Gici Spectral Transform Software Manual

Formerly RKLT software

(version 1.1)

GICI group

Department of Information and Communications Engineering

Universitat Autònoma Barcelona

http://www.gici.uab.es - http://gici.uab.cat/GiciWebPage/downloads.php

January 2010

1 Description

This software is an implementation of the Karhunen-Loêve Transform (KLT) to be used as spectral decorrelator. In addition it features many state-of-the-art techniques for spectral decorrelation:

- Lossy Karhunen-Loêve Transform.
- Matrix factorizations for reversible integer mapping [1, 2].
- Covariance Subsampling [3, 4].
- Clustered and Multi-level transforms [5, 6].

2 Requirements

This software is programmed in Java, so you might need a JAVA Runtime Environment(JRE) to run this application. We have used SUN JAVA 1.5.

JAI The Java Advanced Imaging (JAI) library is used to load and save images in formats other than raw or pgm. The JAI library can be freely downloaded from *http://java.sun.com*. **Note:** You don't need to have this library installed in order to compile the source code.

GSL Eigendecomposition functions are from the GNU Scientific Library (GSL) and have been translated into Java. The authors of the of original code are Gerard Jungman and Brian Gough. (see source files for details)

3 Usage

The application is provided in a single file, a jar file (*dist/rklt.jar*), that contains the application. Along with the application, the source code is also provided. If you need to rebuild the jar file, you can use the `ant` command.

To launch the application you can use the following command:

```
$ java -Xmx1200m -jar dist/rklt.jar --help
```

In a GNU/Linux environment you can also use the shell script `rklt` situated at the root of the RKlt directory.

```
$ ./rklt --help
```

Some examples of usage are provided below:

- Coding and decoding an image with the Reversible KLT:

```
$ ./rklt -i inptfile-16bpppb-bigendian-224x512x512.raw \
         -o rkltfile-16bpppb-bigendian-224x512x512.raw \
         -ig 224 512 512 3 0 -og 224 512 512 4 0 \
         -ti side-information.file -D 0 -d 0

$ ./rklt -i rkltfile-16bpppb-bigendian-224x512x512.raw \
         -o outpfile-16bpppb-bigendian-224x512x512.raw \
         -ig 224 512 512 4 0 -og 224 512 512 3 0 \
         -ti side-information.file -D 0 -d 1
```

- Forward transform with covariance subsampling enabled:

```
$ ./rklt -i inptfile-16bpppb-bigendian-224x512x512.raw \
         -o rkltfile-16bpppb-bigendian-224x512x512.raw \
         -ig 224 512 512 3 0 -og 224 512 512 4 0 \
         -ti side-information.file -D 0 -d 0 \
         -es 0.01
```

- Using the dynamic structure defined in [6]:

```
$ ./rklt -i inptfile-16bpppb-bigendian-224x512x512.raw \
         -o rkltfile-16bpppb-bigendian-224x512x512.raw \
         -ig 224 512 512 3 0 -og 224 512 512 4 0 \
         -ti side-information.file -D 0 -d 0 \
         -es 0.01 --enableClustering 2 56 1
```

- Perform a lossy KLT and create a JPEG2000-compatible bitstream:

```
$ ./rklt -i infile.raw -ig $Z $Y $X 3 0 --lossy \
         --dumpEquivalentMatrix KLT_matrix -d 0

$ ./matrix_range_increase.sh "KLT_matrix.klt" $Z 16 \
  | tr '\n' ',' | sed 's/,$/\n/' > "new_range.txt"

$ echo -n "Mvector_coeffs:I2=" > "KLT_Matrix_KDU_param.txt"
$ hexdump -v -e '"%f,"' "$TEMPFOLDER/KLT_matrices.means" \
  | sed 's/,$//' >> "KLT_Matrix_KDU_param.txt"
$ echo -n " Mmatrix_coeffs:I3=" >> "KLT_Matrix_KDU_param.txt"
$ hexdump -v -e '"%f,"' "$TEMPFOLDER/KLT_matrices.klt" \
  | sed 's/,$//' >> "KLT_Matrix_KDU_param.txt"

$ kdu_compress_fixed -quiet Clayers=1 Cycc=no -no_weights \
           -i infile.raw\*$Z@$((X*Y*2)) -o "output.jpx" \
           Creversible=no Mcomponents=$Z Msigned=yes \
           Mprecision=16 Mvector_size:I2=$Z \
           Mmatrix_size:I3=$((Z*Z)) -s "KLT_Matrix_KDU_param.txt" \
           Mstage_inputs:I1=\{0,$((Z-1))\} \
```

```
Mstage_outputs:I1=\{0,$((Z-1))\} \
Mstage_collections:I1=\{$Z,$Z\} \
Mstage_xforms:I1=\{MATRIX,3,2,0,0\} Mnum_stages=1 Mstages=1 \
Sdims=\{$Y,$X\} Sprecision=$(cat "new_range.txt") Ssigned=yes \
Qstep=0.0000001 -rate $BITRATE
```

where `./matrix_range_increase.sh` is:

```
#!/bin/bash
# Calculates the range increase produced by the KLT
cat "$1" | transpose.pl | hexdump -ve '1/8 "%f\n"' \
 | awk 'BEGIN {a=0; b=0; c='"$2"'; d='"$3"'};
    {if ($1>0) a+=$1; else a-=$1; b++;
     if(b == c) {b=0; print int(log(a)/log(2)-2000)+2000+d; a = 0;} };' \
 | sed 's/-inf/1/'
```

Note: the read buffer of Kakadu for option files must be enlarged to allow for such a long command line.

4 Notes

If you need further assistance, you might want to contact us directly.

References

[1] P. W. Hao and Q. Y. Shi, "Matrix factorizations for reversible integer mapping," *IEEE Trans. Signal Process.*, vol. 49, no. 10, pp. 2314–2324, 2001.

[2] L. Galli and S. Salzo, "Lossless hyperspectral compression using KLT," *IEEE Int'l Geosci. and Remote Sens. Symp. Proc. (IGARSS 2004)*, vol. 1-7, pp. 313–316, 2004.

[3] B. Penna, T. Tillo, E. Magli, and G. Olmo, "A new low complexity KLT for lossy hyperspectral data compression," *IGARSS 2006. IEEE International Conference on Geoscience and Remote Sensing Symposium, 2006.*, pp. 3525–3528, 2006.

[4] ——, "Transform coding techniques for lossy hyperspectral data compression," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 5, pp. 1408–1421, May 2007.

[5] I. Blanes and J. Serra-Sagristà, "Clustered reversible-KLT for progressive lossy-to-lossless 3d image coding," in *Data Compression Conf. 2009 (DCC 2009)*. IEEE Press, Mar. 2009, pp. 233–242.

[6] ——, "Cost and scalability improvements to the Karhunen-Loêve transform for remote-sensing image coding," *IEEE Trans. Geosci. Remote Sens.*, 2010.