基于GPU的图片特征提取与检测

徐 晶 曾苗祥 许 炜
(华中科技大学电子与信息工程系湖北省智能互联网技术重点实验室 武汉 430074)

摘 要 针对目前高速网络中图像数目多、分辨率大、普通PC机上的检测方法对图片检测达不到实时性的问题,提出了一种基于GPU的快速图片检测方案。该方案分别采用SURF(Speed-Up Robust Features,加速鲁棒特征)算法和SVM(Support Vector Machine,支持向量机)算法对图像进行特征提取和特征分类,并利用GPU浮点运算的并行性来优化系统。最后实验证实,相对于普通PC机上实现的方案,使用GPU的检测速度提升5到9倍。

关键词 加速鲁棒特征,支持向量机,GPU
中图法分类号 TP391.4 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.07.032

GPU Based Image Feature Extraction and Detection

XU Jing ZENG Miao-xiang XU Wei
(Internet Technology and Engineering R&D Center, Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract As the performance of image detection method in the normal PC is not satisfactory nowadays because of the huge amount of high resolution image transmitting in current high-speed network, this paper presented a GPU-based solution for image extraction detection. We obtained feature extraction by using SURF(Speed Up Robust Features) algorithm and classified image features based on SVM(Support Vector Machine) algorithm. Furthermore, we utilized the parallelism of GPU float point arithmetic to optimize the system. As a result, the performance of GPU-based system achieves 5 to 9 times faster than the CPU-based system.

Keywords Speed up robust feature, Support vector machine, GPU

1 引言

目前,互联网给人们带来了越来越便捷舒适的生活,但同时不健康的内容也得以迅速传播,其中网络中存在大量含有不良信息的图片,这给人们尤其是青少年的身心健康发展带来了很大的负面影响,因此加强对互联网上的不良图片的检测和监控十分重要。

现阶段对互联网上的图片进行检测的流程是先还原经过网络端口的图像数据,再对图片进行不良信息检测,如果检测为不良图片,则进行后续的处理。本文重点研究的是快速的不良图片检测方案。目前图片检测普遍采用的方法是肤色点检测,在HSV颜色空间上,肤色点的像素主要集中在一个范围内,可以通过像素点的色度信息判别是否为肤色点。研究证明肤色检测能够检测不良图像[1],但是在准确率方面略显不足。为了提升不良图像检测的准确率,Wayne Kelly等人在肤色点检测的基础上引入了图像特征提取与特征分类的方法[2]。David G. Lowe在2004年提出了SIFT(Scale Invariant Feature Transform,尺度不变特征变换)算法,它能准确地提取图像局部特征。接着在2006年,Herbert Bay首次提出了SURF算法,该算法由SIFT算法改进而来,一般来说,标准的SURF算法比SIFT算法快3到5倍,并且在多幅图像下有更好的鲁棒性[3]。文献[4]中提出了对图片特征进行分类的SVM算法,提升了不良图像检测的准确率。

随着越来越多的高分辨率图像文件在网络中传播,目前SURF和SVM算法处理图片的速度已无法满足实时性的需要(本文后续章节将详细说明),因而对图片检测提出了新的挑战。如何加快SURF算法的执行成为系统性能提升的关键。图形处理单元(Graphic Processing Unit,GPU)的出现给图像不良内容检测的性能提升带来了契机,GPU在并行浮点计算上的优势已经越来越多地被认可并加以运用[5]。据此,本文提出了将SURF算法移植到GPU中的思路,并加以实现,以达到实时检测的需求。

本文后续章节中将首先介绍图像检测的系统框架,然后重点说明基于GPU的SURF算法实现方案,最后通过实验数据验证本方案的有效性。实验结果表明本方案能够满足图像快速检测的需要。

2 基于GPU的图像检测框架设计

2.1 现有方法介绍

目前,不良图像检测采用的是SURF算法和SVM算法。其中,SURF算法用于提取图像的局部特征,对图像的旋转、尺度缩放、亮度变化保持不变性[6];SVM算法是一种可训练

到稿日期:2013-09-12 返修日期:2014-01-14 本文受"十二五"国家科技支撑计划课题(2011BAK08B02)资助。
徐 晶(1979—),男,博士,讲师,主要研究方向为无线网络通信、网络安全,E-mail:xujing@mail.hust.edu.cn;曾苗祥(1991—),男,硕士,主要研究方向为图像处理、视频编解码;许 炜(1977—),男,博士,副教授,主要研究方向为网络内容管理。

的机器学习方法，依靠小样本学习后的参考模型对未来输入进行正确预测，在解决小样本、非线性及高维模式识别问题中表现出许多特有的优势[4]。

系统检测流程如图 1 所示。

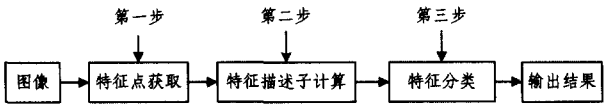

图 1　图像检测流程图

图像检测主要由特征点获取、特征描述子计算和特征分类 3 个步骤组成。通过测量 3 个步骤的实际耗时，可以发现前两个步骤的耗时很大，约为整张图像检测耗时的 99%。表 1 为在 Centos 5.9 64 位操作系统、主频为 3.3GHz 的 Intel i3-2120 CPU 平台下不同分辨率图像检测过程中 3 个步骤的实测耗时结果，单位为秒。

表 1　CPU 中不同分辨率的图像检测 3 个步骤平均耗时

	第 1 步	第 2 步	第 3 步
480 * 320	62.9	60	0.15
640 * 480	106	102	0.15
1280 * 720	274	251	0.15
1920 * 1080	717	603	0.15

由表 1 可知，第 1 步和第 2 步是图片检测性能的瓶颈，因此本文主要对获取特征点和图像特征描述子的算法进行加速。

目前 GPU 的内核数量和存储带宽的急速增长使得它非常适合于大规模的并行计算[7]。在图像中的各个像素或像素区域的计算在 SURF 算法中独立并行，可以使用 GPU 计算图像特征点以提升算法的速度。接下来介绍基于 GPU 的系统实现框架。

2.2　基于 GPU 的特征提取框架

在计算机系统中，CPU 主要用于算法逻辑与控制，但计算速度慢，在 CPU 中计算图像特征点和特征描述子会影响系统的效率。GPU 精于计算，尤其是大规模浮点计算。图像处理中各个像素之间存在着独立无关特性，因此可以利用 GPU 对图像进行并行计算，从而提升系统执行速度。

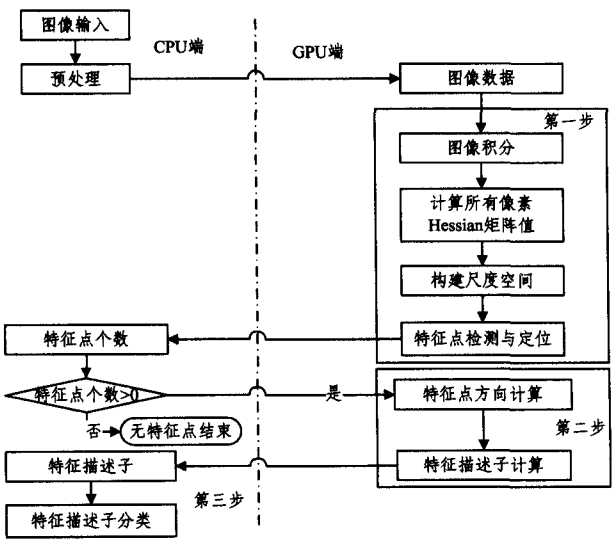

图 2　基于 GPU 的图像检测系统实现框架

本文利用 GPU 在并行浮点计算和内存管理等方面的优势，合理分配 GPU 的资源以及在计算 SURF 特征描述子过程中所承担的角色。基于 GPU 的不良图像检测系统实现框架如图 2 所示。

算法的第一步为图像积分、计算所有像素点 Hessian 矩阵的值、构建尺度空间、特征点检测与定位；第二步为特征点方向计算、特征描述子计算。使用 GPU 计算这两步能显著地提升算法性能。在第三步使用 SVM 算法对特征描述子分类。表 1 的结果显示在图像检测的过程中，SVM 对图像特征的预测分类耗时很少，因此在系统中采用普通方法即可。目前 SVM 方法已经非常成熟，有大量文献对算法进行了描述，本文不再赘述，此部分内容可参考文献[4]。

接下来本文将对基于 GPU 的图像 SURF 特征提取步骤进行重点描述。

3　基于 GPU 的图像特征描述子计算的设计

3.1　图像积分

3.1.1　算法概述

对图像积分可以提升高框状卷积滤波器的计算效率。对于积分图像中的某点 $X(x,y)$，该点的值表示原始图像中原点和点 X 形成的矩形区域里所有像素之和。

$$I_\Sigma(X)=\sum_{i=0}^{i\leqslant x}\sum_{j=0}^{j\leqslant y}I(i,j)$$

将原始图像转换为积分图像之后，计算一个图像区域内的像素点的灰度值之和就可以使用加减运算来解决，如图 3 所示，$S=I_\Sigma(A)-I_\Sigma(B)-I_\Sigma(C)+I_\Sigma(D)$。

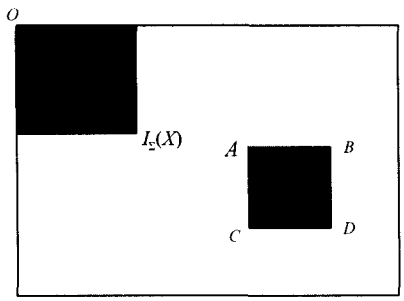

图 3　区域像素积分示意图

3.1.2　GPU 实现的策略

计算积分图像需要对图像中的所有像素的灰度值求和，这些像素的灰度值在 SURF 算法中是相互独立的，可以并行计算，因此本文使用 GPU 对图像积分。

首先将图片的像素表示为子集形式，即每一个子集有 m 像素值，所有子集的并集为整张图片的像素，且子集之间不含重复像素。这样所有像素点的灰度值保存在 $sw[t][m]$（t 为子集的个数）数组中并导入 GPU。接着在 GPU 中进行图像积分运算。

在 GPU 中的每个线程处理 m 个像素的灰度值。GPU 中执行的线程的个数为 t。等待每个线程执行结束之后，再将所有线程的灰度值相加，即为图像的积分图像。实现的算法如下：

1. 获取线程的索引 i 的值；
2. 获取处理图片线程的个数 t；
3. 初始化每个线程的灰度值计算结果 sum[j]；
4. if(i<t)
5. 每个线程计算对应像素子集的和 sum[i]；
6. _syncthreads()；//等待所有线程执行结束，以同步线程；
7. 计算所有 sum[i]的和 sum_total。

sum_total 即为所求的像素积分。

计算某一区域的图像积分可以通过将上述结果进行 3 次

加减运算实现。

3.2 Hessian 矩阵行列式值计算

3.2.1 算法概述

在 SURF 算法中，兴趣点的检测基于尺度空间理论，尺度空间的建立采用的方法是不断增大框滤波模板[6]。SURF 将尺度空间分为若干组(Octaves)，一个组代表了逐步放大的滤波模板对同一个输入图像进行滤波的一系列响应图像。每一组由若干固定的层(Scales)组成。组和层的大小关系如图 4 所示(图中的数字表示正方形滤波器框的边长)。

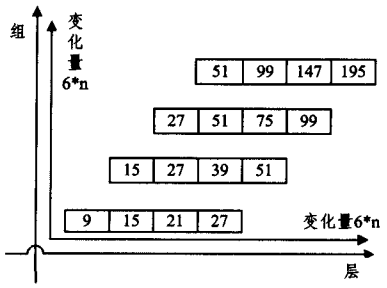

图 4 尺度空间大小变化图

SURF 依靠 Hessian 矩阵行列式的局部最大值定位兴趣点实现对兴趣点的定位[3]。图像中的某点 $X(x,y)$在尺度空间 σ 上的 Hessian 矩阵定义为：

$$H(X,\sigma)=\begin{vmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{vmatrix}$$

$L_{xx}(X,\sigma)$表示在 X 处的高斯二阶偏导与图像的卷积，即 $\partial^2 g(\sigma)/\partial x^2$、$L_{xy}(X,\sigma)$、$L_{yy}(X,\sigma)$具有相似的含义。

算法使用框滤波器近似地求取高斯二阶偏导，通过不同尺寸框滤波模板和积分图像求取 Hessian 矩阵行列式的响应图像。

3.2.2 GPU 实现的策略

尺度空间的计算函数返回某一组某一层的滤波器框的边长，将其放在 GPU 中执行可以提升性能。

在算法中需要计算所有尺度空间中每个像素点的 Hessian 矩阵行列式值，而每个像素点的 Hessian 矩阵计算都是独立的，与其他像素的值无关，因此将其移植到 GPU 中。

实现算法如下：

```
1. 计算线程索引；
2. 计算图像尺度空间；
3. 计算尺度空间滤波器大小；
4. if(滤波器框边长小于图像长和宽)//
5. {
6.    计算尺度空间下图像的点在 x 方向的二阶偏导 dxx；
7.    计算尺度空间下图像的点在 y 方向的二阶偏导 dyy；
8.    计算尺度空间下图像的点在 x 方向和 y 方向的偏导 dxy；
9.    矩阵行列式=dxx * dyy-w * w * dxy * dxy;//w 为权重系数，算
      法中取值为 0.9；
10. }
```

在算法中像素在 GPU 分配的线程中执行，充分利用了 GPU 线程间的并行特性。

3.3 特征点获取

3.3.1 算法概述

为了准确地得到特征点坐标，算法构建了三维尺度空间 (x,y,σ)。在每个 3 * 3 * 3 的局部区域里，进行非最大值抑制，如图 5 所示。

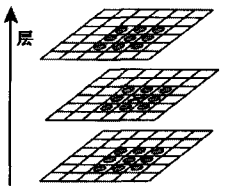

图 5 三维空间示意图

在图 5 的三维空间里，中间像素点的灰度值大于预设的 Hessian 阈值 H 且比临近的 26 个点(同一层的 8 个点和相邻两层的各 9 个点)的响应值都大的点才被定为特征点[3]。

3.3.2 GPU 实现策略

为了得到图像的所有特征点，需对图像中的每个像素进行计算。像素点是否为特征点仅与相邻层次的 9 个像素点和同一层次的 8 个像素点有关。每个像素点是否为特征点在计算上是独立的，本文利用 GPU 大量线程的并行性计算这一步骤。

将每个像素灰度以及其相邻的 26 个像素灰度组成一个数组传递到 GPU 中，每个线程对几个像素点是否为兴趣点做出判断，在线程中将该像素与临近的 26 个像素的灰度值和 Hessian 阈值相比，如果大于临近的 26 个像素灰度值和阈值，则该像素点为特征点。

在 GPU 端的算法如下：

```
1. 获取像素坐标信息；
2. 如果像素点在图像范围内
3. {
4.    获取比较的点的灰度值；
5.    该灰度值与其邻域 26 个像素比较；
6.    如果为极大值点，即比邻域 26 个像素灰度值都大
7.    {
8.       定位兴趣点坐标信息；
9.       计算兴趣点特征尺度信息；
10.   }
11. }
```

3.4 特征点方向计算

3.4.1 算法概述

为了使特征描述算子具有旋转不变的性能，算法为每个特征点赋予了方向。在算法中以兴趣点为圆心、以 $6S$(S 为兴趣点对应的尺度)为半径的圆形邻域里，用尺寸为 $4S$ 的 Haar 小波模板对图像进行处理，即得到了该邻域内每个点所对应的 x,y 方向的响应，然后用以兴趣点为中心的高斯函数($\sigma=2S$)对这些响应进行加权。接着使用圆心角为 60 度的扇形以特征点为中心旋转一周，计算该扇形处于每个角度时，它所包括的图像点的 Haar 小波响应和。把扇形区域环绕一周所形成的矢量都记录下来，取长度最大的矢量，其方向即为该特征点所对应的方向[3]。

3.4.2 GPU 实现策略

在算法中先计算得到特征点的尺度因子，再计算高斯尺度大小，接着得到特征点的矢量长度最大的方向。使用 GPU 可以将每个特征点的计算置于一个线程中执行。算法流程如下：

```
1. 计算尺度因子 s；
2. 计算尺度滤波器大小 z；
3. 如果图像的长宽不在尺度滤波器范围内
4.    不合理，退出；
5. 计算线程索引 i；
6. 如果 i<并行的线程个数
```

```
7. {
8.   计算特征点所有方向的 X 响应;
9.   计算特征点所有方向的 Y 响应;
10.  得到计算能使特征矢量长度最大的 X 和 Y;
11.  计算特征点方向角度;
12.  如果计算出的角度小于 0
13.  将角度增加 360 使其为正值;
14. }
15. 将特征点坐标、方向保存。
```

在得到特征点方向后,就可以进行特征点描述子的计算。

3.5 特征描述子计算

3.5.1 算法概述

为构建特征点描述子向量,先确定一个以特征点为中心的正方形邻域。把该邻域分成 4 * 4 个子块区域,对每个子块区域的 dx、dy、$|dx|$、$|dy|$ 求和(dx 表示水平方向的 Haar 小波响应,dy 表示竖直方向的 Haar 小波响应),得到一个 4 维向量 $V(\Sigma dx, \Sigma dy, \Sigma|dx|, \Sigma|dy|)$,把 4 * 4 个子块的向量连接起来即得到一个 64 维向量,此向量就是描述该特征点的描述子特征向量[3]。

3.5.2 GPU 实现策略

和特征点方向的计算一样,使用 GPU 可以将每个特征点的计算置于一个线程中执行。因此将特征描述子的计算移植到 GPU 中能提升算法的性能。

GPU 中的实现算法如下:

```
1. 获取线程索引 i;
2. 如果 i<16//只有 4 * 4 共 16 个子块
3. {
4.   计算特征点附近的 Σdx 值;
5.   计算特征点附近的 Σdy 值;
6.   计算特征点附近的 Σ|dx| 值;
7.   计算特征点附近的 Σ|dy| 值;
8. }
```

4 * 4 共 16 个子块的 $\{\Sigma dx, \Sigma dy, \Sigma|dx|, \Sigma|dy|\}$ 组成的 64 维向量即为特征点描述子。

得到特征点描述子后,将其模长标准化,才能使用 SVM 进行分类。本方案中特征描述子的标准化计算也是在 GPU 中进行的。在 GPU 中实现的过程如下所示:

```
1. 获取原始特征描述子向量;
2. 计算特征描述子模长的平方;
3. 计算特征点描述子模长;
4. 将描述子标准化。
```

在计算得到标准化的特征描述子之后,就完成了 GPU 上的操作,接着将生成的特征描述子通过 PCIE 传到 CPU 上,使用 SVM 训练得到的支持向量对特征描述子进行分类。

4 实验与结果

4.1 实验方案

本文设计实验测试使用 GPU 获取的图像特征描述子的准确性和实时性。

实验平台的硬件部分包括 Intel(R) Core(TM) i3-2120 型主频为 3.3GHz 的 CPU,和 NVIDIA Geforce GTS 450 型、显存 1GB、显存位宽 128bits、总线接口 PCI Express 2.0 16X、有 192 个 1.242GHz 流处理器内核的 GPU;软件环境为 Centos 5.9 64 位操作系统,Opencv2.4.0 源码安装包。在实验中采用 CPU 上运行的普通方法作为对照。使用二者对相同的图像提取的特征点描述子在经过训练后的 SVM 分类器中分类,根据二者的不同输入得到的分类结果对二者进行比较。

实验 1 测试使用 GPU 实现的算法的准确性。在实验中采用正常图像集和不良图像集各 2000 张,记录采用 GPU 的方案和不采用 GPU 的方案的实验结果。检测的流程已在图 1 中介绍。

实验 2 测试使用 GPU 实现的算法的性能提升倍数。在实验中采用的图像集是不同格式、不同分辨率的正常图像和不良图像。采用的分辨率主要有 1920 * 1080、1280 * 720、640 * 480、480 * 320,图片格式主要是.jpg 这种网络上主流的图像格式。

4.2 准确性实验结果

实验 1 中,测试使用 GPU 和使用 CPU 两种方案对两组图片检测的结果。记录每组图片在两种实现情况下的检测情况。

实验结果如表 2 所列。

表 2 GPU 方法和 CPU 方法的检测结果

	GPU 方法		普通方法	
	不良数目	总图片数	不良数目	总图片数
正常图片	0	2000	0	2000
不良图片	1991	2000	1991	2000

结果显示,GPU 和普通方法实现的 SURF 算法获得的图像特征描述子分类结果完全相同,因此本文使用的 GPU 实现的方案在不良图像检测上是可行的。在检测 2000 张不良图片时都有 9 张漏检情况,主要原因是 SVM 在训练时样本不够全,造成了一些漏检,增加训练集的图片数目可有效地改善这个问题。

4.3 性能提升测试

实验 2 测试使用 GPU 的方案相对于 CPU 的性能提升,分别对正常人物图像和不良图像进行了测试。表 3 列出了使用 GPU 加速和普通方法对于不同分辨率的正常人物图片的检测结果(注:每组测试的图片集为 2000 张,表中的时间为每张图片检测的平均耗时,时间单位为毫秒)。

表 3 正常人物图片性能检测结果

图像分辨率	执行时间/ms	
	CPU 版本	GPU 版本
480 * 320	130	25
640 * 480	238	37
1280 * 720	676	88
1920 * 1080	1400	158

表 4 列出了使用 GPU 加速和普通方法对于不同分辨率的不良信息图片的检测情况。

表 4 不良人物图片性能检测结果

图像分辨率	执行时间/ms	
	CPU 版本	GPU 版本
480 * 320	122	23
640 * 480	208	30
1280 * 720	507	65
1920 * 1080	1318	140

从表 3、表 4 的结果显示:在图像分辨率相同时,正常图像和不良图像的检测时间几乎一样。所以本文提出的方案对于所有图像检测均能提升性能。

在检测图像分辨率为 480 * 320 的不良图片时,使用 GPU 的方案检测速度为不使用 GPU 的方案的 5.3 倍(122/23,下面的计算方法相同),提升了 4.3(5.3−1)倍,当图像分

辨率为 640 * 480 时，速度提升了 5.9 倍，当图像分辨率为 1280 * 720 时，速度提升了 6.8 倍，当图像分辨率为 1920 * 1080 时，速度提升了 8.4 倍。随着图片分辨率的增加，使用 GPU 加速的性能提升倍数也逐渐增大，这是因为更加充分地利用了 GPU 各个线程之间在运算方面的并行性。

表 2－表 4 表明，使用 GPU 对图像进行 SURF 特征提取在保证准确率相同的情况下实现了对算法性能的提升。在图像检测上，基本达到实时性的效果。

结束语 本文设计了一个具备高检测准确率的网络不良图像的实时处理方案，采用 SURF 算法和 SVM 算法对图像进行不良内容检测，利用 GPU 各个线程浮点运算的并行性实现 SURF 算法，性能有了大幅提升，在保证准确率的同时也满足了在实时性上的需要。实验结果验证了本文提出的方案的有效性。在后续的工作中我们将进一步研究算法在性能上提升的可能性，研究更高性能的 GPU 对图像检测性能的提升。同时研究快速的不良视频检测方案，以获得更全面的网络不良内容的实时监控系统，为保证青少年身心健康发展做出一定的贡献。

参 考 文 献

[1] Gang Zhao, Wang Shu-hui, Wang Tai, et al. HSV Color Space and Face Detection Based Objectionable Image Detecting[C]// IEEE FGCNS'08. 2008:107-110

[2] Kelly W, Donnellan A, Molloy D. Screening for Objectionable Images: A Review of Skin Detection Techniques[C]// IEEE IMVIP. 2008:151-158

[3] Bay H, Ess A, Tuytelaars T, et al. SURF: Speed Up Robust Features[C]// IEEE 2008. 2008:346-359

[4] Chapelle O, Haffner P, Vapnik V N. Support vector machines for histogram-based image classification[J]. IEEE Transactions on Neural Networks, 1999, 10(5):1055-1064

[5] Balatsos A, Aleksic M. A bridge for a multiprocessor graphic system[C]// IEEE CCECE 2002. 2002:646-650

[6] 林晓帆，林立文，邓涛. 基于 SURF 描述子的遥感影像配准[J]. 计算机工程，2010，6(12)：216-218

[7] 王志国，王贵锦，施陈博. 积分图像的快速 GPU 计算[J]. 计算机应用研究，2011(10)：3913-3916

(上接第 139 页)

综合上述验证得出结论，微移动并发属性不存在活锁、死锁，并且绑定信息满足活性要求，满足协议的功能属性和安全性。同理，可验证 HMIPv6 协议模型的宏移动并发属性满足协议功能属性、安全性和活性。

3.4 上述方法的评价

化简抽象前后模型的对比如表 1 所列。

表 1 原模型与新模型对比

	宏移动并发属性	微移动并发属性
原/新变迁比	28/16	20/12
原状态数	421	221
新状态数	157	93
原弧数	718	360
新弧数	254	142

从表中可知，对于宏移动并发属性，新模型的发生序列长度仅为原模型的 57.1%，而状态数仅为原模型的 37.3%，弧数仅为原模型的 35.4%。对于微移动并发属性，新模型的发生序列长度仅为原模型的 60.0%，而状态数仅为原模型的 42%，弧数仅为原模型的 39.4%。由上述数据可知，相对于原模型，新模型减少的点火变迁数目越多，新生成的模型的状态空间越小。

综上所述，本文提出的化简和抽象方法能够在保证并发属性的前提下，有效减小验证的复杂程度，对状态空间爆炸问题具有一定的抑制作用，所以能够对较为复杂的并发系统进行验证。

结束语 本文针对并发模型的状态爆炸问题，提出了基于并发属性的模型化简方法和基于功能组合的模型抽象方法，化简方法移去了与并发属性不相关的模型元素，抽象方法提升了模型的抽象层次，化简抽象后模型的状态空间规模显著降低，并且在该并发属性相关行为上与原系统模型保持一致。在化简抽象后模型中运用状态空间分析、模型检测等验证方法得出模型的错误，通过两种模型的对照关系，改正原系统模型中的错误。上述方法有效避免了状态空间爆炸问题，完成了复杂的并发模型验证工作。通过将上述方法应用于 HMIPv6 协议模型，验证了方法的有效性。

下一步，将进一步开发工具软件，实现本文提出的方法，提升模型验证的效率。

参 考 文 献

[1] Dalal S R, Jain A, Karunanithi N, et al. Model-Based Testing in Practice[C]// Proc. of the 21st International Conference on Software Engineering. 1999:285-294

[2] Yan J, Wang J, Chen H W. Survey of Model-based Software Testing[J]. Computer Science, 2004, 31(2):184-187

[3] Utting M. The Role of Model-Based Testing[C]// Verified Software: Theories, Tools, Experiments, LNCS 4171. 2008:510-517

[4] Farooq U, Lam C P, Li H. Towards Automated Test Sequence Generation[C]// Proc. of the 19th Australian Conference on Software Engineering. 2008:441-450

[5] Broy M, Jonsson B, Katoen J P, et al. Model-Based Testing of Reactive Systems[C]// LNCS 3472. Springer, Heidelberg, 2005

[6] Sun T, Ye X, Liu J, et al. CPN based protocol testing sequence generating method[J]. Journal of PLA University of Science and Technology, 2012, 13(2):165-170

[7] Constant C, Jeron T, Marchand H, et al. Integrating Formal Verification and Conformance Testing for Reactive Systems[J]. IEEE Transaction on Software Engineering, 2007, 33(8):558-574

[8] Zeng Y, Zhang L, Zhang Y, et al. Activity Diagram-based Method to Generate Test Sequence of Concurrent Software[J]. Computer Science, 2007, 34(12):286-290

[9] Buchs D, Lucio L, Chen A. Model Checking Techniques for Test Genera-tion from Business Process Models[C]// Proc. of the 14th Ada-Europe International Conference. 2009:59-74