libbttc

1.0

Generated by Doxygen 1.6.3

Fri Mar 23 14:51:59 2012

Contents

Chapter 1

libbttc doxygen documentation

Author

Edgar Simo-Serra <esimo@iri.upc.edu>

Version

1.0

Date

March 2012

1.1 License

1.2 Overview

This library calculates the faces obtained by B-Tree Triangular Coding (BTTC). This is usually for subdividing an image into a triangular mesh. The core library is written in C but an octave/matlab interface is provided.

The main focus of this library is simplicity. The code is simple enough to directly integrate it into another program as it is a single C code file with no external dependencies.

1.3 Installation

To compile the C interface to the library it is sufficient to run:

```
make # compiles the library
```

To install the C interface you additionally must run (as superuser):

```
make install # installs by default to /usr (edit Makefile to change)
```

The octave/matlab interfaces are separate from the C library and can respectively be compiled by:

```
make octave # compiles octave interface
make matlab # compiles matlab interface
```

1.4 Example

This example uses the matlab interface and operates on the classic lena image, which should be 512x512 and grayscale. The matlab function accepts two parameters, first is the image and second is the threshold to use. It can output the result in two different ways. With a single output value it will output all the x,y coordinates of the triangle corners. With two values it will output first a list of the vertices and secondly the IDs of the vertices which correspond to the corners of the triangle faces. The example should be more explanatory.

```
lena  = imread( 'lena512.bmp' ); % Load the image
% Image must be doubles of size must be 2^m-1
dlena = imresize( double(lena), [ 513, 513 ], 'bilinear' );

% First we shall do it with one output parameter
faces = bttc_m( dlena, 60 ); % Image is dlena with a threshold of 60
figure;
imshow( dlena ./ 255 ); % Display the image
hold on;
% Draw all the face lines one by one
for i=1:size(faces,1)
   f = faces(i,:);
   x = [f(:,1) f(:,3) f(:,5) f(:,1)];
   y = [f(:,2) f(:,4) f(:,6) f(:,2)];
   line( x, y, 'LineWidth', 1, 'Color', 'g' );
end
```

```
% Now we can do it with two output parameters
[ Vi, Fi ] = bttc_m( dlena, 60 );
figure;
imshow( dlena ./ 255 ); % Display the image
hold on;
% Draw all the face lines one by one
for i=1:size(Fi,1)
   v = Vi( Fi(i,:), : );
   x = [v(:,1);v(1,1)];
   y = [v(:,2);v(1,2)];
   line( x, y, 'LineWidth', 1, 'Color', 'g' );
end
```

1.5 Changelog

- Version 1.0, March 2012
  - Initial release.

1.6 References

-Distasi, R.; Nappi, M.; Vitulano, S.; "Image compression by B-tree triangular coding," Communications, IEEE Transactions on, vol.45, no.9, pp.1095-1100, Sep 1997

See also

bttc

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 bttc.h File Reference

Functions

- int * bttc (int *n, const double *img, int pitch, int size, double threshold)
  Computes the B-Tree Triangular Coding of an image.

3.1.1 Function Documentation

3.1.1.1 int* bttc (int * *n*, const double * *img*, int *pitch*, int *size*, double *threshold*)

Computes the B-Tree Triangular Coding of an image.

This function returns faces in the form of a long vector grouped into batches of 6.

```
faces = [ batch1 batch2 batch3 ... batchn ]
```

where each batch has the form of

```
batch = [ x0 y0 x1 y1 x2 y2 ]
```

Parameters

→ ***n*** Number of faces found (minimum 2 if no error occurred).

← ***img*** Image to perform the coding on. This image is row-major and has a pitch of 'pitch' between rows.

← ***pitch*** Pitch to shift when looking up rows in the image, should be equal to size if the entire image is being processed.

← ***size*** Must a value in the form of 2^m+1 for m>=1. It indicates the area to process, which goes from [0,size-1] in both x and y directions.

← ***threshold*** Threshold to mesh based on. The range of the threshold depends on the value range of the image and is the limit of error between the linear interpolation of the triangular face and the real value of the pixels.

Returns

NULL on error or the list of faces found on success (with n faces or batches). This value is dynamically allocated and thus must be freed by the user.

Index