# 基于标签路径的Web结构化数据自动抽取

李 贵 陈 成 李征宇 韩子扬 孙 平 孙焕良
(沈阳建筑大学信息与控制工程系 沈阳 110168)

**摘 要** 提出了一种基于标签路径的Web结构化数据自动抽取方法。该方法通过对网页DOM树的解析获取完整标签路径集合,并依据路径相似度测量方法来聚类标签路径,实现目标数据区域的定位,然后通过标签节点坐标位置的特性来分离各个数据项,过滤冗余数据,最终完成数据抽取。实验结果表明,与MDR方法相比,该方法在处理拥有结构化数据的网页时,有更高的查全率和查准率。

**关键词** 标签路径,结构化数据抽取,聚类
**中图法分类号** TP301.6 **文献标识码** A

## Automatic Web Structured Data Extraction Based on Tag Path

LI Gui CHEN Cheng LI Zheng-yu HAN Zi-yang SUN Ping SUN Huan-liang
(Faculty of Information & Control Engineering, Shenyang Jianzhu University, Shenyang 110168, China)

**Abstract** This paper introduces a method based on tag path to extract Web structured data. The method gets complete tag path collection by parsing the DOM tree of the Web document. Clustering of tag paths is performed based on introduced similarity measure and the data area can be targeted, then taking advantage of features of tag position, we can separate and filter record, finally complete data extraction. Experiments show that this method achieves higher precision and recall than the MDR when dealing with the page containing the structured data.

**Keywords** Tag path, Extracting structured data, Clustering

## 1 引言

当前,Web页面已经成为人们获取信息的主要途径。Web页面中的数据多以嵌套的结构化形式展现。这些数据主要来源于后台数据库中。在本文中,网页中的表格数据和那些具有相同模式的一类数据统称为结构化数据。结构化数据作为一种重要的数据组织和表现形式已经广泛地应用于Web页面中。整合从大量网页中抽取来的这些结构化数据可以为人们提供增值服务。但由于HTML语言所表述的Web页面经过浏览器解析后只适合人们浏览,不适合作为一种数据交换方式由计算机来处理[5],因此如何让计算机程序自动准确地抽取出结构化的目标数据,一直是个挑战性极强的难题。

传统的数据抽取方法,无论是手工标注,还是包装器归纳都不能应用于站点数量巨大的情形,而且维护开销很大。因此,全自动地从列表页抽取数据的方法开始被广泛关注。但由于网页中的数据都是被高度包装而且模式互不相同,因此如何准确地定位目标数据区域成为全自动数据抽取的瓶颈问题。

因此,本文提出了一种基于标签路径聚类的方法来解决程序自动抽取的难题。该方法通过网页的整体分析来捕获列表数据对象。我们重点关注文档对象模型DOM树中一个完整的标签路径是怎样重复地出现在网页中,并基于提出的相似度测量方法聚类标签路径从而定位目标数据区域,再通过标签节点坐标位置的特性来分离、过滤数据,最终完成数据抽取。实验结果表明,本文提出的方法在处理包含结构化数据的网页时,拥有更高的准确度。

## 2 相关工作

概括起来,全自动抽取结构化数据的技术包括如下几种方法:EXALG[7]提出了在多张网页中统计标记出现频率来确定一组标记是否能形成一个目标模板的方法。Gengxin Miao[6]等人提出了利用在一张页面中唯一标记路径匹配来抽取数据记录的方法。但前两者在时间复杂度和效率上不尽如人意。MDR[3]是LiuBing等人提出的一个比较经典的数据挖掘算法,它利用的是被称作广义节点的数据段之间的编辑距离来评估连续的结点是否组成了一个数据对象。但MDR有一个局限就是不能处理结构复杂的网页。实验表明,针对大型商务网站中的非表格数据项的抽取,MDR经常会出现错误而挂起。

本文提出的方法主要针对于拥有结构化数据的Web网页的抽取。在实际应用中,我们要抽取的网页虽然源于不同的脚本,拥有着不同的结构,但其全都是含有结构化数据的Web页面,而我们的目标正是将这些页面的列表信息抽取集

本文受国家自然科学基金(61070024)资助。

李 贵(1964—),男,博士,教授,主要研究方向为Web数据挖掘与信息集成、分布对象技术、软件工程,E-mail:Ligui21c@sina.com;陈 成(1988—),男,硕士生,主要研究方向为Web数据挖掘。

成，以提供增值服务。所以本文提出的方法是具有现实意义的。

## 3 基于标签路径的自动抽取方法

基于标签路径的自动抽取方法基本思路如下：

①HTML 解析生成 DOM 树。分析输入的 HTML 文件，为 HTML 页面生成对应的 DOM 树结构。目前可以使用 HTML 解析器 Jsoup[4] 来实现。

②标签路径生成。从 DOM 树中的叶子节点开始遍历，获取每一个叶子节点到根节点的完整路径。

③标签路径聚类。以标签路径集合作为输入，通过平均链接聚类算法，获得潜在的目标数据区域的标签路径。

④路径过滤。对目标路径集进行数据的分离与过滤，得到抽取的目标集合。

⑤数据记录抽取。通过目标集合中叶子节点的坐标位置抽取数据集。

### 3.1 标签路径生成

文档对象模型 DOM 是 W3C 制定的一种独立于平台和具体编程语言的 API 接口标准。它提供了一个标准的对象集合，用以表示和处理 HTML 或 XML 文档及其各组成部分之间的关系。图 1(a)展示了一个简单的 HTML 文档的 DOM 树模型。

本文所指的标签路径是指由根节点到叶子节点的完整标签路径。对应于图 1(a)DOM 树的标签路径如图 1(b)所示。

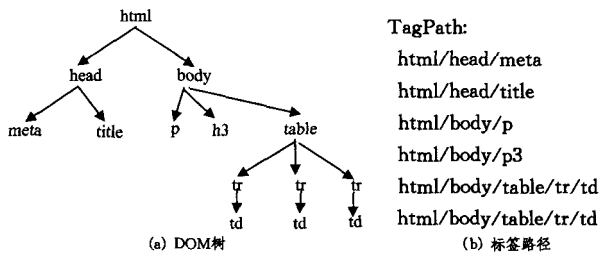


图 1 DOM 树和标签路径

在这里定义一个二元组〈path，index〉，path 表示标签路径字符串，index 表示由该路径上标签节点位置组成的字符串。生成二元组集合的算法 pathGenerator 如下。

**算法 1** pathGenerator

输入：URL

输出：二元组集合 list

```
1)DOM←parseHTML(URL)
2)for(each node in DOM){
3)  if(The node is a leaf)
4)  List〈path,index〉 list←get_path(node);}
5)return list.
```

算法中 get_path(node)表示获取从 node 到根节点 html 的标签路径字符串及相应的位置坐标字符串。

需要注意的是在获得生成的标签路径结果集后，要对其进行清理简化，因为在 HTML 页面中，有很多标签用以布局和美观，例如〈br〉。简化操作可以很大程度上减小聚类工作的冗余度。依据对 HTML 标签的分析，本文总结出以下形式的标签路径可以被列入清理范围：以标签〈pre〉、〈br〉、〈input〉和〈option〉结尾的标签路径。

### 3.2 标签路径聚类

#### 3.2.1 标签路径的相似度定义

我们把每一个标签路径看成是一个由很多标签组成的集合，这样就可以从集合的角度给出两个标签路径的相似度定义公式：

$$\mathrm{Sim}(p_1,p_2)=\frac{|p_1 \underset{\rightarrow}{\cap} p_2|}{\mathrm{Max}\{|p_1|,|p_2|\}} \tag{1}$$

式中，$|p_1|$表示标签路径 $p_1$ 中标签的个数(在实验中我们使用“/”来截出标签)。$|p_1 \underset{\rightarrow}{\cap} p_2|$表示 $p_1$ 和 $p_2$ 从起始标签位置 html 开始相同的标签个数，例如两个标签路径 html/body/table/tr/a 和 html/body/p/a，则 $p_1 \underset{\rightarrow}{\cap} p_2$ 仅包含 html 和 body，而不是 html，body 和 a。在这里没有只考虑两个标签集合的交集，而是从集合的起始位置开始一一匹配。这是因为如果两个标签前继节点不同，那么两个标签即使相同也不再有意义。所以，如果 $P_1$ 表示标签路径为 html/body/table/tr/td/a，$P_2$ 表示为 html/body/p3/a，则 $P_1$，$P_2$ 的相似度 $\mathrm{Sim}(P_1,P_2)=2/6=0.33$。

有了两个标签路径的相似度描述，我们可以定义一个标签路径和一个路径集合的相似度为这个标签路径与路径集中的每一个路径的相似度的平均值，如式(2)所示：

$$\mathrm{Sim\_}Aver(L,p_j)=\frac{\sum_i^n \mathrm{Sim}(p_i,p_j)}{n} \tag{2}$$

#### 3.2.2 标签路径聚类

基于提出的标签路径相似度计算方法，本节使用平均链接聚类算法对标签路径聚类。其核心思想是检查两个目标聚类的数据点之间的平均距离是否大于相似度阈值，如果大于则合并两个聚类，重复处理文档聚类，直到无法再合并为止。具体算法描述如下。

**算法 2** averageLinkCompute

输入：相似度阈值 h，标签路径集合 list

输出：聚类集合 C

```
1)N=List.size();
2)for(i=0;i<N;i++){
3)  str1=List.get(i);
4)  for(j=i+1;j<N;j++)
5)    d[i][j]= Sim(Pi,Pj);}
6)for(m=0;m<N;m++){
7)  if(Pm not used)
8)    for(n=m;n<N;n++)
9)      if(ds[m][n]>=h and Pnnot used)
10)       if(m=n or Sim_Aver(L,Pn)>h){
11)         L←Pn;
12)         Pnused;}
13)     C←L}
14)return C;
```

算法第 1 行获取标签路径集的大小 $N$，算法 2—5 行计算路径之间相似度的邻接矩阵。6—13 行表示如果 $L$ 为空或者标签路径 $p$ 和类 $L$ 的相似度大于阈值，则把 $p$ 加入 $L$ 中并设为已用。最后将不同的聚类 $L$ 加入 $C$ 中。

其中 $\mathrm{Sim}(P_i,P_j)$和 $\mathrm{Sim_}Aver(L,P_n)$分别按照式(1)、式(2)来计算。算法 2 对某市房产信息网 http://www.syfc.com.cn/work/xjlp/new_building.jsp 的处理结果如图 2 所示。

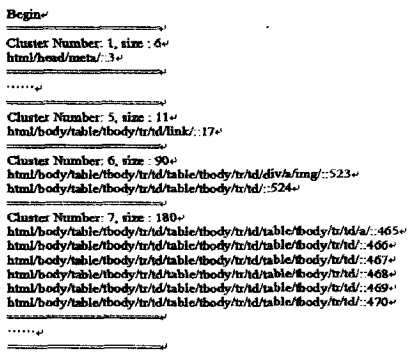


图 2　标签路径聚类结果

为了显示较多聚类结果，图 2 中的每个聚类都删除了很多标签，图中每个标签路径后面的数字表示其路径叶子节点的位置(抽取数据时需要使用)。

### 3.3　数据抽取

在本节中，我们要使用上一节中产生的聚类结果抽取目标数据。在这里，选取拥有标签路径最多的聚类作为我们的抽取目标。所要抽取的都是拥有结构化数据的信息网页，而这些网页中所要表达的主体信息的标签路径是高度相似且反复出现的，那么通过聚类算法得出的众多聚类中，容量最大的聚类一定包含着表达目标信息的标签路径，在此我们不考虑例如门户网站首页之类的网页。

#### 3.3.1　数据项的分离与过滤

图 2 展示的标签路径聚类中，最大的聚类有 180 个标签路径。那么我们又该如何从中分离出每个数据项并且去除其中的冗余数据呢？为了解决这一问题，我们介绍数据项标志位(后文简称标志位)的概念。在定义标志位之前，先来看表 1 中的例子，表 1 的第 1 行和第 3 行就是图 2 中最大聚类的两个标签路径，2，4 行分别表示 1，3 行标签所对应的位置。从表 1 中可以看出，13 列之前，两个标签路径的坐标位置完全相同。而从 13 列 tr 开始，则有所不同。那么就可以认为在坐标位置为 230 的 tbody 下的每一个 tr 都代表着一个数据项，即在目标聚类中所有在 13 列坐标为 463 的 tr 都代表着一个数据项。在这里，就把这个 tr 列的位置称为数据项标志位。

表 1　标签路径和标签坐标位置

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| html | body | table | tbody | tr | td | table | tbody | tr | td | table | tbody | tr | td |
| 1 | 12 | 26 | 27 | 28 | 179 | 223 | 224 | 227 | 228 | 229 | 230 | 463 | 470 |
| html | body | table | tbody | tr | td | table | tbody | tr | td | table | tbody | tr | td |
| 1 | 12 | 26 | 27 | 28 | 179 | 223 | 224 | 227 | 228 | 229 | 230 | 455 | 462 |

寻找数据项标志位算法就是从第一列开始，比较聚类中每一个标签路径在该列的节点的坐标位置是否相同，如出现不同，则返回该列的位置。算法描述如下。

**算法 3**　findItemIndex

输入：最大标签路径聚类集合 list⟨path，index⟩

输出：数据项标志位

```
1)len=min(list);
2)for(i=0;i<len;i++){
3)   p←random(list);
4)   for(j=0;j<list. size;j++)
5)     if(p[i]≠list. get(j)[i])
6)       return j;}
7)return-1
```

算法首行表示获取集合中最短路径的长度 *len*。

现在通过算法 3 得到了数据项标志位，但这个标志位极有可能是不正确的。因为在我们的聚类中有可能包含了噪音标签路径，这样路径的存在可能使得标志位提前。在这里，我们加入过滤算法来检测标志位并且过滤掉噪音标签路径。

过滤算法分为 3 步：首先，按照算法 3 得出的标志位分离数据项。其思想在算法 4 中给出。主要就是将标志位上的数据坐标进行分类，若数据坐标相同则将标签路径分为一类。然后，检测分离出的各个类。我们要检测经过第一步分离后各个类的大小。在这里，需要注意这样两个事实：1)如果我们的标志位因为一些噪音路径的存在而提前的话，那么按照这个标志位分类后的结果一定会产生一个包含了目标数据信息和有可能存在其他噪音的庞大的类，以及一些含有噪音路径的其他类；2)一个数据项属性列的个数一般不会超过 10。基于以上事实，我们就可以设定递归条件。当分类后，某个类的数量大于阈值(实验中阈值取 0.6 * 标签集合的大小)时，我们把这个大类作为目标数据类继续调用过滤算法。当分类后某个类的数量大于 10 小于阈值时，算法将初始标志位加 1 后递归地调用过滤算法。这样进行不断的分离、检测、过滤，最终得到正确的分类结果。过滤算法具体描述如算法 5 所示。

**算法 4**　itemSepatator

输入：最大标签路径聚类集合 list，以及数据项标志位 index

输出：分类后的集合 map⟨key，value⟩，每一个 key 值表示一个类，而 value 存储了该类下的路径节点坐标。

```
1)for(p in list){
2)   if(p [index] has not in the map)
3)      key=p[index];
4)   value=p[list. size()];}//position of leaf node
5) return map;
```

算法从首行开始遍历 list 集合中的标签，将路径在标志位上的标签坐标和叶子节点坐标分别加入到 key 和 value 中，最后返回 map。

**算法 5**　filter

输入：最大标签路径聚类集合 list，以及数据项标志位 index，x 表示阈值。

输出：分类后的集合 map⟨key，value⟩，每一个 key 值表示一个类，而 value 存储了该类下的路径节点坐标。

```
1)map=itemSeparator(list,index);
2)for(⟨key,value⟩ in map){
3)   if(value. size()>=x){
4)      filter(select(list,firstindex,key),++index);break;}
5)   else if(value. size>10&&value. size<x)
6)      filter(list,++index);break;}
7)   return map;
```

表 2 给出了对图 2 中包含 180 个路径的最大聚类过滤的部分结果。其中每一个 key 值都代表着一个数据项，value 中的数值则代表着在该数据项中不同属性的叶子节点的位置。

表 2　过滤结果

| key | Value |
|---|---|
| 231 | {233,234,235,236,237,238} |
| 455 | {457,458,459,460,461,462} |
| 463 | {465,466,467,468,469,470} |

3.3.2　数据抽取

从表 2 的 value 中，得到了一个数据项下不同路径的叶子节点的坐标信息，通过这些坐标，可以定位网页中要抽取的节点，并以此抽取信息(在实验中，我们使用 HTML 解析器 Jsoup 的 Elements.get(int index)方法可以完成按位置抽取数据)。

## 4　实验

根据以上讨论，本文实现了全部算法。具体实验的硬件环境为 Celeron 2.66GHz，1G 内存，160G 硬盘，Windows xp，开发工具为 MyEclipse7.0 以及 JDK1.6。聚类算法中的相似度阈值为 0.75。本实验中所用的页面全部来自于实际网站。

### 4.1　实验结果

实验网页如图 3 所示，网址为 http://www.syfc.com.cn/work/xjlp/new_building.jsp，系统处理时间为 2.2s。


图 3　网页实例

图 3 中矩形框标注区域为目标数据区域样式，图 4 展示了数据抽取结果，其中的每一行代表着一个数据项。

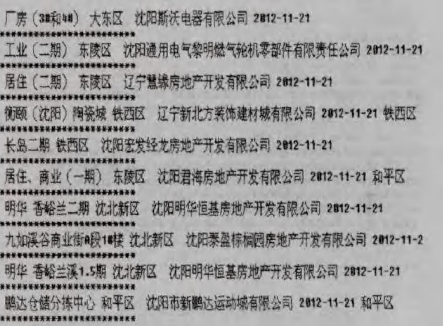


图 4　数据抽取结果

### 4.2　结果评估

本实验比较本文的算法与经典算法 MDR 的抽取效果(MDR 系统可在网址 http://www.cs.uic.edu/~liub/获取到，MDR 系统的相似度阈值取 0.6)。实验结果如表 3 所列，表 3 中的第 1 列给出了每一次实验的序号，其中前 5 行来源于我们实际项目中需要抽取的页面，后面的实验则是两个算法在不同领域的页面中的抽取结果。第 2 列表示页面的 URL，有些表格中 URL 并没有全部列出。第 3 列表示页面中实际数据项个数。最后 4 列中 corr 表示抽取到的正确数据项的个数，found 表示抽取到的总数据项的个数。表中的 er 表示发生错误没有得到数据项。表格的后 3 行给出各数据项列的总数，并使用式(3)和式(4)来计算两个算法的查准率和查全率。对于发生错误而没有得到结果的行，数据项个数按 0 来处理。若将其排除在外，则得到方括号中的结果。

$$Precision=\frac{|\text{抽取到的正确数据项}|}{|\text{抓取的所有数据项}|} \tag{3}$$

$$Recall=\frac{|\text{抽取到的正确数据项}|}{|\text{实际网页中包含的数据项}|} \tag{4}$$

表 3　实验结果

| | URL | Obj | MDR | | Our | |
|---|---|---|---|---|---|---|
| | | | corr | found | corr | found |
| 1 | http://www.syfc.gov.cn/work/index/index.jsp | 30 | 30 | 30 | 30 | 30 |
| 2 | http://www.181.cc/newhouse/house.aspx | 36 | 36 | 41 | 36 | 36 |
| 3 | http://www.dl-fangdi.com.cn/fdc/D01XmxxAction.do | 11 | 11 | 11 | 11 | 11 |
| 4 | http://60.31.254.197/bit-xxzs/xmlpzs/nowwebissue.asp | 21 | 21 | 21 | 21 | 21 |
| 5 | http://www.cq315house.com/315/esfxx.asp?sort=3 | 17 | 17 | 17 | 17 | 17 |
| 6 | http://www.taobao.com/ | 20 | er | er | 20 | 20 |
| 7 | http://www.dangdang.com/ | 25 | er | er | 25 | 27 |
| 8 | http://www.tudou.com/home/item_u113713953s0p7.html | 40 | er | er | 40 | 40 |
| 9 | http://www.sjzu.edu.cn/more.asp?title=信息速递 | 15 | 2 | 2 | 15 | 15 |
| 10 | http://hotels.ctrip.com/ | 25 | 25 | 25 | 25 | 25 |
| | Total | 240 | 142 | 147 | 240 | 242 |
| | Precision | | 97% | | 99% | |
| | Recall | | 59%[91%] | | 100% | |

从表 3 中可以看出，MDR 对于某些站点的网页，容易发生错误而得不到任何结果(这类网页一般都拥有复杂的网页结构，如表中的 6—8 行)。实验结果表明，本文的算法拥有更好的容错性，可以对拥有复杂结构的网页进行抽取，抽取的总体质量要优于 MDR，获得了更高的查全率和查准率。当然，实验样本集选取的不同对查全率和查准率也有一定的影响。

**结束语**　针对网页中的数据记录，本文提出了一种新的数据抽取方法。它通过标签路径聚类来定位目标数据区域，并根据路径叶子节点坐标位置抽取目标数据。该方法是全自动跨领域的，不需要人工干预。实验表明，该方法能够成功地抽取网页中结构化的数据记录。

下一步工作包括：

(1)对抽取出的大量信息，将按照不同属性集成到数据库中。

(2)在抽取中，对热点数据区域链接指向的网页进行二次

(下转第 165 页)

### 2.2 一种改进的 RHT 检测算法

在传统的 RHT 算法中，在得到真实圆之后从边缘点集 $D$ 中删除了轮廓上与真实圆重合的点，而最优拟合结果有很大的概率被传统算法删掉，造成最优拟合结果丢失。因此，我们提出一种改进的 RHT 算法：

1)对原图像进行去噪等预处理；

2)运用 Canny 算子检测得到边缘图像，初始化边缘点集 $D$，并计算边缘点集 $D$ 中每个点的梯度方向 $\nabla f$，初始化参数单元集 $P$=NULL；

3)在点集 $D$ 中随机选取 3 点，判断这 3 点的法线方向与梯度方向的角度之差是否在允许范围内，是，则转 4)；否则转 2)；

4)计算 3 点所确定的圆的参数 $p(a,b,r)$，搜索参数单元集 $P$，若 $\| p-p_c \|$，$\delta$ 是允许的误差范围，则将对应参数单元 $p_c$ 的计数值 $v$ 加 1，若 $v$ 值达到指定阈值 $n_t$(一般很小，本文取为 2)，则 $p_c$ 对应的圆成为候选圆 $C_{候}$；

5)计算边缘点集 $D$ 中落在 $C_{候}$ 上的点数 $m_{pc}$，若 $m_{pc}$ 大于构成圆所需的最小点数 $m_{\min}$，则认为 $C_{候}$ 为真实圆 $C_{真实}$，记录此时落在该 $C_{真实}$ 上的所有属于边缘点集 $D$ 的点，形成点集 $D_{真实}$；

6)随机删除点集 $D_{真实}$ 中 $\lambda D_{真实}$ 个点，$\lambda$ 为删除系数是 0～1 的某个值，将剩下的点重新放入边缘点集 $D$，更新边缘点集 $D$，将 $P$ 置为空，并返回 3)；

7)当随机采样次数 $k$ 大于采样数阈值 $k_{\min}$ 时，若 $P$ 中仍没有参数单元 $p_c$ 对应计数 $v$ 达到 $n_t$，则认为 $D$ 中没有共圆点，算法结束。

## 3 实验结果

实验是在 2G 内存，处理器主频为 2.0GHz，64 位处理器 PC 机上进行的，所处理的图片是经过简单减噪处理后的实际拍摄图片，实现语言是 C++，我们分别选取了值为 0.3 和 0.7作为删减系数来完成实验。处理时间对比如表 1 所列，实验结果如图 3 所示，其中图 3(a)为减噪后的原图像，图 3(b)是 Canny 检测边缘化后的图像，图 3(c)为删减系数 $\lambda$ 为 0.3 时的检测结果，图 3(d)是删减系数 $\lambda$ 为 0.7 的检测结果。实验结果表明，可以引入删减过程来提高最优轮廓出现的可能性，删减系数越小，检测结果越好，但花费的时间较长，可以找到一个最优的删减系数 $\lambda$ 值，满足实际中的要求，这也是接下来的研究方向。

表 1 时间对比

| 删除系数 λ | 计算时间(ms) |
|---|---|
| 0.3 | 673 |
| 0.7 | 205 |

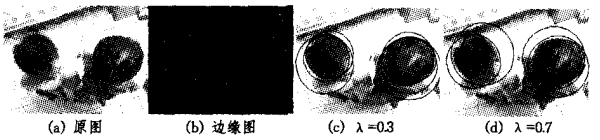


图 3 实验结果对比

**结束语** 通过实验结果对比可以看出，本文改进的 RHT 算法速度上虽然比传统的 RHT 算法有些增加，但是获取的头部轮廓的拟合结果更加准确和优秀，并且增加的时间在实践和实验中是可以接受的。删除系数 $\lambda$ 的选取是实验中的重点，可在今后的研究中做深入探讨。

## 参 考 文 献


[1] 徐培智，徐贵力，黄鑫. 基于随机 Hough 变换的人头检测[J]. 计算机工程，2012，38(1)：64-67

[2] 赵桂霞，黄山. 一种基于随机 Hough 变换圆检测的改进算法[J]. 计算机技术与发展，2008，18(4)：77

[3] 姜伟，高军伟，刘新. 基于边缘背景差法和 Hough 变换的公交乘客头部检测方法研究[J]. 青岛大学学报：工程技术版，2010，25(2)：

[4] Hollitt C. Reduction of computational complexity of Hough transforms using a convolution approach[C]// Image and Vision Computing New Zealand，2009. IVCNZ '09. 24th International Conference，2009：373-378

[5] 黄永林，叶玉堂，陈镇龙，等. 一种新的快速 Hough 变换圆检测方法事[J]. 电子测量与仪器学报，2009，24(9)：837

[6] 于海滨. 基于头部特征提取的人体检测与跟踪及其应用[D]. 杭州：浙江大学信息科学与工程学院，2007

[7] 虞旦，韦巍. 改进的随机圆检测算法[J]. 浙江大学中国图像图形学报，2009，14(8)

[8] Kcrbywn D J，Atherton T J. Circle Detection Using Hough Transform Filters[D]. University of Warwick，U. K. 1995

[9] 屈稳太. 基于弦中点 Hough 变换的椭圆检测方法[J]. 浙江大学学报：工学版，2005，39(8)

[10] Chen Xing，Lu Ling，Gao Yang. A New Concentric Circle Detection Method Based on Hough Transform[C]// The 7th International Conference on Computer Science & Education (ICCSE 2012). Melbourne，Australia，July 2012


(上接第 144 页)

抽取，以获取其中有价值的数据记录。

## 参 考 文 献


[1] 孙吉贵，刘杰，赵连宇. 聚类算法研究[J]. 计算机研究与发展，2008(19)：48-61

[2] Liu Bing. Web Data Mining [M]. 俞勇，薛贵荣，韩定一，译. 北京：清华大学出版社，2009：291-295

[3] Liu Bing，Grossman R，Zhai Ya-nong. Ming data records in web pages [C]// Proceedings of the ACM International on Knowledge Discovery and Data Ming. 2003：601-606

[4] Jsoup：Java Html Parser[OL]. http://jsoup.org/apidocs/

[5] 李效东，顾毓清. 基于 DOM 的 Web 信息提取[J]. 计算机学报，2002，25

[6] Miao G，Tatemura J，Hsiung Wang-pin，et al. Extracting data records from the Web using tag path clustering[C]// Madrid. 2009

[7] Arasu A，Garcia-Molina H. Extracting structured data from Web pages[C]// Proc of ACM SIGMOD International Conference on the Management of Data. 2003：337-348

[8] Cafarella M J，Halevy A，Wang D Z，et al. Exploring the power of tables on the tables on the Web [C]// Proceedings of 34th International Conference on Very Large Data Bases. 2008：538-549