Y_s^i;\theta)$是具有参数θ的Y_s^i的网络输出。所有参数都使用随机梯度下降与标准反向传播进行优化。

[0041] 优选地，步骤2)所述神经网络模块为流水线架构，可以产生较高分辨率内容。

[0042] 优选地，步骤2)所述插值模块应具有较低的运算代价。

[0043] 本发明还提供了一种基于FPGA的实时超分辨系统，包括以下模块：输入模块、调度模块、计算模块、输出模块(输出缓冲区)；其中，计算模块包括神经网络模块和插 值模块；各模块具体为：

[0044] 调度模块，决定当前帧子图的分配；

[0045] 神经网络模块，计算具有高特征值的图像，结果精确；

[0046] 插值模块，计算具有低特征值的图像，速度较高；

[0047] 输出缓冲区，将计算得到的高分辨率图像输出。

[0048] 优选地，所述实时超分辨系统的核心运算部分可以包括两个或多个可独立工作的模 块，即神经网络模块、插值模块，或其余基于模型的方法或基于数据的方法进行超分辨的模块，并且根据当前处理图像的特征来选取并调用其中一个工作。

[0049] 优选地，所述神经网络模块为流水线架构，可以产生较高分辨率内容。

[0050] 优选地，所述插值模块应具有较低的运算代价。

[0051] 与现有技术相比，本发明的有益效果是：

[0052] 本发明提供了一种基于FPGA的实时超分辨方法及系统，将较低分辨率的媒体中的

每帧图像拆分成子图像，并根据其特征值，将当前帧子图像分配：若特征值较高，则由 神经网络模块计算；若特征值较低，则由插值模块计算。输出缓冲区将计算得到的高分 辨率图像输出并重组成高分辨率图像，用户便可以观看到实时的超高分辨率画面。

附图说明

[0053] 此处所说明的附图用来提供对本发明的进一步理解，构成本申请的一部分，本发明 的示意性实施例及其说明用于解释本发明，并不构成对本发明的不当限定。在附图中：

[0054] 图1是本发明提供的基于FPGA的实时超分辨系统的结构框图。

[0055] 图2是全变分计算的访问模式，深色图块影响梯度偏置；

[0056] 其中，1是当前像素点，2是右侧像素点，3是下方像素点。

[0057] 图3是本发明采用的全变分计算的微体系结构示意图；

[0058] 其中，s1，s2，s3是分配器，f1，f2，f3是滤波器；(5)是卷积输入，(6)是卷积计 算，(7)是卷积输出；

[0059] 控制器(分配器)首先读入数据并将FIFO填充N个周期；然后在每个时钟周期中，滤波器将所需数据发送到计算内核，内核消耗所有数据以生成一个输出，分配器将所有 缓冲数据向前移动；以这种方式，缓冲系统继续进行，直到迭代域结束。

[0060] 图4是本发明实施例采用的神经网络的卷积层的结构示意图，包括$f_i \times f_i$的滑动窗口，f_i为第i层的滤波器维度；

[0061] 其中，4是卷积滑动窗口，5是卷积输入，6是卷积计算，7是卷积输出。

[0062] 图5是本发明实施例采用的神经网络的反卷积层的结构示意图，包括$f_i \times f_i$的滑动窗口，f_i为第i层的滤波器维度：

[0063] 其中，8是反卷积输入，9是反卷积计算，10是反卷积输出，11是反卷积滑动窗口。

[0064] 图6是本发明实施例中六种不同配置方式的输出截图；

[0065] 其中，(a)是无分配/直接插值输出，(b)是无分配/神经网络输出，(c)是调度 /直接插值输出，(d)是调度/神经网络输出，(e)是调度/随机混合输出，(f)是调度 /TV值混合输出。

具体实施方式

[0066] 下文中将参考附图并结合实施例来详细说明本发明。需要说明的是，在不冲突的情况下， 本申请中的实施例及实施例中的特征可以相互组合。

[0067] 本实施例提供了一种基于FPGA的实时超分辨技术与设备，图1是根据本发明实施例的结 构框图，它将神经网络和基于插值的方法结合起来。

[0068] 步骤1，给定一个低分辨率(LR)图像X，首先将其裁剪成$N \times N$像素的子图像，步长为k。

[0069] 步骤2，对于每个子图像，通过测量函数$M:R^{N\times N}\rightarrow R$计算其特征值。

[0070] 步骤3，具有高特征值的子图像使用神经网络放大，而其余的仅通过插值放大。

[0071] 步骤4，将放大的子图像组合成高分辨率(HR)图像Y。

[0072] 上述用于超分辨率的算法的伪代码如下：

[0073] Input:LR image X,upscaling factor n,threshold T

[0074] Output:HR image Y

1: Crop **X** into sub-images {x} with a stride k
2: **for all** sub-image x do
3: **if** M(x)≥T **then**
4: y←Upscale(x)

[0075] 5: **else**
6: y←CheapUpscale(x)
7: **end if**
8: **end for**
9: Mosaic **Y** with upscaled sub-images {y}

[0076] 优选地,采用全变分(Total Variation,TV)分类方法作为算法1中的掩蔽测量M。注意，TV的各向异性版本被用于更简单的计算。将N×N子图像视为Z中的二维矩阵,其中Z是欧几里 得空间$R^{N\times N}$。为了定义离散的TV,引入了离散(线性)梯度算子$\nabla: Z\rightarrow Z\times Z$。如果x∈Z,$\nabla x$是Z×Z中的一个向量,它由式1给出：

[0077] $$(\nabla x)_{i,j}=((\nabla x)^{V}_{i,j},(\nabla x)^{H}_{i,j}) \qquad (式\ 1)$$

[0078] 其中：

[0079] $$(\nabla x)^{V}_{i,j}=\begin{cases}x_{i+1,j}-x_{i,j} & if\ i<N\\ 0 & if\ i=N\end{cases}$$

[0080] $$(\nabla x)^{H}_{i,j}=\begin{cases}x_{i,j+1}-x_{i,j} & if\ j<N\\ 0 & if\ j=N\end{cases}$$

[0081] 其中i,j=1,2,…,N.

[0082] 则总的TV值被定义为式2：

[0083] $$J(x)=\sum_{i\le i,j\le N}||(\nabla x)_{i,j}||_1 \qquad (式\ 2)$$

[0084] 其中||y||=$|y_1|+|y_2|$,对于y=(y1,y2)∈R^2

[0085] 优选地,选择TV(Total Variation)作为分类方法,原因如下:1)TV值显示图像 块的高频强度。高TV值带有更多高频信息,如边缘和纹理,这些信息无法通过插值方 法很好地恢复。2)自然图像块的TV值分布接近瑞利分布。因此,可以通过设置合理的 阈值,筛选出一部分区块。3)TV值很容易计算。

[0086] 优选地,采用沙漏形卷积神经网络,即FSRCNN-s。该神经网络可以学习原始LR和目标HR图像之间的端到端映射,无需预处理。FSRCNN-s具有如下特征:将卷积层表 示为Conv(c_i,f_i,n_i),将反卷积层表示为DeConv(c_i,f_i,n_i),其中变量c_i,f_i和n_i分别 代表信道数量,滤波器尺寸和滤波器数量。FSRCNN-s可以分解为以下五个阶段(层)。

[0087] 1)特征提取Conv(1,5,32)使用大小为5×5的滤波器从原始LR图像中提取32个特征映射。

[0088] 2)缩小Conv(32,1,5)使用大小为1×1的滤波器将LR特征维度从32减少到5。

[0089] 3)映射Conv(5,3,5)使用大小为3×3的滤波器非线性地将LR特征映射到HR特征上。

[0090] 4)扩展Conv(5,1,32)使用大小为1×1的滤波器将HR特征维度从5扩展到32。

[0091] 5)反卷积DeConv(32,9,1)使用大小为9×9的滤波器上采样并聚合先前的特征。

[0092] FSRCNN-s在每个卷积层之后使用参数整形线性单元(PReLU)。激活函数定义为式3：

[0093] $f(x_i)=\max(x_i,0)+a_i\min(x_i,0)$ (式3)

[0094] 其中,xi是第i个信道上激活f的输入信号,a_i是负数部分的系数。与将参数a_i固 定为零的ReLU不同,该参数在PReLU中可以学习。

[0095] FSRCNN-s采用均方误差(MSE)作为成本函数。优化目标表示为式4：

[0096] $$min_\theta \frac{1}{n}\sum_{i=1}^{n}||F(Y_s^i;\theta)-X^i||_2^2$$ （式4）

[0097] 其中,Y_s^i和X^i是训练数据中的第i个LR和HR子图像对,并且$F(Y_s^i;\theta)$是具有参数 θ的Y_s^i的网络输出。所有参数都使用随机梯度下降与标准反向传播进行优化。

[0098] 优选实施例一:由三个主要部分组成：

[0099] • 调度模块:根据前文中提到的等式计算每个块的TV值。然后将其TV值大于预先确定的阈值的块分配给神经网络,而将其他值分配给插值模块。

[0100] • 流水线神经网络:FSRCNN-s是在流水线阶段实现的流水线结构。每个阶段的乘法 器数量也被配置为实现阶段的均衡吞吐量。

[0101] • 插值模块:简单且快速的插值模块,处理TV值低于阈值的模块。双线性算法能够以较低的运算成本和较好的性能提高图像质量。最后,从网络或内插组件输出的数据块 进行组合以生成最终输出的高分辨率图像。

[0102] 在基于TV的掩膜中,像素的垂直和水平梯度分别依赖于它本身、它之下和右侧的像素。计算模式可以被认为是模板计算,即,每个点作为其值和其相邻元素处的值的函 数被更新(迭代)。图2描述了计算的访问模式。要计算某一像素x[offset]处的渐变， 必须访问图中深色的像素,包括该像素本身(x[offset]),该像素右侧的像素(x[right]) 和该像素下方的像素(x[down])。

[0103] 当迭代计算完全流水线化时,计算内核需要从一个阵列在一个时钟周期内中加载多 个元素,所以内存分区是必要的,以避免存储器端口争用。本发明采用微体系结构来将访存与计算解耦。如图3所示,微架构主要包含配备内存控制器和数据互连的缓冲系统。 在不同的数组中没有数据重用的机会,所以缓冲系统彼此独立。在每个缓冲系统中,FIFO 提供的存储与常规数据复用缓冲区相同,而FIFO之间的数据路径分离器和滤波器则用 作内存控制器和数据互连。每个缓冲系统接收一个数据流而无需额外的外部存储器访问。 在计算开始之前,控制器首先读入数据并将FIFO填充N个周期。然后在每个时钟周期 中,滤波器将所需数据发送到计算内核,内核消耗所有数据以生成一个输出,控制器将 所有缓冲数据向前移动。以这种方式,缓冲系统继续进行,直到迭代域结束。表1显示 了缓冲系统的填充过程。

[0104] 表1缓冲系统的填充过程

[0105]

时钟周期	数据流	滤波器状态			FIFO 数据状态	
		滤波器 1	滤波器 2	滤波器 3	FIFO1	FIFO2
1	x[0][0]	d	d	f→s	0	0
2	x[0][1]	d	f→s	s	0	1
N	x[0][N-1]	d	s	s	N-2	1
N+1	x[1][0]	d→f	s→f	s→f	N-1	1
N+1-…	x[1][1]-…	f	f	f	N-1	1

[0106] 为了提高系统吞吐量,本发明实例将整个神经网络组织为流水线结构,每个网络层 都作为流水线阶段。所有的特征图和权重,以及偏向量和PReLU参数都存储在BRAM 中。可以保留芯片上的数据,主要是因为神经网络尺寸较小和分块算法导致小特征映射。表2提供了以下各节中使用的符号。

[0107] 表2所用符号含义说明

[0108]

符号	含义
c_i	第 i 层的输入信道数量
f_i	第 i 层的滤波器维度
n_i	第 i 层的输出信道数量
$Conv(c_i, f_i, n_i)$	第 i 个卷积层
N_i	第 i 层的输入特征映射维度
k	子图像步长
$\#Conv$	卷积层数量
S_{HD}	全高清图像尺寸
s	倍增因数
F_r	框架运行速率（fps）
$\mathcal{B}$	I/O 带宽（bps）
$\mathcal{C}$	BRAM 容量（bits）
$\mathcal{WL}$	字长（bits）

[0109] 本实施例中采用的神经网络具有如下要点：

[0110] 1)卷积层:每个卷积层Conv(c_i,f_i,n_i)包括大小为$f_i \times f_i$的$c_i \times n_i$个滤波器,产生n_i个 输出。使用了并行计算的$c_i \times n_i$处理单元(PE),即每个过滤器有一个PE。处理过程中 有三个主要步骤:1.输入在每个输入特征映射上的$f_i \times f_i$滑动窗口生成f^2个元素的输入向量。2.计算相应的PE计算输入向量和滤波器的内积。3.输出部分和被累加并存储在目标 像素中。这三个步骤以流水线方式执行。图4是卷积层架构示意图。

[0111] 2)反卷积层:神经网络中的反卷积可以看作是卷积的结构逆过程。反卷积层DeConv(c_i,f_i,n_i)使用尺寸为$f_i \times f_i$的的$c_i \times n_i$个滤波器取样,并综合先前的c_i映射图。由于存储器端口限制和中间数据的重用,滑动窗口也应用于反卷积层。滑动窗口保留部分 结

果并更新。该层还具有三级流水线:1.输入,输入像素是从最后一个卷积层的输出特 征映射中获得的。2.计算,输出像素使用输入像素和滤波器进行计算。3.输出,滑动窗口 每次更新目标特征映射上的列。值得注意的是其余的f_{i-s}列保留在窗口中以供进一步重 用,并且新的s列像素被初始化为零。图5描绘了反卷积层架构。

[0112] 3)流水线平衡:资源分配也被考虑,以平衡整个流水线。在卷积阶段Conv(c_i,f_i,n_i) 中,有$c_i \times n_i \times f_i^2 \times N_{i+1}^2$次乘法运算(参数符号含义见表2),需注意$N_{i+1}$是该层输出特征映射的维数。为了平衡每个阶段的吞吐量,应该在阶段中分配每个阶段中乘法器数量(DSP)的数量,使其与阶段中的乘法次数成比例,同时保持总体利用率超过可用DSP 总量。表3显示了每一层和相关数据的乘数分配。根据乘法器的乘法数成比例地分配乘 法器,来获得理想的每个DSP的数量1(理想#DSP)。然后相应地计算理想的II。需要 手动设置每层的II(从而实现必要的性能),并获得实现这种II的所需数量的DSP。

[0113] 表3资源分配与相关数据

[0114]

层	c_i	f_i	n_i	N_i	#Mult	理论值		分配值	
						#DSP	II	#DSP	II
提取	1	5	32	36	819200	201	4076	200	4096
缩小	32	1	5	32	163840	40	4096	32	4096
映射	5	3	5	32	202500	50	4050	45	4500
扩展	5	1	32	30	144000	35	4115	32	4500
反卷积	32	9	1	30	2332800	573	4072	519	4500
总计	-	-	-	-	3662340	899	4115	828	4500
可用(ZC706)	-	-	-	-	-	900	-	900	-

[0115] 本发明实施例中使用双线性插值方法作为神经网络方法的替代(即算法1中的CheapUpscale)。从输出的角度来看,双线性插值与反卷积过程非常相似。例如,在2倍 放大的情况下,输入像素$X_{i,j}$将其值扩展到以反卷积核的输出,以$Y_{2i,2j}$为中心的3×3窗 口:$\begin{bmatrix}1/4 & 1/2 & 1/4\\ 1/2 & 1 & 1/2\\ 1/4 & 1/2 & 1/4\end{bmatrix}$。同样,这种结构可以使用滑动窗口来避免大量的加载/存储寻址。 该方法还解释了为什么可以采用去卷积来放大而不是预放大。

[0116] 子图像步长k以效率和质量都会影响系统性能。为了生成一个$f_i \times f_i$滤波器的有效卷积结果,应该用尺寸为$\frac{f_i-1}{2}$的额外边界来放大输入特征映射。因此,要通过所有卷积层 获得有效的k×k输出,应有:

[0117] $$N_i \equiv k + \sum_i^{\#conv}(f_i - 1) \qquad (式\ 5)$$

[0118] 应该考虑步长/步幅k的几个约束条件:

[0119] 1)I/O带宽约束。由于每个子图像必须用额外的像素进行放大才能进行卷积,因此小跨度会带来大的边框与块的比率,这会导致I/O带宽的低效利用。为了满足I/O 带宽限制,有:

[0120] $$\left(\frac{N_1}{k}\right)^2 \times S_{HD} \times F_r \times \mathcal{WL} < \mathcal{B} \qquad (式\ 6)$$

[0121] 2)存储容量限制。高步长会造成大尺寸的特征映射,这使得将所有特征地图存储在 芯片上变得困难。为了满足存储容量限制,有:

[0122] $$\left(\sum_{i=1}^{\#Conv+1}\left(N_i^2 \times c_i\right) + (k \times s)^2\right) \times \mathcal{WL} \times 2 < \mathcal{C} \qquad (式\ 7)$$

[0123] 3)缩放性能约束。通过使用相应数据求解联立以上三个方程和不等式,可以得到2≤k≤57。

[0124] 在赛灵思Xilinx ZC706评估板上测试上述系统,XC7Z045 FFG900-2 AP SoC具有350个逻辑单元,19.1Mb Block RAM,900个DSP片,360个最大I/O引脚和16个最 大收发器数。将其工作频率设置为100MHz,并使用16位固定数据类型。设计由赛灵 思SDSoC开发环境v2016.3实现。在具有20核英特尔至强CPU E52630 v3@2.30GHz 和64GB主内存的服务器上运行设计流程。使用来自SJTU Media Lab的超高分辨率4K 视频序列,其是YUV 4:2:0颜色采样,每个样本8位,帧率为30fps。原始的4K图 像被用作地面实况,2K LR图像通过下采样获得。本专利实施例提出的超分辨率系统可 生成重建的4K HR图像。

[0125] 为了评估系统性能,使用峰值信噪比(PSNR)和结构相似度(SSIM),这两者都是 定量评估图像分辨率质量的广泛使用的指标。这些指标测量重建的HR图像与相应的基 本事实之间的差异。使用以下公式计算PSNR:

[0126] $$PSNR = 10\log_{10}\frac{R^2}{MSE}$$

[0127] 其中R是输入图像数据类型中的最大波动。例如,当图像使用8位无符号整数数据类型进行编码,因此R为255.MSE表示均方根误差,其计算公式如下:

[0128] $$MSE = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(I_1(i,j) - I_2(i,j)\right)^2$$

[0129] 其中H和W是输入图像的高度和宽度,并且I1(i,j)和I2(i,j)是两个图像的对应像素 值。SSIM质量评估指标基于三个参数的计算,即亮度参数,对比参数和结构参数。总 体指标是三者的乘积:

[0130] $SSIM(X,Y) = [l(X,Y)]^{\alpha} \cdot [c(X,Y)]^{\beta} \cdot [s(X,Y)]^{\gamma}$

[0131] 其中:

[0132] $$l(X,Y) = \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1}$$

[0133] $$c(X,Y) = \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2}$$

[0134] $$s(X,Y) = \frac{\sigma_{XY} + C_3}{\sigma_X\sigma_Y + C_3}$$

[0135] 其中,μ_X,μ_Y,σ_X,σ_Y,和σ_{XY}分别是图像X和Y的局部均值,标准差,协方差。对于其他常量,通常设定$\alpha=\beta=\gamma=1$,$C_1=(K_1\times L)^2$,$C_2=(K_2\times L)^2$,$C_3=C_2/2$,其 中$K_1=0.01$,$K_2=0.03$,$L=255$。值得注意的是,人眼对亮度信息最为敏感,因此只在 YCbCr图像中处理和测量强度通道。

[0136] 不同TV阈值和块大小的性能之间的关系需要通过实验得到。这两个因素非常关键，因为不同的阈值会改变每个处理模块的工作负载，从而影响性能(速度和质量)。同时，块大小决定了实现的资源利用率。实验可以帮助进一步确定FPGA系统实施中的设计参 数，可以分为以下几种但不限于：

[0137] 1)不同子图中的TV值之间差别很大，并且可能与原始图像的视觉属性有关。实验发现TV分布遵循瑞利分布。在本实施例中，选择50作为基准值，根据统计数据，其中 比例为25.3％。

[0138] 2)具有相同块大小的不同TV阈值：在这组实验中，选择30作为块大小并且将TV 值阈值从30增加到70，步幅为10。测试每个块的平均值以评估性能，当阈值越高时， 性能越优。显然，当选择更高的阈值时，更多的块将用神经网络处理，这通常导致更好 的结果。

[0139] 3)与对应的TV值阈值不同的块大小：在这组实验中，将块大小从10增加到50， 步幅为10，并根据块区域设置相应的阈值。使用30的块大小和50的TV阈值作为对照 组。结果，选择精细的块可以获得更高的重建质量。但是，这是以更高的计算复杂度为 代价的。

[0140] 4)总体比较：在这组实验中，对比了表4中不同配置的六种解决方案。考虑预处理方法(分块/无)和放大方法(神经网络/插值)，测试了所有四种可能的组合。第五和第 六种解决方案都使用分块和混合放大方法，其中25.3％的块通过根据神经网络放大，其 余块通过插值放大。作为对照，第五个解决方案为每个块随机选择两种方法，而第六个 解决方案使用总变化门限进行调度。图6显示了六种配置的示例输出。

[0141] 表4六种不同方式比较

[0142]

编号	处理	提升方式	#Mult.	PSNR(dB)	SSIM
1	无	插值	$6.6*10^7$	35.51	0.9138
2	无	神经网络	$8.2*10^9$	38.55	0.9421
3	调度	插值	$6.6*10^7$	35.51	0.9138
4	调度	神经网络	$8.4*10^9$	38.55	0.9420
5	调度	随机混合	$2.2*10^9$	36.10	0.9211
6	调度	根据 TV 混合	$2.2*10^9$	37.36	0.9287

[0143] 根据结果可以得出如下结论：

[0144] 1)神经网络比插值算法明显显示出更好的质量(+3.04dB)，运算代价则高出两倍数量级。

[0145] 2)使用适当的填充后，将图像裁剪成小块子图可以达到与不裁剪几乎相同的质量。

[0146] 3)根据TV阈值调度块比随机调度更好(+1.26dB)。

[0147] 4)与全神经网络方法相比，混合方法节省了约75％的乘法成本，质量降低在可接受的范围内(-1.19dB)。

[0148] 对于全高清1920×1080输入到超高清3940×2160输出的超分辨，本发明实施例中的系统可以达到的平均帧速率为23.9fps，29.3fps和31.7fps，分别对应使用了一个，两 个和三个插值器。

[0149] 上述实施例提供的技术方案,可以实时对较低分辨率的媒体提升分辨率,具有如下 优点:

[0150] 1)将准确但复杂的神经网络与快速但简单的插值算法相结合,可以为大尺寸输入产 生高速和高质量的输出。

[0151] 2)提出了一个用于分析和优化的定量模型,以平衡有限硬件资源的利用率,可获得 的帧速率和视觉性能。

[0152] 3)本专利实施例提出的超分辨率系统产生比现有文献报道的更高分辨率的视频,即 在嵌入式FPGA板上以大约30fps的帧速率从1920×1080FHD源产生的3940×2160 UHD视频。

[0153] 显然,本领域的技术人员应该明白,上述的本发明的FPGA系统设置方式可以用Verilog,C,C++等语言来实现;神经网络、插值模块均可以采用不同的原理与类型;调 度模块可以用片内或片外处理来实现,各模块可以集中在单个的计算装置上,或者分布 在多个计算装置所组成的网络上,可选地,它们可以用计算装置可执行的程序代码来实 现,从而可以将它们存储在存储装置中由计算装置来执行,或者将它们中的某个或多个 模块或步骤制作成单个集成电路模块来实现。这样,本发明不限制于任何特定的硬件和 软件结合。

[0154] 以上所述仅为本发明的优选实施例而已,并不用于限制本发明,对于本领域的技术 人员来说,本发明可以有各种更改和变化。凡在本发明的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

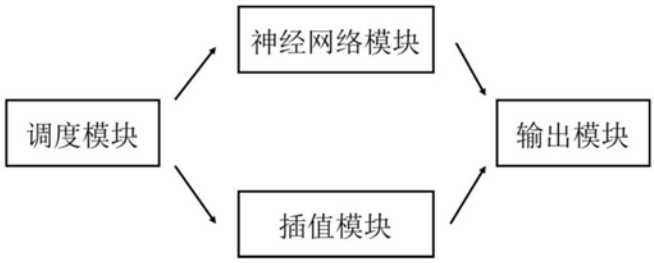

神经网络模块
调度模块
输出模块
插值模块

图1

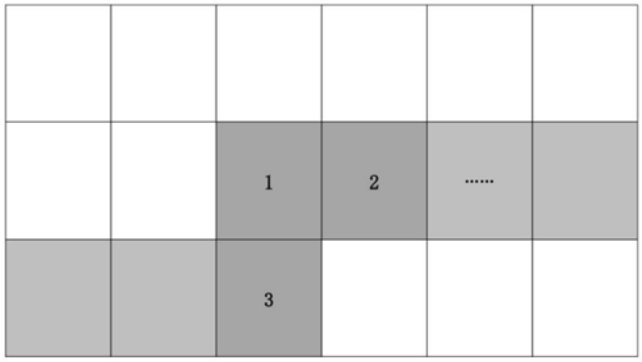

1
2
……
3

图2

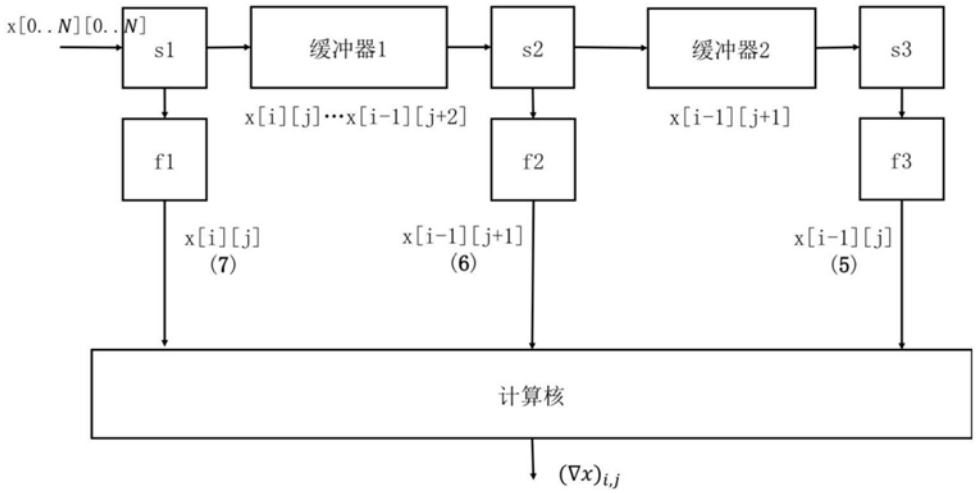

x[0..N][0..N]
s1
缓冲器1
s2
缓冲器2
s3
x[i][j]…x[i-1][j+2]
x[i-1][j+1]
f1
f2
f3
x[i][j]
(7)
x[i-1][j+1]
(6)
x[i-1][j]
(5)
计算核
$(\nabla x)_{i,j}$

图3

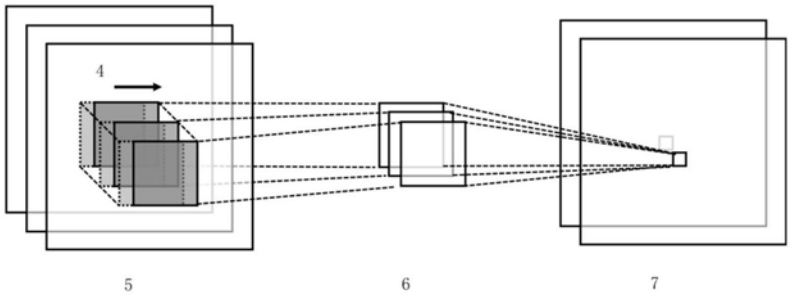


图4

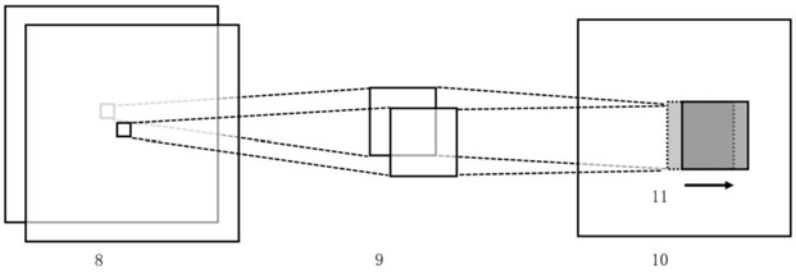


图5

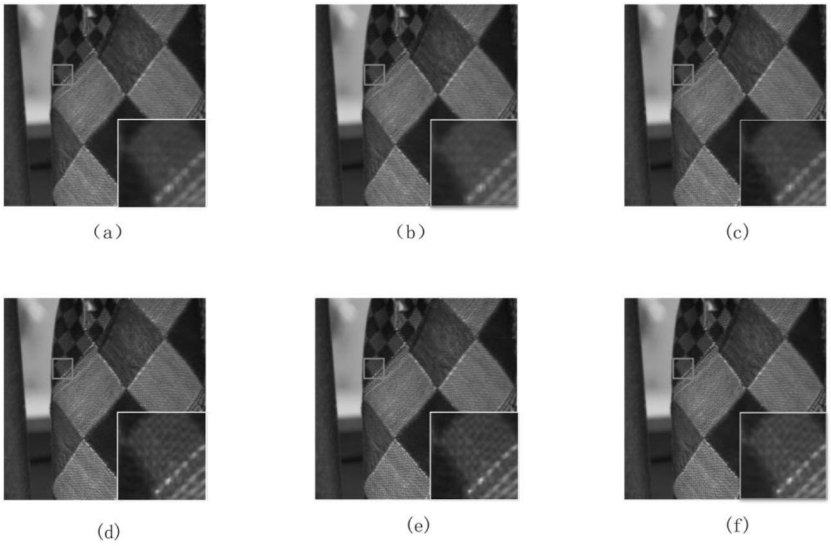


图6