Software documentation

Trygve Randen

September 13, 1994

Contents

Part I

Programs

Chapter 1

Programs for complete filtering and segmentation

1.1 segment (shell script)

Purpose

Perform a complete filtering, neural network training and classification in order to test and evaluate classification by a specific method on a specific textured image.

Description

For large images, run:"

```
setenv TMPDIR <dir>"
```

first, where `<dir>` is a directory with plenty of free space."

Usage

```
segment [-dec]
    [-sn structure number <integer>]
    [-f filter name <string>] [-symm]
    [-ref reference image <filename>]
    [-class classifier <string>]
    [-classes number of classes <integer>]
    [-cut border width <integer>]
    [-oimg output image <filename>]
    [-ostat output statistics <filename>]
```

-U Print usage information

-sn filter structure number

-f filter name (default i-2-1-09). Options: Farro: Farrokhnia's (90) method, Gabor: Gabor filter with structure given by -sn, f..: FIR QMF PR filters, i..: IIR QMF PR filters and w..: Wavelet QMF PR filters

-dec decimate the filter output?

-symm symmetrically extend prior to filtering?

-ref reference (tag) image

-class classifier (lvq/som/k-means) (def. lvq)

-classes number of classes (default 2)

-cut cut away a border to avoid edge effects (default 0)

-oimg output classified image (default segment.hips)

-ostat output statistics file (default segment.stats)

Date created

March 1994 (shell script)

Chapter 2

Filter programs

2.1 filter (shell script)

Purpose

Filter an image

Usage

```
filter [-dec]
    [-sn structure number <integer>]
    [-f filter name <string>] [-symm]
    [-if input image <filename>]
    [-ff output feature file <filename>]
    [-sf output statistics <filename>]
```

-U Print usage information

-sn filter structure number

-f filter name (default i-2-1-09). Options: Farro: Farrokhnia's (90) method, Gabor: Gabor filter with structure given by -sn, f..: FIR QMF PR filters, i..: IIR QMF PR filters and w..: Wavelet QMF PR filters

-dec decimate the filter output?

-symm symmetrically extend prior to filtering?

-if input image file

-ff output feature file

-sf output statistics file (default segment.stats)

Date created

March 1994

2.2 qmf (Quadrature Mirror Filterbank)

Purpose

Filter bank based on tree-structured prototype filters. Employing both decimated and full-rate analysis filterbank and decimated synthesis filterbank, and both general FIR filters and IIR filters based on polyphase allpass filters.

Description

The program will read the filterparameters from files specified in the parameters and perform the filtering on the input image.

Note that a filter identifier starting with 'f' is assumed to represent a FIR filter, an identifier starting with 'i' is a polyphase-allpass IIR filter and an identifier starting with 'w' is a wavelet FIR filter.

Usage

```
qmf [-sn struct<integer> |
     -s struct<string> |
     -sb struct<string>]
    [-f filter-id.<string>] [-d]
    [-analysis | -synthesis]
    [imagefile]
```

-f Filter specification file to be used as default filter [f16b]

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

-d Critically decimated filter (default FALSE). When a decimated filter is used, there will be one output frame per input frame (with the same size as the input — a mosaic of the decimated subbands). When a fullrate filter is used, there will be one output frame per subband per input frame — the same size as the input image.

-analysis Analysis filterbank (default TRUE).

-synthesis Synthesis filterbank.

Date created

March 1993

2.3 decimate

Purpose

Decimate an image sequence in both vertical and horizontal direction.

Description

Read the input image and copy pixels on the decimation grid to an image that is written to the standard out. Only the region of interest is processed, the rest is left out.

This program may also perform "simulated" decimation. Simulated decimation means copying the pixelvalues that would have been kept in the decimation onto the pixelvalues that would have been removed.

Usage

```
decimate
     [-N vertical decimation<integer>
      [horizontal decimation<integer>]]
     [-sn struct<integer> |
      -s struct<string> |
      -sb struct<string>]
     [imagefile]
```

-N If only vertical decimation is specified, horizontal decimation is set equal to the vertical decimation. Default value is two.

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

If any of the parameters `-s`, `-sb` or `-sn` is used, then this program performs "simulated" decimation on the image corresponding to the critical sampling of a filter with the specified filterstructure, and the simulated decimatedimage is then decimated by the factor defined by `-N`.

Date created

March 1993

2.4 mcf (Multi Channel Filter)

Purpose

Multichannel Gabor filter with nonlinear transform and smmothing. Preprocessor for textured images for texture segmentation.

Description

Read in the input image, transform it by a DFT and apply a set of one channel filters to this image. The set of filters includes:

- Filtering by a Gabor filter with frequency and orientation selective properties
- Some kind of linear transform on the image pixels. Currently implemented is the absolute value nonlinearity and the $\tanh(\cdots)$ nonlinearity as described by Farrokhnia.
- Averaging (blurring) of the filtered image by a Gaussian lowpass filter

The output from this program is a image sequence where the input image has one output image per filter in the filter set. The output is scaled to the maximum dynamic range of the output format. Be aware that this scaling implies that training a classifier on the output of the filter-response of one specific image and classification on the output of another is impossible, because the two outputs will probably be scaled differently.

See: "Multi-channel filtering techniques for texture segmentation and surface quality inspection", Farshid Farrokhnia, Ph.D. dissertation, Michigan State University, Dept. of Elect. Eng., 1990

And: "Determining Gabor-filter parameters for texture segmentation", Dennis F. Dunn, William E. Higgins and Joseph Wakeley, SPIE Vol. 1826, Intelligent robots and computer vision XI (1992), Pages 51-63

Usage

```
mcf [-Even | -Gen [SigmaX<real>
                   [SigmaY<real>]] |
     -s struct<string> |
     -sn struct<integer> |
     -sb struct<string>]
    [-suvf factor in sigma-u in
           filter<real>
     [..sigma-v..<real>]]
    [-theta numargs<integer> arg1...argn]
    [-U0 numargs<integer> arg1...argn]
```

```
[-V0 numargs<integer> arg1...argn]
[-tanh | -nnl]
[-alpha alpha in nonlin.
        transf.<real>]
[-sf gaussian window fact.<real>]
[-navg] [imagefile]
```

-Even Use the even symm. Gabor filter, as described by Farrokhnia

-Gen Use the general Gabor filter, as described by Dunn. Parameters: SigmaX, SigmaY

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

-tanh Use the $\tanh(\cdots)$ nonlinearity, as described by Farrokhnia

-alpha alpha parameter in $\tanh(\cdots)$ nonlinearity, as described by Farrokhnia

-U0 Centre frequency parameter(s) to the Gabor filter and the $\tanh(\cdots)$ nonlinearity

-V0 Second centre frequency parameter(s) for a general Gabor filter. If specified, the number of values must equal the number of values for U0. See Dunn.

-theta Angle parameter(s) to the Gabor filter.

-suvf Sigma U factor: parameter to the Gabor filter (Farrokhnia) Sigma V factor: parameter to the Gabor filter (Farrokhnia)

-sf Factor to sigma in $\tanh(\cdots)$

-navg Do not average the feature-images

-nnl Do not perform any nonlinear transform, or rather only apply the magnitude operation.

A parametervalue of -1 means automatic calculation This program will not accept inputsequences with more than one frame. Specifying no parameters implies applying a even-symmetric Gabor filter with Farrokhnias default parameters.

Date created

January 1993

2.5 centrfreq

Purpose

Determine the centre frequencies of a general Gabor filter giving the maximum step signature for a given texture pair.

Description

See: "Determining Gabor-filter parameters for texture segmentation", Dennis F. Dunn, William E. Higgins and Joseph Wakeley, SPIE Vol. 1826, Intelligent robots and computer vision XI (1992), Pages 51-63.

Usage

```
centrfreq
    [-s % of samples to use for
          estimating centre
          frequencies<real>]
    [-sigma SigmaX<real> [SigmaY<real>]]
    [-seed random number seed<integer>]
    [-2mcf]
    imagefile1 [<] imagefile2
```

-s Percentage of pixels to use in computation (default 5)

-sigma Parameter in the Gabor filter

-seed Pseudo-random number generator seed

-2mcf Make the output be of the format required by mcf

Date created

March 1993

2.6 blurr

Purpose

Perform blurring (smoothing/low-pass filtering) on an image filtered by the program mcf without blurring or qmf.

Description

The blurring of a particular filtered frame is determined by the centre frequency of the corresponding passband. This program normally reads the filter structure in the same manner as the program qmf and performs blurring by the appropriate filter. It is also possible to specify a fixed centre frequency, as an alternative to specifying he filter structure.

Normal operation in image segmentation is to preceed the application of this program by the application of the tanh program

See: “Multi-channel filtering techniques for texture segmentation and surface quality inspection”, Farshid Farrokhnia, Ph.D. dissertation, Michigan State University, Dept. of Elect. Eng., 1990

Usage

```
blurr [-sn struct<integer> |
       -s struct<string> |
       -sb struct<string> |
       -cf centre frequency<real>]
      [-sf gaussian window factor<real>]
```

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

-cf Centre frequency which to tune the Gaussian lowpass filter to. Refer to Farshid Farrokhnia's Ph. D. dissertation for a description of the relationship between centre frequency and gaussian lowpass filter.

-d Blurr a critically sampled filter output, as created by `qmf -d` $\cdots$.

Date created

March 1993

2.7 tanh

Purpose

Apply the tanh(..) nonlinearity.

Description

See: “Multi-channel filtering techniques for texture segmentation and surface quality inspection”, Farshid Farrokhnia, Ph.D. dissertation, Michigan State University, Dept. of Elect. Eng., 1990 If any structure specification is given (parameter `-s`, `-sn` or `-sb`), it is assumed that the input is decimated, as generated by `qmf -d` $\cdots$, and each subband is treated separately (important when scaling is concerned).

Usage

```
tanh [-sn struct<integer> |
      -s struct<string> |
      -sb struct<string>]
    [-alpha alpha in nonlin.
         transf.<real>]
    [imagefile]
```

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

-alpha α parameter in tanh(..) nonlinearity

Date created

March 1993

Chapter 3

Conversion programs

3.1 hips2lvq

Purpose

Convert HIPS image sequences to a LVQ (Leaning Vector Quantization) datafile.

Description

The HIPS images are stored frame by frame while the LVQ datafiles are stored vector by vector (the elements in the vectors are pixelvalues from corresponding pixel positions in the frames). This implies that the program must build the LVQ datafile column by column as the frames are read in. For large images, this means that it needs a lot of disk space temporarily.

Alternately, the program can take decimated (as created by `qmf -d` $\cdots$) files as input (by using either of the filter structure parameters – `-s -sn -sb`). Then the first input frame is assumed to be in decimated format, while the rest are assumed to be full rate frames. This operation is much faster than the full rate operation.

Usage

```
hips2lvq [-d decimals<integer>]
    [-train training LVQ file<filename>]
    [-out LVQ datafile<filename>]
    [-p vector retain percentage<real>]
    [-seed random number seed<integer>]
    [-m mask file<string>]
    [-sn struct<integer> |
     -s struct<string> |
    -sb struct<string>]
    [imagefile1 imagefile2...imagefileN]
```

-d Number of decimals in output, default 0.

-train Let the last column be the tag data. The training dataset will be stored in the specified file, (p percent of the samples).

-p Percent of vectors to be kept. Used when the program creates training data to LVQ, when LVQ is supposed to be trained on a subset of the pixels in the image.

-seed Random seed. A negative value implies use of system clock.

-m Mask image. Only pixels where the corresponding pixels in the mask image are nonzero will be exported to LVQ format.

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

Input files Minimum one input file, there may be as many as wanted.

Date created

January 1993

3.2 hips2txt

Purpose

Convert a HIPS-2 image into a ASCII text file, with columns separated by spaces and lines separated by carriage returns.

Usage

```
hips2txt [-d decimals<integer>]
  [-lowleft] [imagefile]
```

-d Number of decimals in output, default 0

-lowleft Set origo to the lower left corner

Date created

February 1993

3.3 lvq2hips

Purpose

Convert a LVQ (Learning Vector Quantization) datafile to HIPS-2 format. The main use of this program is converting back classified data to an image.

Description

The input LVQ-file is assumed to be the output from classify, without any blank lines or any comment-lines. If not, the operation may not be predictable. The result is written to the standard output.

Usage

```
lvq2hips [-m mask file<string>] [-vis]
   [-p [prototype<filename>] |
    -s rows<integer> [cols<integer>]]
   [imagefile]
```

-m Mask image. The LVQ datafile is assumned to contain only the pixels where the corresponding pixels in the mask image are nonzero.

-vis The input is assumed to be of SOM (Self Organizing Map) format.

-p HIPS-2 image (header) file, input, this file is used to determine the size of the output image

-s Size of the output image

imagefile LVQ datafile. Input.

Date created

January 1993

3.4 raw2hips

Purpose

Program for converting a raw datafile to HIPS-2 format.

Description

Takes number of rows and columns from the commandline and converts the image. The output is written to the standard output.

Restriction: does only accept byte-format input.

Usage

```
raw2hips [-s rows<integer>
   [cols<integer>]]
   [-byte | -short | -int | -float |
    -complex | -double | -sbyte |
    -ushort | -uint | -hex]
```

-s Size of the input image, default 128 rows and the same number of columns as rows.

-byte Read a byte formatted raw file

-short Read a short formatted raw file

-int Read a int formatted raw file

-float Read a float formatted raw file

-complex Read a complex formatted raw file

-double Read a double formatted raw file

-sbyte Read a sbyte formatted raw file

-ushort Read a ushort formatted raw file

-uint Read a uint formatted raw file

-hex Read a byte-image stored in hexadesimal format

Date created

February 1993

3.5 txt2hips

Purpose

Program for converting a ASCII text datafile to HIPS-2 format.

Description

Restrictions:

- Does only accept byte-format input.
- For the simplicity of the programming, the user must specify imagesize.

Usage

```
txt2hips [-s rows<integer>
             [cols<integer>]]
```

-s rows [columns] Imagesize, default 128 rows and columns=rows

Date created

February 1993

3.6 dec2seq

Purpose

Convert the output from "qmf -d ..." (which is a mosaic of critically decimated subbands) into a sequence, where each subband is one frame. The frames will be the size of the least decimated subband, and more subsampled frames will have pixels replicated to fill the frame.

Usage

```
dec2seq [-fs]
  [-sn struct<integer> |
   -s struct<string> |
   -sb struct<string>]
  [imagefile]
```

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

-fs Full size output (i.e. don't subsample with the least common subsampling factor).

Data formats handled directly

- byte
- short
- int
- float

Date created

6/3-94

3.7 gif2hips

Purpose

Convert from GIF to HIPS format

Usage

```
gif2hips
```

Date created

March 1994

3.8 hips2eps

Purpose

Convert a HIPS image to Encapsulated PostScript

Usage

```
hips2eps [-bpp bits per pixel<integer>]
  [-scale] [imagefile]
```

-bpp Bits per pixel. If this parameter is not specified, it is determined automatically by the input header.

-scale Scale the image to exploit the maximum dynamic range of the output.

Date created

21 February 1994

3.9 hips2gif

Purpose

Convert from HIPS to GIF format

Usage

```
hips2gif
```

Date created

March 1994

3.10 hips2jpeg

Purpose

Convert from HIPS to JPEG format

Usage

```
hips2jpeg
```

Date created

March 1994

3.11 hips2pnm

Purpose

Convert from HIPS to PNM format

Usage

```
hips2pnm
```

Date created

March 1994

3.12 hips2rast

Purpose

Convert from HIPS to SUN raster format

Usage

```
hips2rast
```

Date created

March 1994

3.13 hips2tiff

Purpose

Convert from HIPS to TIFF format

Usage

```
hips2tiff
```

Date created

March 1994

3.14 hips2viff

Purpose

Convert from HIPS to VIFF (Khoros) format

Usage

```
hips2viff [-byte|-int|-float|-scale]
```

-byte Convert using byte format

-int Convert using integer format

-float Convert using floating point format

-scale Convert using byte format, scaled to the maximum dynamic range

Date created

March 1994

3.15 jpeg2hips

Purpose

Convert from JPEG to HIPS format

Usage

```
jpeg2hips
```

Date created

March 1994

3.16 pnm2hips

Purpose

Convert from PNM to HIPS format

Usage

```
pnm2hips
```

Date created

March 1994

3.17 rast2hips

Purpose

Convert from SUN raster to HIPS format

Usage

```
rast2hips
```

Date created

March 1994

3.18 seq2dec

Purpose

Convert a sequence to a format like the output from "qmf -d ..." (which is a mosaic of critically decimated subbands). This actually implies decimation of the sequence.

Usage

```
seq2dec
   [-sn struct&lt;integer&gt; |
    -s struct&lt;string&gt; |
    -sb struct&lt;string&gt;]
   [imagefile]
```

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

Date created

7/3-94

3.19 tiff2hips

Purpose

Convert from TIFF to HIPS format

Usage

```
tiff2hips
```

Date created

March 1994

3.20 tohips

Purpose

Convert from several image formats to HIPS

Description

Check the input image file name, and if indicated, convert to HIPS format. Otherwise: link outfile to infile

Usage

```
tohips <infile> <outfile>
```

Date created

March 1994

3.21 viff2hips

Purpose

Convert from VIFF (Khoros) to HIPS format

Usage

```
viff2hips
```

Date created

March 1994

Chapter 4

Other image processing programs

4.1 snr

Purpose

Compute the signal to noise ratio between two images

Description

Same program as snr written by John Haakon Husoey, but adapted to HIPS version 2.

Usage

```
snr imagefile1 [<] imagefile2
```

4.2 struct2latex

Purpose

Convert a filter structure description to a LaTeX picture indicating the subband-split on a square. This is a handy documentation tool for documenting filterstructures. The output image will be within a square of 1 x 1 in a standard unit determined by the enclosing LaTeX document.

Description

Read a filtertree specification from the commandline in one of three formats, and print a LaTeX compatible output to the standard output. For any of the inputs, the quad-tree specification is printed to the standard error.

Usage

```
struct2latex
   [-sn struct<integer> |
    -s struct<string> |
    -sb struct<string>]
```

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

Date created

April 1993

4.3 mktex

Purpose

Generate synthetic textures consisting of repeated line-segment texture elements.

Description

Program to generate artificial textures as defined in a .tx file. Relies on the OutputImage routine in `mktex-imout.c`.

The program generates textures consisting of texture elements with specified form repeated over a grid, with a range of random offsets and rotations.

Based on a program by: Y. King, April 1991, Laboratory for Computational Intelligence, University of British Columbia

Usage

```
mktex [<] infile.tx [>] outfile.hips
```

Format of the inputfile

```
texture: rectangle (<upper left corner>) - (<lower right corner>)
 orientation = [<min rot. angle>,<max rot. angle>]:<resolution>
 offset = [<max x offset>,<max y offset>]:[<x resolution>,
   <y res.>]
 pixels = <greylevel-value of pixels>

 <linesegments in texture element>
```

The following example file defines an image with two textures, triangles and arrows.

```
image: 128 x 256
grid: 6 x 12

texture: rectangle (0,0) - (127,127)
orientation = [0,360]:90
offset = [0,0]:[1,1]
pixels = 255

(-5,-5) - (5,-5)
(5,-5) - (5,5)
(5,5) - (-5,-5)

texture: rectangle (0,127) - (127,255)
orientation = [0,360]:90
offset = [3,5]:[1,1]
pixels = 255

(-5,-5) - (5,-5)
(5,-5) - (5,5)
(5,-5) - (-5,5)
```

It generates a texture with 128 pixels in the vertical (x) direction and 256 pixels in the horisontal (y) direction (defined by the 'image:' line).

The texture elements will be rotated by 0, 90, 180, 270 or 360 degrees. The last texture will have offset from the grid by 0, 1, 2 or 3 pixels in vertical direction and 0, 1, 2, ... 5 pixels from the grid in horizontal direction.

Date created

January 1993

4.4 oddeven

Purpose

Program for generating an odd or an even texture, see: "Visual discrimination of textures with identical third-order statistics", B. Julesz, E. N. Gilbert and J. D. Victors, Biological Cybernetics 31, pp 137-140, 1978

Description

Generates either an 'odd' texture (a texture where the 2x2 pixel blocks have an odd number of black pixels) or an 'even' texture (a texture where the 2x2 pixel blocks have an odd number of black pixels).

Usage

```
oddeven [-s rows<integer>
            [cols<integer>]]
   [-odd | -even]
```

-s rows [columns] Image size

-odd Generate an 'odd' image

-even Generate an 'even' image

Date created

January 1993

4.5 gentex (shell script)

Purpose

Generate various texture mosaics

Description

This program generates the textured images used in testing and evaluation of the different filters. It assumes to find the Brodatz textures on the directory `/textures/brodatz`, with names like `D17.hips`. The program will, for each specified texture, generate one textured image with the name *texture-ID*`.hips` and one reference image with the name *texture-ID*`.ref.hips`, the reference image has pixelvalue one for the all the pixels on the locations of one thexture, two at the next texture and so on.

For most textures, this program adds lines to the description of each output file like "uniform-texture-region: 0 0 127 127", one line per uniform texture region. The numbers indicate upper left and lower right corner of one area for each uniform texture region. The purpose of this is enabling automatic extraction of each texture by programs that calculate i.e. texture-frequency.

Usage

```
gentex [-overwrite] operation
```

-overwrite Overwrite files with the same name as the files the program generates. By default, the program will skip processing if the file of the specified texture already exists.

-all Generate all textures. For a list of all textures, run `gentex` without parameters.

-clean Remove textures and reference frames generated by this program.

texture-ID For a list of all textures, run `gentex -U`

Date created

February 1993

4.6 complvq

Purpose

Compare two LVQ-formatted datafiles, to see what percentage they differ by.

Description

Prints number of equal and nonequal lines to the standard error, and all the nonequal lines are printed to the standard output. Skips comment-lines. Assumes that the lines have a maximum length of 4000.

Warning: this program is not intended to be maximal flexible, but be a quick test of two files, for testing/debugging purposes.

Usage

```
complvq file1 file2
```

Date created

February 1993

4.7 blackpst

Purpose

Compute percentage of black pixels (pixelvalue 0) in an image.

Usage

```
blackpst [imagefile]
```

Date created

February 1993

4.8 accur

Purpose

Compute the classification accuracy of a HIPS classification image.

Description

Accepts two files as input, the correct classification and the classification results. Computes the number of differing pixels.

Usage

```
accur [-perc] imagefile1 [<] imagefile2
```

-perc Output percentage, not absolute values

Date created

April 1993

4.9 dscale

Purpose

Scale and convert an image from "qmf -d ..." to byte, giving each subband maximum range.

Usage

```
dscale
   [-sn struct<integer> |
    -s struct<string> |
    -sb struct<string>]
   [imagefile]
```

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

Date created

2/3-94

4.10 kmeans_classify (shell script)

Purpose

Classify with the K-means algorithm

Description

Converts dataformats and calls the K-means algorithm of the Khoros package.

Usage

```
kmeans_classify [-dec]
    [-sn structure number <integer>]
    [-ff feature file <filename>]
    [-cf classified file <filename>]
    [-sf statistics file <filename>]
    [-classes number of classes <int>]
```

-U Print usage information

-sn filter structure number

-dec decimate the filter output?

-ff feature file (HIPS image (sequence))

-cf classified (output) file (HIPS image)

-sf output statistics file

-classes number of classes (default 2)

Date created

March 1994

4.11 lvq_classify (shell script)

Purpose

Classify with the LVQ (Learning Vector Quantizing)

Description

For large images, run:"

```
setenv TMPDIR <dir>"
```

first, where `<dir>` is a directory with plenty of free space."

Usage

```
lvq_classify [-dec]
    [-sn structure number <integer>]
    [-ff feature file <filename>]
    [-tf tag file <filename>]
    [-cf classified file <filename>]
    [-sf statistics file <filename>]
    [-perc percent of features for
           training <float>]
    [-noc number of codebook vectors
          <integer>]
    [-rlen run length <integer>]
```

-U Print usage information

-sn filter structure number

-dec decimate the filter output?

-ff feature file (HIPS image (sequence))

-tf tag file (HIPS image)

-cf classified (output) file (HIPS image)

-sf output statistics file

-perc percentage of feature vectors to use for training

-noc number of codebook vectors

-rlen run length

Date created

March 1994

4.12 masking

Purpose

Use the first file to mask out regions of the second file, write result to the standard output.

Description

Displays the masked image with a menu of options. Several save options exist, which will print the masked file to the standard output.

Usage

```
masking <maskfile> <image>
```

Date created

March 1994

4.13 mindec

Purpose

Return the least common decimation factor for a filter structure

Usage

```
mindec
    [-sn struct<integer> |
     -s struct<string> |
     -sb struct<string>]
```

-s Filter structure string, defined as a quad tree. See the module *filtertree* (CompileFiltStructTree) for a description of the structure.

-sb Filterstructure string, defined as a binary tree. See the module *filtertree* (CompileFiltStructTreeFromBin) for a description of the structure.

-sn Filter structure number, between 1 and 11 (default 4). See the module *filtertree* (BuildFiltStructTree) for a description of the structure.

Date created

4/3-94

4.14 size

Purpose

Extract the size of a sequence, and print it to stdout

Usage

```
size [-frames | -rows | -cols] [imagefile]
```

-frames Output number of frames

-rows Output number of rows

-cols Output number of columns

Date created

5 February, 1994

4.15 som_classify (shell script)

Purpose

Classify with the SOM (Self Organizing Map)

Description

Applies two passes of the Self Organizing Map clustering algorithm For large images, run:"

```
setenv TMPDIR <dir>"
```

first, where `<dir>` is a directory with plenty of free space."

Usage

```
som_classify [-dec]
    [-sn structure number <int>]
    [-ff feature file <filename>]
    [-cf classified file <filename>]
    [-sf statistics file <filename>]
    [-rlen runlengths <int> <int>]
    [-alpha alphas <int> <int>]
    [-radius radies <int> <int>]
    [-classes number of classes <int>]
```

-U Print usage information

-sn filter structure number

-dec decimate the filter output?

-ff feature file (HIPS image (sequence))

-cf classified (output) file (HIPS image)

-sf output statistics file

-rlen run lengths of the two passes

-alpha α parameter to SOM of the two passes

-radius radius of neighborhood of SOM of the two passes

-classes number of classes (default 2)

Date created

March 1994

4.16 statsnorm

Purpose

Normalize an image according to its statistics, i.e. set the mean to zero and the variance to one.

Description

The output will be float.

Usage

```
statsnorm [imagefile]
```

Data formats handled directly

- float

Date created

June 1993

4.17 winstat

Purpose

Compute the local statistics (variance or standard deviation) of an image.

Description

Usage

```
winstat [-w rows<integer> [cols<integer>]]
    [-lm | -gm | -nm]
    [-var | -sd] [imagefile]
```

-w Window size

-lm Use local mean when computing the variance (default)

-gm use global mean...

-nm Don't use mean (assume that the mean is zero)

-var Calculate windowed variance (default)

-sd Calculate windowed standard deviation

Date created

June 1993

Chapter 5

Other utility programs

5.1 inspec2bibtex

Purpose

Convert inspec export data to bibtex format

Description

Read inspec data from standard in and write bibtex data to standard out.

Usage

```
inspec2bibtex < inspec-file > bibtex-file
```

Date created

30 January 1994

5.2 tempnam

Purpose

Provide a filename which may be created without overwriting any other file – for temporary file creation.

Description

Usage

```
tempnam
```

Date created

1 March, 1994

Part II

Program libraries

Chapter 6

Filter modules

6.1 analysis

Purpose

Fullrate analysis tree-structured filter bank.

Description

analysis-bank

Fullrate analysis tree-structured filter bank.

Usage

analysis-bank

```
analysis_bank (y,ft,nr,nc)
     float **y;
     FilterTree ft;
     int nr,nc;
```

- Input image
- Filter structure tree
- Number of rows
- Number of columns

Date created

March 1993

6.2 dec_analysis

Purpose

Decimated analysis filter bank

Description

dec_analysis_bank

Apply a decimated analysis filterbank to the input array/image and write the result to the standard output

Usage

dec_analysis_bank

```
dec_analysis_bank(hd,y,ft,frame_no)
     struct header *hd;
     float **y;
     FilterTree ft;
     int frame_no;
```

- Input image header, floating point image
- Array-reference to the same image with the pixels accessible as an array: y[m][n]. See the module matrix2.
- Filterstructure tree, type FilterTree, see module *filtertree*.
- Frame number of input/output frame.

Date created

March 1993

6.3 dec_synthesis

Purpose

Synthesis filterbank, to be appled to a image filtered by the decimated analysis filterbank `dec-analysis` or a critically decimated image filtered by the filterbank `analysis`.

Description

dec_synthesis_bank

Apply a decimated synthesis filterbank to the input array/image and write the result to the standard output

Usage

dec_synthesis_bank

```
dec_synthesis_bank(hd,y,ft,frame_no)
     struct header *hd;
```

```
        float **y;
        FilterTree ft;
        int frame_no;
```

- Input image header, floating point image
- Array-reference to the same image with the pixels accessible as an array: y[m][n]. See the module matrix2.
- Filterstructure tree see module *filtertree*.
- Frame number of input/output frame

Date created

March 1993

6.4 filter_2x2bnd

Purpose

General filtering by QMF (Quadrature Mirror Filters)

Description

filter_2x2bnd

Filter the input image array by a filter given by a filter identifier

Usage

filter_2x2bnd

```
 filter_2x2bnd(filt_id,img,x_dim,y_dim,
        ro,kappa,N,analysis)
        char *filt_id;
        float **img;
        int x_dim, y_dim;
        int ro, kappa;
        int N;
        int analysis;
```

- Filter-identifier, a character string like i.e. 'f16b', 'i_2_1_smpl' etc. Note that a filter identifier starting with 'f' is assumed to represent a FIR filter with the sum of the coeffisients equal to 1.0, an identifier starting with 'i' is a polyphase-allpass IIR filter and an identifier starting with 'w' is a wavelet FIR filter. See: John Haakon Husoey, PhD dissertation, 1990, Norwegian Institute of Technology
- Image array, accessible as a float array x[m][n]
- x (m - vertical) dimension, number of pixels
- y (n - horizontal) dimension, number of pixels
- Vertical subband, ro, 0 or 1; 0 is lowpass
- Horizontal subband, kappa, 0 or 1
- N: Filter by $F(z^N)$.
- A boolean flag indicating wheter this is an analysis bank (TRUE) or a synthesis bank (FALSE)

Date created

March 1993

6.5 filtertree

Purpose

This library provides a general way of specifying the subband-split structure of a herarchical tree-structured QMF filterbank.

Description

Each node in the type FilterTree represents one set of QMF filters, splitting the input in a LL, LH, HL and LL band.

BuildFiltStructTree

Build a filter structure quad tree based on a filter-structure number

CompileFiltStructTree

Convert a string of parentheses and commas to a filterstructure tree

CompileFiltStructTreeFromBin

Convert a string of parentheses and commas to a filterstructure tree. The input string is representing a binary tree, and the conversion binary to quad is done by setting "((,),(,))"="((,,,),(,,,),(,,,),(,,,))", "((,),)"="((,,,),,,)" and "(,(,))"="(,,,(,,,))".

getNSubbands

Get the number of subands below the current tree node

getNSubLevels

Get the number of filterlevels below the current tree node as the input to CompileFiltStructTree based on a filtertree.

Usage

BuildFiltStructTree

```
FilterTree
BuildFiltStructTree (struct_no,filter)
     int struct_no;
     char *filter;
```

- Structure number, integer between 1 and 11
- Filter-id string, filter to use on all nodes of the filtertree

CompileFiltStructTree

```
FilterTree
CompileFiltStructTree (structStr,filter)
     char *structStr;
     char *filter;
```

- Structure description string. Format: [filter-id]([LL-band split],[LH-band split], [HL-band split],[HH-band split]) Examples:

```
((,,,),,,)
f8a(,(,,,),f32d(,,,),)
```

- Filter-id string, filter to use on all nodes of the filtertree where the structure description string doesn't specify a different filter.

CompileFiltStructTreeFromBin

```
FilterTree
CompileFiltStructTreeFromBin (structStr,
     filter)
     char *structStr;
     char *filter;
```

- Structure description string. Format: [filter-id]([L-band split],[H-band split]) Examples:

```
((,),)
((,),((,),(,)))
((,),)
```

- Filter-id string, filter to use on all nodes of the filtertree where the structure description string doesn't specify a different filter.

getNSubbands

```
int
getNSubbands (ft)
     FilterTree ft;
```

- Filter tree node

getNSubLevels

```
getNSubLevels (ft)
     FilterTree ft;
```

- Filter tree node

Date created

April 1993

6.6 matrix2

Purpose

Make a HIPS-2 image accessible as a 2D matrix

Description

Assigns a column vector with pointes to corresponding lines in a HIPS-2 image structure. Assigning

```
y=(float **) matrix2(&hd);
```

will make the pixels of the image associated with hd accessible as

```
y[i][j]
```

The function automatically detects the data type of the header and allocates space according to this.

matrix2

Make only the region of interest (see the HIPS-2 documentation for setting the region of interest) accessible as the array.

matrix2-all

Make the whole image available as an array

Usage

matrix2

```
void **
matrix2 (hd)
     struct header *hd;
```

- Image header

matrix2_all

```
void **
matrix2_all (hd)
     struct header *hd;
```

- Image header

Date created

March 1993

6.7 qmf_util

Purpose

Utility routines for QMF filters

Description

Functions with only minor or no modifications from the functions written by John Haakon Husoey of the same name. Note that although the function may be the same, it may refer to structures that has been altered (in `struct-defs.h`).

extr_subband_line

insert_subband_line

allpass_c

read_filtfile

vector

ivector

dvector

matrix

matrix1_f

matrix2_f

matrix4_f

matrix2_uch

dmatrix

imatrix

free_vector

free_ivector

free_dvector

free_matrix

free_dmatrix

free_imatrix

free_submatrix

submatrix

convert_matrix

free_convert_matrix

New functions of fuctions that has been completely rewritten:

read_fir

Read the FIR filter-taps from a file. The first number in the filterfile indicates number of taps. If the number of taps is negative, then the filter is assumed to be non-symmetric. For symmetric filters, the filtertap file begins with the center tap, and only has half the tap-file. This function will search for the filter specification in the directory given in the environmentvariable FILT_DIR.

read_iir

Read the IIR filter-taps from a file This function will search for the filter specification in the directory given in the environmentvariable FILT_DIR.

firfilt_p

FIR filter with periodic extension.

firfilt_s

FIR filter with symmetric extension. NB! this filter will not work properly if the input from the level above is not subsampled.

free_matrix2_f

Release memory occupied by `matrix2_f`.

Usage

read_fir

```
read_fir(filt_par,filter_id)
     struct fir_filterspec *filt_par;
     char *filter_id;
```

- Filterparameter pointer.
- Filteridentifier.

read_iir

```
read_iir(filt_par,filter_id)
     struct filterspec *filt_par;
     char *filter_id;
```

- Filterparameter pointer.
- Filteridentifier.

firfilt_s

```
firfilt_s(filt_par,cur_branch,x,y,
     len,ana_syn)
     struct fir_filterspec *filt_par;
     int cur_branch;
     float *x, *y;
```

```
    int len;
    Boolean ana_syn;
```

- Filterparameters.
- Current branch: 0 or 1; lowpass or highpass.
- Input data vector.
- Output data vector.
- Vector length.
- Flag indicating wheter this is an synthesis bank (TRUE), or analysis (FALSE).

firfilt_p

```
firfilt_p(fil_par,cur_branch,x,y,len)
    struct fir_filterspec *fil_par;
    int     cur_branch;
    float   *x, *y;
    int     len;
```

- Filterparameters.
- Current branch: 0 or 1; lowpass or highpass.
- Input data vector.
- Output data vector.
- Vector length.

free_matrix2_f

```
free_matrix2_f(n1,n2,ptr)
    int n1, n2;
    float **ptr;
```

- Vertical dimension.
- Horizontal dimension.
- Pointer to the matrix.

Date created

March 1993

6.8 even_gabor

Purpose

Create the frequency response of an even-symmetric Gabor filter

Description

even_gabor

Create the frequency response of an even symmetric Gabor filter

Usage

even_gabor

```
even_gabor (hd, U, SigmaU, SigmaV, theta)
  struct header *hd;
  double U,SigmaU,SigmaV,theta;
```

- Image header
- Centre frequency U of the Gabor function
- σ_U of the Gabor function
- σ_V of the Gabor function
- Rotation angle, θ

Date created

January 1993

6.9 gen_gabor

Purpose

Create the frequency response of a general Gabor filter

Description

gen_gabor

Create the frequency response of a general Gabor filter

Usage

gen_gabor

```
gen_gabor (hd, U0, V0, SigmaX, SigmaY,
  struct header *hd;
  double U0,V0,SigmaX,SigmaY,theta;
```

- Image header
- Centre frequency U of the Gabor function
- Centre frequency V of the Gabor function
- σ_X of the Gabor function
- σ_Y of the Gabor function
- Rotation angle θ

Date created

February 1993

6.10 mcf_gauss

Purpose

Gaussian lowpass filter.

Description

mcf_gauss

Applies a gaussian lowpass filter to the input. The filtering is performed in the Fourier domain, and this function therefore applies both forward and reverse FFT

Usage

mcf_gauss

```
mcf_gauss(hdi,hdo,sigma)
  struct header *hdi, *hdo;
  double sigma;
```

- Input image (complex)
- Output image (complex)
- σ, parameter to the Gaussian

Date created

January 1993

6.11 mcf_invfourtr

Purpose

Inverse fourier tranform with scaling so that the maximum absolute value is 1.0

Description

mcf_invfourtr

Inverse fourier tranform with scaling so that the maximum absolute value is 1.0

Usage

mcf_invfourtr

```
mcf_invfourtr (hd)
  struct header *hd;
```

- Image header

Date created

January 1993

6.12 mcf_tanh_nonlin

Purpose

Apply the tanh(..) nonlinearity described in F. Farrokhnia's PhD dissertation to an image

Description

mcf_tanh_nonlin

Function applying the tanh(..) nonliarity

Usage

mcf_tanh_nonlin

```
mcf_tanh_nonlin(hdi,hdo,alpha)
  struct header *hdi, *hdo;
  double alpha;
```

- Input image header.
- Output image header.
- alpha: parameter to the nonlinear transform.

Date created

January 1993

6.13 mcf_util

Purpose

Various utility routines for multichannel filtering

Description

parseUV

Convert the commandline-list for U0 or V0 to a numeric list.

parseTheta

Convert the commandline-list for theta to a numeric list

read_ft

Read in the input image and convert it to its fourier transformed counterpart.

mcf_write_image

Transform the frame into the maximum range of the output format, and write the frame

max_range

Linear scale an image into the maximal range for the specified format.

scale

Linear scale an image to the range defined by the calling parameters.

Usage

parseUV

```
parseUV (hd,uv, uv_num, genDef)
     struct header *hd;
     Listarg *uv;
     double ***uv_num;
     Boolean genDef;
```

- Input image header (to determine number of rows and columns)
- List of U0 or V0 strings — input.
- List of numeric U0 or V0 — output.
- Flag indicating wheter to generate the defalt values (for U0 only)

parseTheta

```
parseTheta (theta, theta_num)
     Listarg *theta;
     double ***theta_num;
```

- List of theta — input
- List of numeric theta — output

read_ft

```
read_ft(hd, hdft, method, filename)
     struct header *hd;
     struct header *hdft;
     int *method;
     Filename filename;
```

- Input format header (will be kept in order to convert back)
- Input header, converted to appropriate format and Fourier transformed
- Conversion method
- Input filename

mcf_write_image

```
mcf_write_image(hd,hdo,fr,MaxRange)
     struct header *hd;
     struct header *hdo;
     int fr;
     Boolean MaxRange;
```

- Input format header (the same format as the output will be stored as)
- Output header
- Frame number
- Flag, when TRUE, convert the output written image to the maximum dynamic range of the output format. This flag will normally be FALSE for floating point images, and TRUE for fixed point images, such as byte images.

max_range

```
max_range(hdi,hdo,format)
     struct header *hdi,*hdo;
     int format;
```

- Input image header
- Output image header
- Pixel format

scale

```
scale (hdi,hdo,n_min,n_max)
     struct header *hdi,*hdo;
     float n_min,n_max;
```

- Input image header
- Output image header
- Minimal value of the output
- Maximal value of the output

Date created

January 1993

6.14 one_channel_filter

Purpose

One channel filtering with some preprocessing options

Description

one_channel_filter

- Filter with frequency and orientation selective properties.
- Some kind of linear transform on the image pixels.
- Smmothing (blurring) of the filtered image by a Gaussian lowpass filter.

Usage

one_channel_filter

```
one_channel_filter (hdi,hdo,U0, V0,
            theta, Sigma1, Sigma2,
            SigmaLP, alpha, filter,
            nonlin, blurr, twoangle)
     struct header *hdi, *hdo;
     double U0, V0, theta, Sigma1,
            Sigma2, SigmaLP, alpha;
     int filter, nonlin, blurr;
     Boolean twoangle;
```

- Input image
- Output image
- Vert. centre frequency
- Horz. centre frequency
- Rotation of filter
- Vert. sigma for filter
- Horz. sigma for filter
- Sigma for low-pass 'feature' filter
- Alpha in tanh(..) nonlinearity
- Filter type identifier (NO-FILTER, GENERAL-GABOR or EVEN-GABOR)
- Nonlinearity identifier (NO-NONLIN or TANH-NONLIN)
- Blurring identifier (blurring=lowpass filtering) (NO-BLURR or BLURR)
- Use both positive and negative angle? If true, then the frequeny responses of both the Gabor filter with +theta angle and theta+90 degrees angles will be added to form one frequency response of the filter. Remember that filtering is performed by multiplication in the Fourier domain.

Date created

January 1993

6.15 win_fourtr

Purpose

Code for computing the windowed Fourier-transform of an image.

Description

win_fourtr

Compute the windowed Fourier-transform of an image. Assumes that the input image is complex (`h-complex`) formatted.

Usage

win_fourtr

```
win_fourtr(hdi,hdo,s,sc,SigmaX,SigmaY)
     struct header *hdi, *hdo;
     Coord *s;
     int sc;
     double SigmaX, SigmaY;
```

- Input image.
- Output image.
- List of window centre corrdinates.
- Index in list of window centre corrdinates.
- Sigma X: parameter to the Gaussian window.
- Sigma Y.

Date created

March 1993

6.16 struct_gabor

Purpose

Filter the input image by a even-symmetric Gabor filter with the filter structure determined by a quadfilertree descriptor.

Usage

struct_gabor

```
struct_gabor(ft,hdi,hd,method,sf,alpha,
     nonlin,blurr)
     FilterTree ft;
     struct header *hdi;
     struct header *hd;
     int method;
     double sf;
     double alpha;
```

```
int nonlin;
int blurr;
```

- Filter structure tree
- Input (nonfiltered) header
- Input format (nonconverted) header
- Conversion method
- Factor to sigma in lowpass filter
- Alpha in nonlinear transform
- Nonlinear transform method
- Smoothing method

Date created

April 1993

Chapter 7

General libraries

7.1 error

Purpose

Error handler

Usage

error_handler

```
error_handler(errcode,str)
     int errcode;
     char *str;
```

- Error-code, see `error.h`.
- Error message string

Date created

February 1993

7.2 io

Purpose

Routine for writing to the user console.

Description

printPstFin

Print the percentage of completion of a process. The percentage written last will be deleted and the current will be printed on the same location. The number printed will be counter/max*100

Usage

printPstFin

```
printPstFin (msg,counter,max)
     char *msg;
     int counter;
     int max;
```

- Message to the user
- Counter
- Maximum counter value, counter = max $\Rightarrow$ 100

Date created

March 1993

7.3 mktex_imout

Purpose

Output texture generated by mktex in HIPS-2 format

Description

OutputImage

Output texture generated by mktex in HIPS-2 format

Usage

OutputImage

```
OutputImage(image,xsize,ysize,argc,argv)
     int *image;
     int ysize,xsize;
     int argc;
     char **argv;
```

- Image array.
- Vertical (x) dimension (number of pixels).
- Horizontal (y) dimension.
- argc (number of commandline arguments).
- argv (commandline argument vector).

Date created

January 1993

7.4 set_vars

Purose

A common script (common code) for all shell scripts that sets variables with global information, like information on what directory to find what file etc.

Description

This script sets variables for the calling script. Changing the directory structure means changing the variable values set by this program. This script also defines lists of textures and various other global variables.

Usage

The program is called by:

```
. set_vars
```

Date created

February 1993

7.5 k-means-rsh

Purpose

Classify an image sequence by the K-means algorithm in a remote shell Used as a subroutine to kmeans-classify

Usage

```
k-means-rsh [-ff feature file <filename>]
    [-cf classified file <filename>]
    [-sf statistics file <filename>]
    [-classes number of classes <integer>]
```

-U Print usage information

-sn filter structure number

-dec decimate the filter output?

-ff feature file (Khoros image (sequence))

-cf classified (output) file (Khoros image)

-sf output statistics file

-classes number of classes (default 2)

Date created

March 1994