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：5850

==

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]

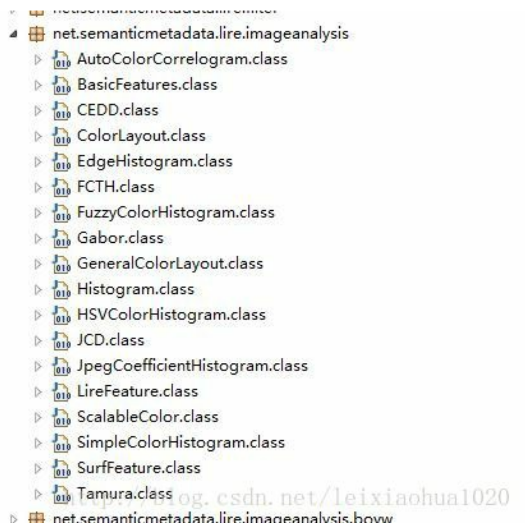
```java
ColorLayout vd = new ColorLayout();
vd.extract(bimg);
```

此外调用了vd的getByteArrayRepresentation()方法：

[java]

```java
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：



```
net.semanticmetadata.lire.imageanalysis
  AutoColorCorrelogram.class
  BasicFeatures.class
  CEDD.class
  ColorLayout.class
  EdgeHistogram.class
  FCTH.class
  FuzzyColorHistogram.class
  Gabor.class
  GeneralColorLayout.class
  Histogram.class
  HSVColorHistogram.class
  JCD.class
  JpegCoefficientHistogram.class
  LireFeature.class
  ScalableColor.class
  SimpleColorHistogram.class
  SurfFeature.class
  Tamura.class
net.semanticmetadata.lire.imageanalysis.bovw
```

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]

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * LIRe is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥�
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * --------------------
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */
package net.semanticmetadata.lire.imageanalysis;

import net.semanticmetadata.lire.imageanalysis.mpeg7.ColorLayoutImpl;
import net.semanticmetadata.lire.utils.SerializationUtils;

/**
 * Just a wrapper for the use of LireFeature.
 * Date: 27.08.2008
 * Time: 12:07:38
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public class ColorLayout extends ColorLayoutImpl implements LireFeature {

    /*
        public String getStringRepresentation() {
        StringBuilder sb = new StringBuilder(256);
        StringBuilder sbtmp = new StringBuilder(256);
        for (int i = 0; i < numYCoeff; i++) {
            sb.append(YCoeff[i]);
            if (i + 1 < numYCoeff) sb.append(' ');
        }
        sb.append("z");
        for (int i = 0; i < numCCoeff; i++) {
            sb.append(CbCoeff[i]);
            if (i + 1 < numCCoeff) sb.append(' ');
            sbtmp.append(CrCoeff[i]);
            if (i + 1 < numCCoeff) sbtmp.append(' ');
        }
        sb.append("z");
        sb.append(sbtmp);
        return sb.toString();
    }

    public void setStringRepresentation(String descriptor) {
        String[] coeffs = descriptor.split("z");
        String[] y = coeffs[0].split(" ");
        String[] cb = coeffs[1].split(" ");
        String[] cr = coeffs[2].split(" ");

        numYCoeff = y.length;
        numCCoeff = Math.min(cb.length, cr.length);

        YCoeff = new int[numYCoeff];
        CbCoeff = new int[numCCoeff];
        CrCoeff = new int[numCCoeff];

        for (int i = 0; i < numYCoeff; i++) {
            YCoeff[i] = Integer.parseInt(y[i]);
        }
        for (int i = 0; i < numCCoeff; i++) {
            CbCoeff[i] = Integer.parseInt(cb[i]);
            CrCoeff[i] = Integer.parseInt(cr[i]);

        }
    }
     */

    /**
     * Provides a much faster way of serialization.
     *
     * @return a byte array that can be read with the corresponding method.
     * @see net.semanticmetadata.lire.imageanalysis.CEDD#setByteArrayRepresentation(byte[])
     */
    public byte[] getByteArrayRepresentation() {
        byte[] result = new byte[2 * 4 + numYCoeff * 4 + 2 * numCCoeff * 4];
        System.arraycopy(SerializationUtils.toBytes(numYCoeff), 0, result, 0, 4);
        System.arraycopy(SerializationUtils.toBytes(numCCoeff), 0, result, 4, 4);
        System.arraycopy(SerializationUtils.toByteArray(YCoeff), 0, result, 8, numYCoeff * 4);
        System.arraycopy(SerializationUtils.toByteArray(CbCoeff), 0, result, numYCoeff * 4 + 8, numCCoeff * 4);
        System.arraycopy(SerializationUtils.toByteArray(CrCoeff), 0, result, numYCoeff * 4 + numCCoeff * 4 + 8, numCCoeff * 4);
        return result;
    }
    }

    /**
     * Reads descriptor from a byte array. Much faster than the String based method.
     *
     * @param in byte array from corresponding method
     * @see net.semanticmetadata.lire.imageanalysis.CEDD#getByteArrayRepresentation
     */
    public void setByteArrayRepresentation(byte[] in) {
        int[] data = SerializationUtils.toIntArray(in);
        numYCoeff = data[0];
        numCCoeff = data[1];
        YCoeff = new int[numYCoeff];
        CbCoeff = new int[numCCoeff];
        CrCoeff = new int[numCCoeff];
        System.arraycopy(data, 2, YCoeff, 0, numYCoeff);
        System.arraycopy(data, 2 + numYCoeff, CbCoeff, 0, numCCoeff);
        System.arraycopy(data, 2 + numYCoeff + numCCoeff, CrCoeff, 0, numCCoeff);
    }

    public double[] getDoubleHistogram() {
        double[] result = new double[numYCoeff + numCCoeff * 2];
        for (int i = 0; i < numYCoeff; i++) {
            result[i] = YCoeff[i];
        }
        for (int i = 0; i < numCCoeff; i++) {
            result[i + numYCoeff] = CbCoeff[i];
            result[i + numCCoeff + numYCoeff] = CrCoeff[i];
        }
        return result;
    }

    /**
     * Compares one descriptor to another.
     *
     * @param descriptor
     * @return the distance from [0,infinite) or -1 if descriptor type does not match
     */

    public float getDistance(LireFeature descriptor) {
        if (!(descriptor instanceof ColorLayoutImpl)) return -1f;
        ColorLayoutImpl cl = (ColorLayoutImpl) descriptor;
        return (float) ColorLayoutImpl.getSimilarity(YCoeff, CbCoeff, CrCoeff, cl.YCoeff, cl.CbCoeff, cl.CrCoeff);
    }
}
```

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。

[java]

```java
/**
 * This is the basic interface for all content based features. It is needed for GenericDocumentBuilder etc.
 * Date: 28.05.2008
 * Time: 14:44:16
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public interface LireFeature {
    public void extract(BufferedImage bimg);

    public byte[] getByteArrayRepresentation();

    public void setByteArrayRepresentation(byte[] in);

    public double[] getDoubleHistogram();

    float getDistance(LireFeature feature);

    java.lang.String getStringRepresentation();

    void setStringRepresentation(java.lang.String s);
}
```

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()：

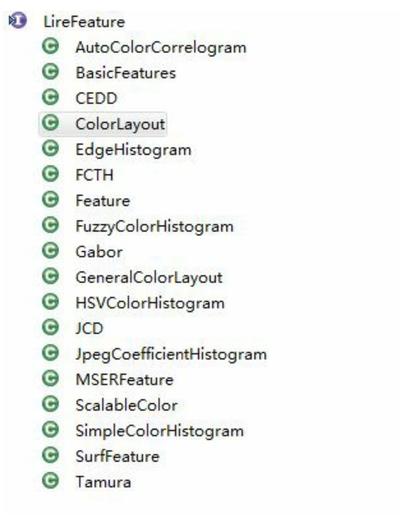
5.getDistance(LireFeature feature)：

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：


LireFeature
AutoColorCorrelogram
BasicFeatures
CEDD
ColorLayout
EdgeHistogram
FCTH
Feature
FuzzyColorHistogram
Gabor
GeneralColorLayout
HSVColorHistogram
JCD
JpegCoefficientHistogram
MSERFeature
ScalableColor
SimpleColorHistogram
SurfFeature
Tamura

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getByteArrayRepresentation()则在ColorLayout中实现。

[java]

```java
package net.semanticmetadata.lire.imageanalysis.mpeg7;

import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;


/**
 * Class for extrcating & comparing MPEG-7 based CBIR descriptor ColorLayout
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public class ColorLayoutImpl {
    // static final boolean debug = true;
    protected int[][] shape;
    protected int imgYSize, imgXSize;
    protected BufferedImage img;

    protected static int[] availableCoeffNumbers = {1, 3, 6, 10, 15, 21, 28, 64};

    public int[] YCoeff;
    public int[] CbCoeff;
    public int[] CrCoeff;

    protected int numCCoeff = 28, numYCoeff = 64;

    protected static int[] arrayZigZag = {
            0, 1, 8, 16, 9, 2, 3, 10, 17, 24, 32, 25, 18, 11, 4, 5,
            12, 19, 26, 33, 40, 48, 41, 34, 27, 20, 13, 6, 7, 14, 21, 28,
            35, 42, 49, 56, 57, 50, 43, 36, 29, 22, 15, 23, 30, 37, 44, 51,
            58, 59, 52, 45, 38, 31, 39, 46, 53, 60, 61, 54, 47, 55, 62, 63
    };

    ...
    public void extract(BufferedImage bimg) {
        this.img = bimg;
        imgYSize = img.getHeight();
        imgXSize = img.getWidth();
        init();
    }
    ...
}
```

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