基于外沿三角形网格划分的凸壳并行化处理

黄 涛 李 燕 周启海 王 静

(西南财经大学信息技术应用研究所 成都 610074) (西南财经大学经济信息工程学院 成都 610074)

摘 要 实现复杂问题的并行化处理的最基本问题之一,是如何将复杂问题分割成若干个子问题。首先研究了凸壳的一些特殊几何性质,然后利用这些性质将所讨论的点集分割在一些网格中。同时论证了凸壳顶点只能位于这些网格中的外沿三角形网格中,并且在各外沿三角形网格中所求得的凸壳顶点彼此相互独立,从而为凸壳并行化处理的设计与实现带来了极大便利。

关键词 凸壳,并行处理,网格划分

中图法分类号 TP301.6 **文献标识码** A

Parallelized Processing for a Convex Hull Based on Division of Around Triangular Grid

HUANG Tao LI Yan ZHOU Qi-hai WANG Jing

(Research Institute of Information Technology Application, Southwestern University of Finance and Economics, Chengdu 610074, China)

(School of Economic Information Engineering, Southwestern University of Finance and Economics, Chengdu 610074, China)

Abstract Using the parallelization to deal with complex questions, a most basic issue is how to divide the complex question into some sub-questions. Firstly studied some special geometric property about convex shell, then used these property to divide the set of points into some triangle grids. Simultaneously proved convex shell umbo is only located in the periphery triangle grid and looked for convex shell umbo in these periphery triangle region mutually independently.

Keywords Convex shell, Parallelly processing, Grid division

1 引言

凸壳问题自20世纪70年代被提出以来,一直受到国内外众多专家学者的关注,相继构造出许多凸壳算法,如基于最大基线倾角智能逼近的凸壳新算法[1]、求平面点集凸壳的一个最优算法[2]、双域单向水平倾角最小化圈绕凸壳新算法[3]、基于凸多边形的凸壳算法[4]、平面点集凸壳的一种快速算法[5]、求解简单多边形和平面点集凸包的新算法[6],但这些算法都是串行算法。凸壳问题在最坏情况下的时间复杂度,其已知下限为 $O(n\log(n))$[7]。有些串行算法的时间复杂度已经达到此下限(如平面点集凸壳的一种快速算法[5])。因此,要进一步提高求解凸壳问题效率,必须从其它方面进行研究。近年来,并行计算获得了较快发展,这无疑给复杂问题的解决提供了一种有效的方法。要实现凸壳问题的并行处理,一个较为关键的问题是:如何将凸壳问题划分成若干个子问题。本文就这一问题进行了研究,并提出了凸壳的三角形网格划分方法。

2 并行算法设计过程及基本划分思想简介[8]

并行算法设计过程一般可以分为4个阶段:任务划分、通信分析、任务组合和处理机映射,如图1所示。

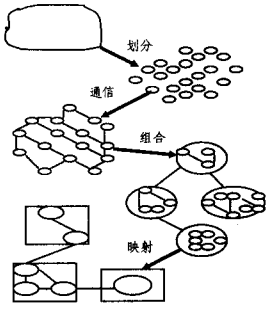

图1 并行算法设计过程

划分:将整个问题分解成一些子问题,其目的是尽量开拓并行执行的机会。

通信:确定诸任务执行中所需交换的数据和协调诸任务的执行,并由此可检测上述划分的合理性。

组合:按性能要求和实现代价来考察前两个阶段的结果,必要时可将一些子任务合并成更大的子任务,以提高性能或减小通信开销。

映射:将每个子任务分配到一个处理机中,在此过程中应尽量达到的目的是:最小化全局执行时间和通信成本以及最大化处理机的利用率。

到稿日期:2008-06-30

黄 涛(1972—),男,博士生,讲师,主要研究方向为计算机应用;**李 燕**(1983—),男,硕士研究生,主要研究方向为信息技术及管理;**周启海**(1947—),男,教授,博(硕)士生导师,主要研究方向为计算几何、算法研究与应用、财经计算、同构化信息处理等,E-mail:zhouqh@swufe.edu.cn;**王 静**(1983—),女,硕士研究生,主要研究方向为信息技术及管理。

通过对并行算法设计过程的分析可知，划分的优劣直接会影响到通信分析、任务组合和处理机映射这3步工作能否顺利完成且效果较好。因此，设计并行算法时的划分处理，是基础的也是关键的一步。

常用的划分思想，可简介如下。

均匀划分思想——处理问题分为两步：(1)将给定的问题分割成 P 个相互独立而规模相近的子问题；(2)用 P 台处理器分别并行求解各子问题。

分治划分思想——将一个大而复杂的问题分割成若干个特性相同的子问题分而治之。若所得到的子问题的规模仍然过大，可反复使用分治策略，直到诸子问题较容易求解时为止。并行分治法可分为3步：(1)将原问题划分成若干个规模相近的子问题；(2)同时递归地求解诸子问题；(3)归并各子问题的解为原问题的解。在本文中将主要用到分治划分思想。

3 对已有划分方法的评析

设 S 为二维平面内有限点集 $S=\{s_i(x_i,y_i)\mid 1\leqslant i\leqslant n<+\infty\}$ 中的点(其中 n 为 S 中所含点的个数)，$x(s_i)$ 是点 s_i 的 X 坐标值，$H(S)$ 是点集 S 的凸壳。因求解以最左、最右点连线为分界线的上凸壳 $UH(S)$ 与下凸壳 $LH(S)$ 的方法是同构的，故只需讨论上凸壳 $UH(S)$ 的解法即可。

首先，把包含在上凸壳所围成的区域中的点按 X 坐标值从小到大进行排序，不妨设为 $x(s_1)<x(s_2)<\cdots<x(s_n)$。求上凸壳时可采用分治技术，令

$S_1=\{s_1,s_2,\cdots,s_{[n/2]}\}$，$S_2=\{s_{[n/2]+1},s_{[n/2]+2},\cdots,s_n\}$

那么，可将 S_1，S_2 放入到2台处理机中处理。一旦凸壳 $H(S_1)$，$H(S_2)$ 均被求出后，只需求出 $H(S_1)$ 与 $H(S_2)$ 这两个凸壳的上公切线(如图2粗虚线所示)，即可求出其上凸壳 $UH(S)$。

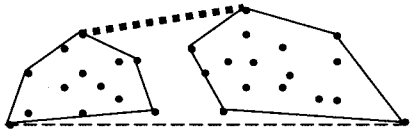

图2 上凸壳中两个子凸壳的上公切线

但是，此分割方法存在如下两个缺陷：(1)已知排序的时间复杂度为 $O(n\log(n))$[7]，而有些凸壳串行算法的时间复杂度已经达到凸壳问题在最坏情况下的时间下限 $O(n\log(n))$。故在分解原问题时进行的排序便成为该方法提高效率的瓶颈，从而使并行处理不能从根本上提高解决问题的效率。(2)若凸壳 $H(S_1)$，$H(S_2)$ 保留了 S 中绝大多数的点，那么求两凸壳上公切线的计算量较大。另一方面，若用此方法将点列 $x(s_1)<x(s_2)<\cdots<x(s_n)$ 分割成多个子部分，放入多台处理机中处理，随着子问题数目的增多，将使各子问题的解归并为原问题解的复杂性大增。而将子问题的解归并为原问题解的难易程度正是衡量划分优劣的一个重要标准。

因此，有必要研究和构造可根据实际情况需要而将原点集划分为多个规模相近子集的三角形网格划分方法，且它不会使各子问题的解的归并复杂性随着其子问题数目的增多而增大。

4 构造点集的三角形网格划分

为了叙述简便，本文假设二维点集 S 中的点不在同一条直线上(因为若在同一条直线上，则没有讨论的必要)。构造如下4个集合：

$A=\{a(x_1,y_1)\in S\mid \forall(x,y)\in S$ 有 $x\geqslant x_1\}$

$B=\{b(x_2,y_2)\in S\mid \forall(x,y)\in S$ 有 $y_2\leqslant y\}$

$C=\{c(x_3,y_3)\in S\mid \forall(x,y)\in S$ 有 $x_3\geqslant x\}$

$D=\{d(x_4,y_4)\in S\mid \forall(x,y)\in S$ 有 $y_4\geqslant y\}$

集合 A,B,C,D 中的点分别为通常所说的点集 S 中最左、最下、最右、最上的点。设 a_1，c_1 分别为集合 A，C 中 Y 坐标值最大的点；a_2，c_2 分别为集合 A，C 中 Y 坐标值最小的点。b_1，d_1 分别为集合 B，D 中 X 坐标值最大的点；b_2，d_2 分别为集合 B，D 中 X 坐标值最小的点，如图3所示。

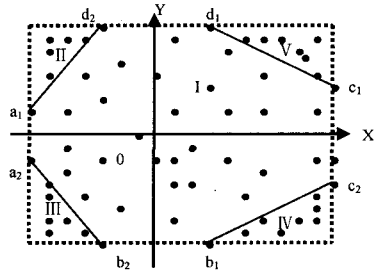

图3 三角形网格初始划分示意图

显然，a_1，c_1，a_2，c_2，b_1，d_1，b_2，d_2 均为凸壳的顶点，而中心区域Ⅰ中的点均不是凸壳的顶点。设点集 S 在区域Ⅱ，Ⅲ，Ⅳ，Ⅴ中的点所构成的集合为 S_1，S_2，S_3，S_4。称区域Ⅱ，Ⅲ，Ⅳ，Ⅴ为此划分阶段的外沿三角形网格；a_1，d_2 为与三角形网格Ⅱ相关联的凸壳顶点对，也称 a_1，d_2 为与点集 S_1 相关联的凸壳顶点对(简称 S_1 的关联顶点对)。对区域Ⅲ，Ⅳ，Ⅴ，可依此类推。

凸多边形的下列几何命题显然成立：

命题1 对于凸多边形任意一条边所在的直线而言，凸多边形所围成的区域在该直线的同一侧。

命题2 包含在 S_1 中的凸壳顶点不可能与包含在 S_2 中的凸壳顶点相连接构成凸壳的一条边，即在 S_1 与 S_2 中求凸壳的顶点是相互独立的。

可用命题1来证明如下命题：

证明：构造如下两个集合 $E=\{e\in S_1 \mid e$ 为凸壳的顶点$\}$，$F=\{f\in S_2 \mid f$ 为凸壳的顶点$\}$。

(1)若 E 与 F 至少有一个为空集，则命题2显然成立。

(2)若 E 与 F 均不为空集，假设存在两点 $e_1\in E$，$f_1\in F$，它们相连接构成凸壳的一条边。显然 a_1，c_1(因为 S 中的点不在同一条直线上，所以 a_1 与 c_1 不可能重合)在线段 e_1f_1 所在直线的两侧，这与命题1相矛盾。故假设不成立。

综上可知命题2成立。

同理可证 S_1，S_2，S_3，S_4 两两组合均有类似的结论，即在 S_1，S_2，S_3，S_4 中所求得的凸壳顶点彼此相互独立。

下面构造外围算法 AⅠ来求 S_1，S_2，S_3，S_4。

首先定义符号 $\max X(T)$ 表示点集 T 中 X 坐标值最大的点所组成的集合，依此类推可定义 $\max Y()$，$\min X()$，$\min Y()$。据此，可得如下算法。

输入：点集 S

输出：$(S_i,T(S_i),s_{i1},s_{i2})i=1,2,3,4$ (其中 $T(S_i)$ 为 S_i 中点的个数，s_{i1}，s_{i2} 为 S_i 的关联顶点对且 s_{i1} 的 X 坐标值小于 s_{i2}

的 X 坐标值)

Begin

Step1 求出:$a_1=\max Y(\min X(S))$,$a_2=\min Y(\min X(S))$,
$b_1=\max X(\min Y(S))$,$b_2=\min X(\min Y(S))$,
$c_1=\max Y(\max X(S))$,$c_2=\min Y(\max X(S))$,
$d_1=\max X(\max Y(S))$,$d_2=\min X(\max Y(S))$,
以及线段 a_1c_2 的中点 h_0(因为 S 中的点不在同一条直线上,所以 a_1 与 c_2 不可能重合)。

Step2 求出点 a_1 与 d_2 所在直线的方程 F_1、点 a_2 与 b_2 所在直线的方程 F_2、点 b_1 与 c_2 所在直线的方程 F_3、点 c_1 与 d_1 所在直线的方程 F_4。

Step3 对 S 中的每一个点(例如 s),分别求出 $F_i(s)$ 和 $F_i(h_0)$ $i=1,2,3,4$。若 $F_1(h_0)$ 与 $F_1(s)$ 异号,则将点 s 放入 S_1 中,并且 $T(S_1)$ 加 1;若 $F_2(h_0)$ 与 $F_2(s)$ 异号,则将点 s 放入 S_2 中,并且 $T(S_2)$ 加 1;若 $F_3(h_0)$ 与 $F_3(s)$ 异号,则将点 s 放入 S_3 中,并且 $T(S_3)$ 加 1;若 $F_4(h_0)$ 与 $F_4(s)$ 异号,则将点 s 放入 S_4 中,并且 $T(S_4)$ 加 1(从图 3 可以直观地看出 s 最多只能在 S_1,S_2,S_3,S_4 中的任何一个集合中);若 s 不在 S_1,S_2,S_3,S_4 中的任何一个集合中,则 s 必位于凸多边形 $a_1a_2b_2b_1c_2c_1d_1d_2$ 中(如图 3 所示),此时 s 为凸壳内点。令 a_1,d_2 对应为 s_{11},s_{12};a_2,b_2 对应为 s_{21},s_{22};b_1,c_2 对应为 s_{31},s_{32};d_1,c_1 对应为 s_{41},s_{42}。通过这一步的操作将有可能成为凸壳顶点的点分割到 4 个外沿三角形网格中。

End

可以将以上 4 个外沿三角形网格进一步分割。

设 $S_1\neq\Phi$,构造集合 $M=\{m\in S_1 \mid s\in S_1, m$ 到直线 a_1d_2 的距离大于等于 s 到直线 a_1d_2 的距离$\}$,因为开始假定 $S_1\neq\Phi$,所以 $M\neq\Phi$。设 m_1 为 M 中 X 坐标值最小的点,m_2 为 M 中 X 坐标值最大的点,连接 a_1m_1,m_1m_2,m_2d_2(如图 4 所示),则 S_1 中的点被分割在 S_{11},S_{12},S_{13} 3 个区域中,显然 S_{11} 中的点不可能是凸壳顶点。称 S_{12},S_{13},S_2,S_3,S_4 为此划分阶段的外沿三角形网格。

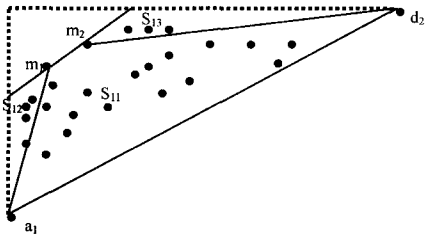

图 4

命题 3 m_1 与 m_2 均为凸壳的顶点。

证明:

(1)若 $M=S_1$,显然命题成立。

(2)若 $S_1\setminus M\neq\Phi$

假设点 m_1 不是凸壳的顶点,则包含在 S_1 中的凸壳顶点只能位于 S_{12},S_{13} 中或为 m_2。

由上述假设可知,必存在两个点,不妨记为 z_1,z_2,满足如下两个条件:(1) z_1 与 z_2 是凸壳的顶点且线段 z_1z_2 为凸壳的一条边;(2) z_1 位于 S_{12} 中或 $z_1=a_1$,z_2 位于 S_{13} 中或 $z_2=d_2$ 或当 $m_2\neq m_1$ 时 $z_2=m_2$。由于已设定 $S_1\neq\Phi$,因此 $z_1=a_1$ 与 $z_2=d_2$ 不可能同时成立,则显然 m_1 与 d_2 位于直线 z_1z_2 的两侧或 m_1 与 a_1 位于直线 z_1z_2 的两侧。这显然与命题 1 相矛盾,故假设不成立,即 m_1 是凸壳的顶点。同理可证 m_2 也是凸壳的顶点。

综上可知命题 3 成立。

下面构造算法 AⅡ来解决此类分割。

输入:(S_i,s_{i1},s_{i2})其中 s_{i1},s_{i2} 为 S_i 的关联顶点对。

输出:$(S_{ik},T(S_{ik}),s_{ik1},s_{ik2})k=1,2$

Begin

Step1 求出点 s_{i1} 与 s_{i2} 所在直线的方程 F。

Step2 求出集合 $W=\{w\in S_i \mid s\in S_i, w$ 到直线 $s_{i1}s_{i2}$ 的距离大于等于 s 到 $s_{i1}s_{i2}$ 的距离$\}$,$w_1=\min X(W)$,$W_2=\max X(W)$,线段 $s_{i1}s_{i2}$ 的中点 w_0。

Step3 求出点 s_{i1} 与 w_1 所在直线的方程 F_1,点 s_{i2} 与 w_2 所在直线的方程 F_2。

Step4 对 S_i 中的每一个点(例如 s),分别求出 $F_i(s)$ 和 $F_i(w_0)$ $i=1,2$。若 $F_1(w_0)$ 与 $F_1(s)$ 异号,则将点 s 放入 S_{i1} 中,并且 $T(S_{i1})$ 加 1;若 $F_2(w_0)$ 与 $F_2(s)$ 异号,则将点 s 放入 S_{i2} 中,并且 $T(S_{i2})$ 加 1。显然 s 最多只能在 S_{i1} 与 S_{i2} 中的一个集合中。若 s 既不在 S_{i1} 也不在 S_{i2} 中,则 s 为凸壳的内点。令 s_{i1},w_1 对应为 s_{i11},s_{i12},w_2,s_{i2} 对应为 s_{i21},s_{i22}。

End

把算法 AⅡ应用到 S_2,S_3,S_4,就得到 $4\times2=8$ 个相互独立的外沿三角形网格。进一步可将算法 AⅡ应用到这 8 个外沿三角形网格,这样就可以得到 $8\times2=16$ 个外沿三角形网格。将此过程反复进行,就可以得到多个外沿三角形网格,如图 5 所示。为了平衡各处理机的任务量,在每次分割前首先比较当前分割阶段已有的各外沿三角形网格中所含点的个数,然后将点数最多的那一个外沿三角形网格应用算法 AⅡ进行分割(上述过程的实现见算法 AⅢ)。这些外沿三角形网格有一个共同的特点:其中有一条边的两个端点为凸壳的顶点。在将各子问题的解归并为原问题解的过程中将用到此特性。只需比较各子凸壳顶点列的首尾点,即可将各子问题的解链接成原问题的解。

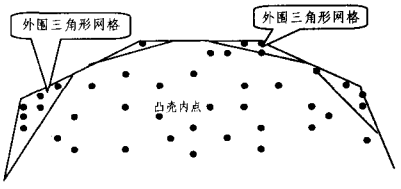

图 5

算法 AⅢ

输入:点集 S

输出:$(S_i,T(S_i)\ (s_{i1},s_{i2}))$

Begin

Step1 AⅠ(S) /* 调用算法 AⅠ */

Step2 找出当前含点数最大的外沿三角形网格所对应的集合且统计当前非空外沿三角形网格的个数 L。若 $L=0$,则凸壳顶点已求出程序停止执行;若 $L\neq0$ 且 L 小于处理机的台数 p,则进入 Step3;若 $L=p$,则对 S 的

分割操作完成，程序停止执行。

Step3 调用算法 AⅡ对在 Step2 中确定的点数最大的集合进行分割，返回 Step2。

End

显然算法 AⅢ的时间复杂度为 $O(n)$。

由于 S 为有限集，所以在算法 AⅢ结束时只会得到有限个 S 的子集，即有限个外沿三角形网格。不妨设共得到 t 个外沿三角形网格。若 $T(S_i)=0$，则令 $H(S_i)=\{S_i$ 的关联顶点对 $s_{i1}, s_{i2}\}$，对非空的 p 个子集可以放入 p 台处理机中并行处理。不妨记 p 个非空集为 $V_1, V_2, \cdots, V_p$。

利用三角形网格划分方法对原点集进行分割，有如下几个优点：(1)可以根据实际情况的需要将原点集中有可能成为凸壳顶点的点分割在多个外沿三角形网格中；(2)在各外沿三角形网格中求凸壳顶点是彼此相互独立的(可以用类似命题 2 的证明方法进行证明)，因此在各子问题的处理过程中不需要机间通信，这不仅可以降低通信成本，还可以避免因通信阻塞而限制并行执行的效率；(3)极易将各子问题的解归并为原问题的解；(4)对原问题进行分解的时间复杂度为 $O(n)$。

5 基于工作站机群的凸壳并行算法

工作站机群(COW)是实现并行计算的一种新的主流技术。这种系统将一群工作站用某种结构的网络互联起来，充分利用各工作站的资源，统一调动、协调处理，以实现高效的并行计算。COW 的一般结构如图 6 所示。当采用上述三角形网格划分方法对原问题进行分割后，在诸子问题的求解过程中一个子问题的解决不依赖其它子问题的处理结果，所以对主机与各工作站以及工作站与工作站之间的通信要求较低。这非常适合工作站机群的体系结构。主机将原问题分割成若干个同构的、规模大致相等的子问题，然后将各子问题播送到各工作站(一到多个人通信)。各工作站独立地完成所接收到的任务，最后将子问题的解返回到主机(单点收集)并在主机中将各子问题的解归并为原问题的解。

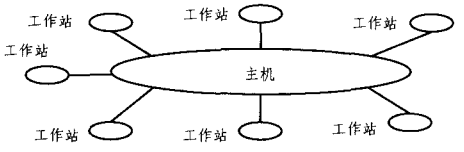

图 6

输入：二维点集 $S=\{s_1, s_2, \cdots, s_n\}$

输出：二维点集的凸壳 $H(S)$

Begin

Step1 AⅢ(S) / * 调用算法 AⅢ * /

Step2 For $i=1$ to p do / * 将在 Step1 中得到的 p 个非空集播送到 p 个工作站 * /

Send (V_i, v_{i1}, v_{i2}) to workstation P_i

/ * v_{i1}, v_{i2} 为集合 V_i 的关联顶点对 * /

Receive (V_i, v_{i1}, v_{i2}) from case

End for

Step3 For $k=1$ to p par-do / * 在 p 台工作站中并行处理 * /

HULL(v_i, v_{i1}, v_{i2}) / * 调用已有的凸壳串行算法 * /

End for

Step4 For $i=1$ to p do / * 将 p 个工作站的结果播送回主机 * /

Send $H(V_i, v_{i1}, v_{i2})$ to case

Receive $H(V_i, v_{i1}, v_{i2})$ from workstation P_i

End for

Step5 通过比较各子凸壳顶点列(包括在 Step1 与 Step3 中得到的子凸壳顶点列，在 Step1 中得到的各子凸壳顶点为各空外沿三角形网格的关联顶点对)的首尾元素将各子问题的解归并为原问题的解。

End

结束语 在解决一个复杂的问题时，我们常用的一种方法是将复杂问题分割成多个子问题。分割的思想有多种，如均匀划分技术、分治技术、平衡树技术、倍增技术等。衡量一种分割方法在处理特定问题时的优劣，通常有如下标准：各子任务的均衡性、并行处理时的通信成本及软件工程成本、子问题的解归并为原问题解的难易程度等。本文构造的三角形网格划分在解决二维凸壳问题时能很好地满足以上标准。最后本文根据此分割的特点构造了一种基于工作站机群的凸壳并行算法。在结束此文时笔者想发表一点愚见：人们已经构造了许多凸壳的串行算法，且有一些算法已经达到了凸壳问题在最坏情况下的时间下限，因此若想进一步提高解决凸壳问题的效率，凸壳的并行化处理应是一个值得研究的领域。

参 考 文 献

[1] 周启海，黄涛，吴红玉，等. 基于最大基线倾角智能逼近的凸壳新算法[J]. 计算机科学，2007(9)：206-208

[2] 郑永果，徐晓丹，岳昊. 求平面点集凸壳的一个最优算法[J]. 福建电脑，2005(7)：47-48

[3] 黄涛，周启海. 双域单向水平倾角最小化圈绕凸壳新算法[J]. 计算机科学，2007(11)：208-211

[4] 刘丽娜，唐振军，张显全. 基于凸多边形的凸壳算法[J]. 计算机科学，2006(9)：218-221

[5] 马丽平，杨炳儒，樊广佺. 平面点集凸壳的一种快速算法[J]. 地理与地理信息科学，2006(06)：38-41

[6] 刘光惠，陈传波. 求解简单多边形和平面点集凸包的新算法[J]. 计算机科学，2007，34(12)：222-226

[7] 曾纯强. 用归约原理求解凸壳问题的计算复杂性的研究[J]. 软件导刊，2006(11)：24-25

[8] 陈国良. 并行计算 [M]. 北京：高等教育出版社，2002

(上接第 240 页)

[9] Bouzidi A, Baaziz N. Contourlet Domain Feature Extraction for Image Content Authentication // Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing. 2006：202-206

[10] Do M N, Vetterli M. Framing Pyramids. IEEE Transactions Signal Processing, 2003, 51(9)：2329-2342

[11] Do M N, Vetterli M. The contourlet transform: an efficient directional multiresolution image representation. IEEE Transactions on Image Processing, 2005, 14(12)：2091-2106

[12] Cox I J, Kilian J, Leighton F T. Secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing, 1997, 6(12)：1673-1678

[13] Barni M, Bartolini F, Piva A. Improved wavelet based watermarking through pixel-wise masking. IEEE Transactions Image Processing, 2001, 10(5)：783-791