

原 LIRe 源代码分析 5：提取特征向量[以颜色布局为例]

2013年11月02日 17:24:48 阅读数：5857

=====

LIRe源代码分析系列文章列表：

[LIRe 源代码分析 1：整体结构](#)

[LIRe 源代码分析 2：基本接口（DocumentBuilder）](#)

[LIRe 源代码分析 3：基本接口（ImageSearcher）](#)

[LIRe 源代码分析 4：建立索引（DocumentBuilder）\[以颜色布局为例\]](#)

[LIRe 源代码分析 5：提取特征向量\[以颜色布局为例\]](#)

[LIRe 源代码分析 6：检索（ImageSearcher）\[以颜色布局为例\]](#)

[LIRe 源代码分析 7：算法类\[以颜色布局为例\]](#)

=====

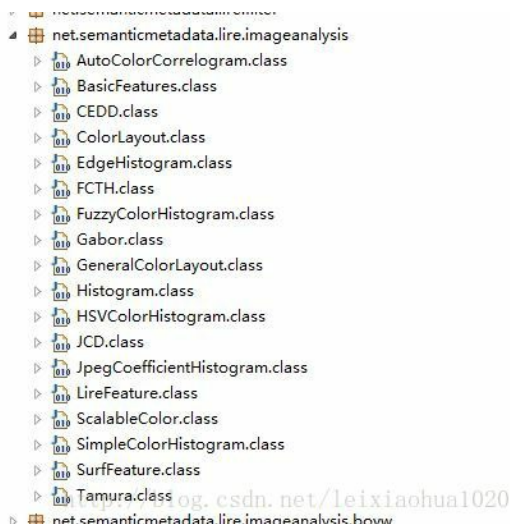
在上一篇文章中，讲述了建立索引的过程。这里继续上一篇文章的分析。在ColorLayoutDocumentBuilder中，使用了一个类型为ColorLayout的对象vd，并且调用了vd的extract()方法：

```
[java] 1. ColorLayout vd = new ColorLayout();
      2. vd.extract(bimg);
```

此外调用了vd的getByteArrayRepresentation()方法：

```
[java] 1. new Field(DocumentBuilder.FIELD_NAME_COLORLAYOUT_FAST, vd.getByteArrayRepresentation())
```

在这里我们看一看ColorLayout是个什么类。ColorLayout位于“net.semanticmetadata.lire.imageanalysis”包中，如下图所示：



由图可见，这个包中有很多的类。这些类都是以检索方法的名字命名的。我们要找的ColorLayout类也在其中。看看它的代码吧：

```
[java] 1. /*
      2.  * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
      3.  * LIRe is free software; you can redistribute it and/or modify
      4.  * it under the terms of the GNU General Public License as published by
      5.  * the Free Software Foundation; either version 2 of the License, or
      6.  * (at your option) any later version.
      7.  *
      8.  * LIRe is distributed in the hope that it will be useful,
      9.  * but WITHOUT ANY WARRANTY; without even the implied warranty of
     10.  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
     11.  * GNU General Public License for more details.
```

```

11.  * GNU General Public License for more details.
12.  *
13.  * You should have received a copy of the GNU General Public License
14.  * along with LIRE; if not, write to the Free Software
15.  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
16.  *
17.  * We kindly ask you to refer the following paper in any publication mentioning Lire:
18.  *
19.  * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 欽
20.  * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
21.  * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
22.  *
23.  * http://doi.acm.org/10.1145/1459359.1459577
24.  *
25.  * Copyright statement:
26.  * -----
27.  * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
28.  * http://www.semanticmetadata.net/lire
29.  */
30. package net.semanticmetadata.lire.imageanalysis;
31.
32. import net.semanticmetadata.lire.imageanalysis.mpeg7.ColorLayoutImpl;
33. import net.semanticmetadata.lire.utils.SerializationUtils;
34.
35. /**
36.  * Just a wrapper for the use of LireFeature.
37.  * Date: 27.08.2008
38.  * Time: 12:07:38
39.  *
40.  * @author Mathias Lux, mathias@juggle.at
41.  */
42. public class ColorLayout extends ColorLayoutImpl implements LireFeature {
43.
44.     /*
45.      public String getStringRepresentation() {
46.      StringBuilder sb = new StringBuilder(256);
47.      StringBuilder sbtmp = new StringBuilder(256);
48.      for (int i = 0; i < numYCoeff; i++) {
49.          sb.append(YCoeff[i]);
50.          if (i + 1 < numYCoeff) sb.append(' ');
51.      }
52.      sb.append("z");
53.      for (int i = 0; i < numCCoeff; i++) {
54.          sb.append(CbCoeff[i]);
55.          if (i + 1 < numCCoeff) sb.append(' ');
56.          sbtmp.append(CrCoeff[i]);
57.          if (i + 1 < numCCoeff) sbtmp.append(' ');
58.      }
59.      sb.append("z");
60.      sb.append(sbtmp);
61.      return sb.toString();
62.      }
63.
64.      public void setStringRepresentation(String descriptor) {
65.          String[] coeffs = descriptor.split("z");
66.          String[] y = coeffs[0].split(" ");
67.          String[] cb = coeffs[1].split(" ");
68.          String[] cr = coeffs[2].split(" ");
69.
70.          numYCoeff = y.length;
71.          numCCoeff = Math.min(cb.length, cr.length);
72.
73.          YCoeff = new int[numYCoeff];
74.          CbCoeff = new int[numCCoeff];
75.          CrCoeff = new int[numCCoeff];
76.
77.          for (int i = 0; i < numYCoeff; i++) {
78.              YCoeff[i] = Integer.parseInt(y[i]);
79.          }
80.          for (int i = 0; i < numCCoeff; i++) {
81.              CbCoeff[i] = Integer.parseInt(cb[i]);
82.              CrCoeff[i] = Integer.parseInt(cr[i]);
83.          }
84.      }
85.      */
86.
87.
88.      /**
89.       * Provides a much faster way of serialization.
90.       *
91.       * @return a byte array that can be read with the corresponding method.
92.       * @see net.semanticmetadata.lire.imageanalysis.CEDD#setByteArrayRepresentation(byte[])
93.       */
94.      public byte[] getByteArrayRepresentation() {
95.          byte[] result = new byte[2 * 4 + numYCoeff * 4 + 2 * numCCoeff * 4];
96.          System.arraycopy(SerializationUtils.toBytes(numYCoeff), 0, result, 0, 4);
97.          System.arraycopy(SerializationUtils.toBytes(numCCoeff), 0, result, 4, 4);
98.          System.arraycopy(SerializationUtils.toByteArray(YCoeff), 0, result, 8, numYCoeff * 4);
99.          System.arraycopy(SerializationUtils.toByteArray(CbCoeff), 0, result, numYCoeff * 4 + 8, numCCoeff * 4);
100.         System.arraycopy(SerializationUtils.toByteArray(CrCoeff), 0, result, numYCoeff * 4 + numCCoeff * 4 + 8, numCCoeff * 4);
101.         return result;
102.     }

```

```

102.    }
103.
104.    /**
105.     * Reads descriptor from a byte array. Much faster than the String based method.
106.     *
107.     * @param in byte array from corresponding method
108.     * @see net.semanticmetadata.lire.imageanalysis.CEDD#getByteArrayRepresentation
109.     */
110.    public void setByteArrayRepresentation(byte[] in) {
111.        int[] data = SerializationUtils.toIntArray(in);
112.        numYCoeff = data[0];
113.        numCCoeff = data[1];
114.        YCoeff = new int[numYCoeff];
115.        CbCoeff = new int[numCCoeff];
116.        CrCoeff = new int[numCCoeff];
117.        System.arraycopy(data, 2, YCoeff, 0, numYCoeff);
118.        System.arraycopy(data, 2 + numYCoeff, CbCoeff, 0, numCCoeff);
119.        System.arraycopy(data, 2 + numYCoeff + numCCoeff, CrCoeff, 0, numCCoeff);
120.    }
121.
122.    public double[] getDoubleHistogram() {
123.        double[] result = new double[numYCoeff + numCCoeff * 2];
124.        for (int i = 0; i < numYCoeff; i++) {
125.            result[i] = YCoeff[i];
126.        }
127.        for (int i = 0; i < numCCoeff; i++) {
128.            result[i + numYCoeff] = CbCoeff[i];
129.            result[i + numCCoeff + numYCoeff] = CrCoeff[i];
130.        }
131.        return result;
132.    }
133.
134.    /**
135.     * Compares one descriptor to another.
136.     *
137.     * @param descriptor
138.     * @return the distance from [0,infinite) or -1 if descriptor type does not match
139.     */
140.
141.    public float getDistance(LireFeature descriptor) {
142.        if (!(descriptor instanceof ColorLayoutImpl)) return -1f;
143.        ColorLayoutImpl cl = (ColorLayoutImpl) descriptor;
144.        return (float) ColorLayoutImpl.getSimilarity(YCoeff, CbCoeff, CrCoeff, cl.YCoeff, cl.CbCoeff, cl.CrCoeff);
145.    }
146. }

```

ColorLayout类继承了ColorLayoutImpl类，同时实现了LireFeature接口。其中的方法大部分都是实现了LireFeature接口的方法。先来看看LireFeature接口是什么样子的：

注：这里没有注释了，仅能靠自己的理解了。

```

1.  /**
2.   * This is the basic interface for all content based features. It is needed for GenericDocumentBuilder etc.
3.   * Date: 28.05.2008
4.   * Time: 14:44:16
5.   *
6.   * @author Mathias Lux, mathias@juggle.at
7.   */
8.  public interface LireFeature {
9.      public void extract(BufferedImage bimg);
10.
11.      public byte[] getByteArrayRepresentation();
12.
13.      public void setByteArrayRepresentation(byte[] in);
14.
15.      public double[] getDoubleHistogram();
16.
17.      float getDistance(LireFeature feature);
18.
19.      java.lang.String getStringRepresentation();
20.
21.      void setStringRepresentation(java.lang.String s);
22.  }

```

我简要概括一下自己对这些接口函数的理解：

- 1.extract(BufferedImage bimg)：提取特征向量
- 2.getByteArrayRepresentation()：获取特征向量（返回byte[]类型）
- 3.setByteArrayRepresentation(byte[] in)：设置特征向量（byte[]类型）
- 4.getDoubleHistogram()：

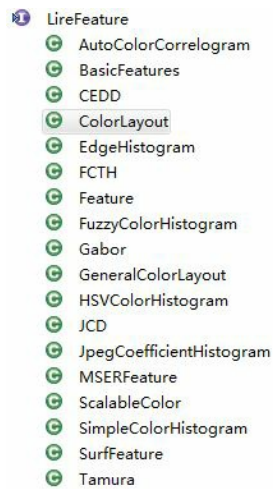
5.getDistance(LireFeature feature)：

6.getStringRepresentation()：获取特征向量（返回String类型）

7.setStringRepresentation(java.lang.String s)：设置特征向量（String类型）

其中咖啡色的是建立索引的过程中会用到的。

看代码的过程中发现，所有的算法都实现了LireFeature接口，如下图所示：



不再研究LireFeature接口，回过头来本来想看看ColorLayoutImpl类，但是没想到代码其长无比，都是些算法，暂时没有这个耐心了，以后有机会再看看吧。以下贴出个简略版的。注意：该类中实现了extract(BufferedImage bimg)方法。其他方法例如getBytesRepresentation()则在ColorLayout中实现。

```
[java]
1. package net.semanticmetadata.lire.imageanalysis.mpeg7;
2.
3. import java.awt.image.BufferedImage;
4. import java.awt.image.WritableRaster;
5.
6.
7. /**
8.  * Class for extrcating & comparing MPEG-7 based CBIR descriptor ColorLayout
9.  *
10.  * @author Mathias Lux, mathias@juggle.at
11.  */
12. public class ColorLayoutImpl {
13.     // static final boolean debug = true;
14.     protected int[][] shape;
15.     protected int imgYSize, imgXSize;
16.     protected BufferedImage img;
17.
18.     protected static int[] availableCoeffNumbers = {1, 3, 6, 10, 15, 21, 28, 64};
19.
20.     public int[] YCoeff;
21.     public int[] CbCoeff;
22.     public int[] CrCoeff;
23.
24.     protected int numCCoeff = 28, numYCoeff = 64;
25.
26.     protected static int[] arrayZigZag = {
27.         0, 1, 8, 16, 9, 2, 3, 10, 17, 24, 32, 25, 18, 11, 4, 5,
28.         12, 19, 26, 33, 40, 48, 41, 34, 27, 20, 13, 6, 7, 14, 21, 28,
29.         35, 42, 49, 56, 57, 50, 43, 36, 29, 22, 15, 23, 30, 37, 44, 51,
30.         58, 59, 52, 45, 38, 31, 39, 46, 53, 60, 61, 54, 47, 55, 62, 63
31.     };
32.
33.     ...
34.     public void extract(BufferedImage bimg) {
35.         this.img = bimg;
36.         imgYSize = img.getHeight();
37.         imgXSize = img.getWidth();
38.         init();
39.     }
40.     ...
41. }
```

版权声明：本文为博主原创文章，未经博主允许不得转载。<https://blog.csdn.net/leixiaohua1020/article/details/13775467>

文章标签：[lire](#) [源代码](#) [索引](#) [检索](#) [lucene](#)

个人分类：[LiRe](#) [MPEG7/图像检索](#)

所属专栏：[开源多媒体项目源代码分析](#)

此PDF由[spyyg](#)生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com