Face Detection with Alignment

Generated by Doxygen 1.8.6

Thu May 22 2014 17:35:03

Contents

Chapter 1

Main Page

This project provides a C++ implementation for human face detection with Alignment from unconstrained photos. Below is a brief manual, detailed documentation is generated using doxygen.

Tip All the data structurs and functions are arranged in C style, you can use the *Files* tab to have an overview.

Implemented by Hang Su.

Installation

Checkout latest code

svn checkout http://eharmony-photofeature.googlecode.com/svn/trunk/ eharmony-photofeature-read-only

Dependencies

The code depends on OpenCV to load/dispaly image. To install OpenCV on Ubuntu/Debian:

sudo apt-get install libopencv-dev

A cblas library should also be installed, e.g.

Ubuntu/Debian

sudo apt-get install libblas-dev

Fedora

sudo yum install atlas-sse3-devel.x86_64

Compilation

make all

Usage example

1. #include "eHimage.h"
2. #include "eHfacemodel.h"
3. #include "eHposemodel.h"
4. #include "eHbbox.h"

```
5. #include <vector>
6.
7. int main(int argc, char** argv){
8.     //load face model & body model
9.     facemodel_t* facemodel = facemodel_readFromFile("face_p146.xml");
10.    posemodel_t* posemodel = posemodel_readFromFile("pose_BUFFY.xml");
11.
12.    //load a jpeg image
13.    image_t* img = image_readJPG(argv[1]);
14.
15.    //detect faces and show results
16.    std::vector<bbox_t> faces = facemodel_detect(facemodel,posemodel,img);
17.    image_showDetection(img, faces, "Face Detection Results");
18.
19.    //destruct image and models
20.    image_delete(img);
21.    facemodel_delete(facemodel);
22.    posemodel_delete(posemodel);
23.
24.    return 0;
25. }
```

Image data structure and operation

A structure type *image_t* is defined to present images. Some usesful operations are also provided, e.g. to load a jpeg image from file, use

image_ptr image_readJPG (const char *filename)

To delete an image, use

void image_delete (image_ptr img)

Face detection

Face detection is based on algorithm described in [1]. Pre-trained models are provided in XML format.

To load a face model, use

facemodel_t * facemodel_readFromFile (const char *filepath)

To detect faces, use

vector< bbox_t > facemodel_detect (const facemodel_t *model, const image_ptr img)

A body detection model (described below) can also be combined to help improve detection performace

vector< bbox_t > facemodel_detect (const facemodel_t *facemodel, const posemodel_t *posemodel, const image_ptr img)

Results can be visualized using

void image_showDetection (const image_ptr img, const vector< bbox_t > boxes, const std::string &winname)

or

void image_showFaces (const image_ptr img, const vector< bbox_t > boxes, const std::string &winname)

Finally, a face model can be deleted using

void facemodel_delete (facemodel_t *model)

Body/pose detection

Human body detection is based on algorithm described in [2]. Pre-trained models are provided in XML format.

To load a pose model, use

posemodel_t* posemodel_readFromFile(const char* filepath)

To detect poses, use

vector<bbox_t> posemodel_detect(const posemodel_t* model, const image_ptr img)

Results can be visualized using

void image_showDetection (const image_ptr img, const vector< bbox_t > boxes, const std::string &winname)

Finally, the model can be deleted using

void posemodel_delete(posemodel_t* model)

References

[1] X. Zhu, D. Ramanan. **"Face detection, pose estimation and landmark localization in the wild"** Computer Vision and Pattern Recognition (CVPR) Providence, Rhode Island, June 2012.

[2] Y. Yang, D. Ramanan. **"Articulated Pose Estimation using Flexible Mixtures of Parts"** Computer Vision and Pattern Recognition (CVPR) Colorado Springs, Colorado, June 2011.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 eHbbox Struct Reference

Multi-box "bounding box", used for detection result.

`#include <eHbbox.h>`

Public Attributes

- std::vector< fbox_t > boxes

  part locations
- double score

  detection score
- int component

  component id for certain models
- fbox_t outer

  outer "real" bounding box of the detection
- double area

  area of outer

4.1.1 Detailed Description

Multi-box "bounding box", used for detection result.

4.1.2 Member Data Documentation

4.1.2.1 int eHbbox::component

component id for certain models

Face model has 13 components (id 0∼12) for viewpoints ranging from 90 degree (profile facing right) to -90 degree (profile facing left).

4.1.2.2 double eHbbox::score

detection score

Warning

not calibrated

The documentation for this struct was generated from the following file:

- eHbbox.h

4.2 eHbox_f Struct Reference

box with floating-point boundaries

`#include <eHbox.h>`

Public Attributes

- double x1
  left bound
- double y1
  up bound
- double x2
  right bound
- double y2
  bottom bound

4.2.1 Detailed Description

box with floating-point boundaries

Boxes are defined as tuples (x1, y1, x2, y2), where x1<=x2, y1<=y2; so (x1,y1) is the top-left corner, (x2,y2) is the bottom-right corner.

The documentation for this struct was generated from the following file:

- eHbox.h

4.3 eHbox_i Struct Reference

box with integer boundaries

`#include <eHbox.h>`

Public Attributes

- int x1
  left bound
- int y1
  up bound
- int x2
  right bound
- int y2
  bottom bound

4.3.1 Detailed Description

box with integer boundaries

Boxes are defined as tuples (x1, y1, x2, y2), where x1<=x2, y1<=y2; so (x1,y1) is the top-left corner, (x2,y2) is the bottom-right corner.

The documentation for this struct was generated from the following file:

- eHbox.h

4.4 eHfacemodel Struct Reference

Face detection model.

`#include <eHfacemodel.h>`

Public Attributes

- vector< filter_t > filters
  part filters
- vector< facedef_t > defs
  deformation params
- vector< vector< facepart_t > > components
  part infos
- int maxsize [2]
  XXX.
- int len
  not used
- int interval
  interval of pyramid
- int sbin
  bin for building hog feature
- double delta
  not used
- double thresh
  threshold for detection score
- double obj
  not used

4.4.1 Detailed Description

Face detection model.

4.4.2 Member Data Documentation

4.4.2.1 vector<filter_t> eHfacemodel::filters

part filters

Note

All part filters should be of the same size

The documentation for this struct was generated from the following file:

- eHfacemodel.h

4.5 eHfeatpyramid Struct Reference

Image feature pyramid.

`#include <eHfeatpyramid.h>`

Public Attributes

- mat3d_ptr * feat
  features of each level
- double * scale
  scaled of each level
- int len
  levels of pyra, size of feat & scale
- int interval
  levels within a double-size interval
- int imy
  image height
- int imx
  image width

4.5.1 Detailed Description

Image feature pyramid.

The documentation for this struct was generated from the following file:

- eHfeatpyramid.h

4.6 eHfilter Struct Reference

Image feature filter.

`#include <eHfilter.h>`

Public Attributes

- int i
  filter index, not used
- mat3d_t w
  filter [y,x,f], where (y,x) is location, f is feature index

4.6.1 Detailed Description

Image feature filter.

The documentation for this struct was generated from the following file:

- eHfilter.h

4.7 eHimage Struct Reference

Basic image data structure.

`#include <eHimage.h>`

Public Attributes

- double * data
  pixel value data
- double * ch [3]
  a view into each channel
- size_t sizy
  image height
- size_t sizx
  image width
- size_t nchannel
  number of channels
- int imsize [3]
  [sizy sizx nchanel]
- bool is_shared
  whether share data with a parent image
- size_t stepy
  step between columns
- size_t stepyx
  step between channels

4.7.1 Detailed Description

Basic image data structure.

Note

Using column major (Fortran) style
Data is associated with (double* data), (double* ch[3]) only provide a view into data

The documentation for this struct was generated from the following file:

- eHimage.h

4.8 eHmatrix2d Struct Reference

2D matrix

`#include <eHmatrix.h>`

Public Attributes

- double * vals
  values
- size_t sizy
  matrix height
- size_t sizx
  matrix width

4.8.1 Detailed Description

2D matrix

Note

matrix is stored in column-major style

The documentation for this struct was generated from the following file:

- eHmatrix.h

4.9 eHmatrix3d Struct Reference

3D matrix

`#include <eHmatrix.h>`

Public Attributes

- double * vals
  values
- size_t sizy
  matrix height
- size_t sizx
  matrix width
- size_t sizz
  matrix depth

4.9.1 Detailed Description

3D matrix

Note

matrix is stored in column-row-page order

The documentation for this struct was generated from the following file:

- eHmatrix.h

4.10 eHmatrixkd Struct Reference

k-dimension matrix

`#include <eHmatrix.h>`

Public Attributes

- double * vals
  values
- size_t k
  number of dimensions
- size_t * siz
  sizes along each dimension

4.10.1 Detailed Description

k-dimension matrix

The documentation for this struct was generated from the following file:

- eHmatrix.h

4.11 eHposemodel Struct Reference

Human body/pose mdoel.

`#include <eHposemodel.h>`

Public Attributes

- vector< posebias_t > biases
  bias towards part combinations
- vector< filter_t > filters
  part filters
- vector< posedef_t > defs
  deformation params
- vector< posepart_t > parts
  part configurations
- int maxsize [2]
  XXX.
- int len
  not used
- int interval
  interval of feature pyramid
- int sbin
  bin for building hog feature
- double thresh
  threshold for detection score
- double obj
  not used

4.11.1 Detailed Description

Human body/pose mdoel.

4.11.2 Member Data Documentation

4.11.2.1 vector<filter_t> eHposemodel::filters

part filters

Note

all filters should be of the same size

The documentation for this struct was generated from the following file:

- eHposemodel.h

Chapter 5

File Documentation

5.1 eHbbox.h File Reference

Multi-box bounding box type and operations.

```
#include "eHbox.h"
#include <vector>
```

Classes

- struct eHbbox

  Multi-box "bounding box", used for detection result.

Macros

- #define EH_BBOXS_PRUNE 30000

  Default pruning parameter for bbox_v_nms()

Typedefs

- typedef struct eHbbox bbox_t

  Multi-box "bounding box", used for detection result.

Functions

- void bbox_calcOut (bbox_t *)

  Filling the fields of given bbox: outer, area.
- void bbox_clipboxes (bbox_t &bbox, const int *imsize)

  Clip the boxes to image boundary.
- void bbox_v_resize (std::vector< bbox_t > &bboxes, double scale)

  Resize the input bboxs (in-place)
- void bbox_v_move (std::vector< bbox_t > &bboxes, const int *offset)

  Move the input bboxs (in-place)
- void bbox_v_nms (std::vector< bbox_t > &bboxes, double overlap, unsigned prune=EH_BBOXS_PRUNE)

  Non-maximum suppression.

5.1.1 Detailed Description

Multi-box bounding box type and operations. A "bounding box" is defined as a collection of boxes, which serve as detection results of different parts for part-based detection algorithms.

Author

Hang Su

Date

2012-08-13

5.1.2 Macro Definition Documentation

5.1.2.1 #define EH_BBOXS_PRUNE 30000

Default pruning parameter for bbox_v_nms()

See Also

bbox_v_nms()

5.1.3 Function Documentation

5.1.3.1 void bbox_clipboxes (bbox_t & *bbox,* const int * *imsize*)

Clip the boxes to image boundary.

Parameters

imsize	imsize[0]=height, imsize[1]=width

See Also

fbox_clip()

5.1.3.2 void bbox_v_move (std::vector< bbox_t > & *bboxes,* const int * *offset*)

Move the input bboxs (in-place)

See Also

fbox_move()

5.1.3.3 void bbox_v_nms (std::vector< bbox_t > & *bboxes,* double *overlap,* unsigned *prune* = EH_BBOXS_PRUNE)

Non-maximum suppression.

Greedily select high-scoring detections and skip detections that are significantly coverd by a previously selected detection.

Parameters

bboxes	an array of detection results, changed inside function
overlap	two results are not both kept if their overlap ratio exceed this value
prune	initial bboxes are pruned for higher speed

5.1.3.4 void bbox_v_resize (std::vector< bbox_t > & *bboxes*, double *scale*)

Resize the input bboxs (in-place)

See Also

fbox_resize()

5.2 eHbox.h File Reference

Box types and operations.

```
#include <stdlib.h>
#include <vector>
```

Classes

- struct eHbox_f

  box with floating-point boundaries
- struct eHbox_i

  box with integer boundaries

Typedefs

- typedef struct eHbox_f fbox_t

  box with floating-point boundaries
- typedef struct eHbox_i ibox_t

  box with integer boundaries

Functions

- void fbox_set (fbox_t *box, double x1, double y1, double x2, double y2)

  Set values for given box.
- ibox_t fbox_getibox (fbox_t *box)

  Round floating-point box to intege box.
- void ibox_set (ibox_t *, int x1, int y1, int x2, int y2)

  Set values for given box.
- double fbox_interArea (const fbox_t &box1, const fbox_t &box2)

  Compute area of intersection between two boxes.
- void fbox_clip (fbox_t &box, const int *imsize)

  Clip box so it's limited by the image size.
- fbox_t fbox_merge (const vector< fbox_t > &boxes, const int *idxs, int num, const double *padding=NULL)

  Merge boxes, resulting box can be optionally padded.
- fbox_t fbox_getResized (const fbox_t &box, double scale)

Get a resized box, input box is not changed.

- fbox_t fbox_getMoved (const fbox_t &box, const int *offset)

  Get a moved box, input box is not changed.

- void fbox_resize (fbox_t *box, double scale)

  Resize the input box.

- void fbox_move (fbox_t *box, const int *offset)

  Move the position of input box by given offset.

5.2.1 Detailed Description

Box types and operations. Boxes are defined as tuples (x1, y1, x2, y2), where x1<=x2, y1<=y2; so (x1,y1) is the top-left corner, (x2,y2) is the bottom-right corner.

Author

Hang Su

Date

2012-08-13

5.2.2 Function Documentation

5.2.2.1 fbox_t fbox_getMoved (const fbox_t & *box*, const int * *offset*)

Get a moved box, input box is not changed.

See Also

fbox_move()

5.2.2.2 fbox_t fbox_getResized (const fbox_t & *box*, double *scale*)

Get a resized box, input box is not changed.

See Also

fbox_resize()

5.2.2.3 fbox_t fbox_merge (const vector< fbox_t > & *boxes*, const int * *idxs*, int *num*, const double * *padding* = NULL)

Merge boxes, resulting box can be optionally padded.

Parameters

boxes	array of boxes to be merged
idxs	indexs within vector, indicating which boxes to merge
num	length of idxs
padding	[l r u d] as ratio of corresponding edge, if NULL is passed in, no padding is performed

Returns

merged box

5.2.2.4 void fbox_move (fbox_t * box, const int * offset)

Move the position of input box by given offset.

See Also

fbox_getMoved()

5.2.2.5 void fbox_resize (fbox_t * box, double scale)

Resize the input box.

See Also

fbox_getResized()

5.3 eHfacemodel.h File Reference

face detection model and operations

```
#include <vector>
#include "eHimage.h"
#include "eHfilter.h"
#include "eHbbox.h"
#include "eHposemodel.h"
```

Classes

- struct eHfacemodel

  Face detection model.

Typedefs

- typedef struct deformation_face **facedef_t**
- typedef struct part_face **facepart_t**
- typedef struct eHfacemodel facemodel_t

  Face detection model.

Functions

- facemodel_t * facemodel_parseXml (char *xmlstr)

  Parse face model from xml style string.
- facemodel_t * facemodel_readFromFile (const char *filepath)

  Read face model from file.
- vector< bbox_t > facemodel_detect (const facemodel_t *model, const image_ptr img, double thrs)

  Perform face detection.
- vector< bbox_t > facemodel_detect (const facemodel_t *model, const image_ptr img)

  Perform face detection using threshold inside model.
- vector< bbox_t > facemodel_detect (const facemodel_t *facemodel, const posemodel_t *posemodel, const image_ptr img, double thrs_face, double thrs_pose)

  Perform face detection with help of body detection.

- vector< bbox_t > facemodel_detect (const facemodel_t *facemodel, const posemodel_t *posemodel, const image_ptr img)

  Perform face detection with help of body detection using thresholds inside models.
- void facemodel_delete (facemodel_t *model)

  Delete a face model, release related memory.

5.3.1 Detailed Description

face detection model and operations

See Also

Xiangzin Zhu, Deva Ramanan, "Face Detection, Pose Estimation, and landmark Localization in the Wild". In CVPR 2012.

Author

Hang Su

Date

2012-08-13

5.3.2 Function Documentation

5.3.2.1 vector<bbox_t> facemodel_detect (const facemodel_t * *model,* const image_ptr *img,* double *thrs*)

Perform face detection.

Parameters

model	face detection model
img	where to find faces from
thrs	threshold used for pruning results

Returns

array of detected faces (together with part locations)

5.3.2.2 vector<bbox_t> facemodel_detect (const facemodel_t * *model,* const image_ptr *img*)

Perform face detection using threshold inside model.

Parameters

model	face detection model
img	where to find faces from

Returns

array of detected faces (together with part locations)

5.3.2.3 vector<bbox_t> facemodel_detect (const facemodel_t * *facemodel,* const posemodel_t * *posemodel,* const image_ptr *img,* double *thrs_face,* double *thrs_pose*)

Perform face detection with help of body detection.

Parameters

facemodel	face detection model
posemodel	body pose detection model
img	where to find faces from
thrs_face	threshold used for pruning face detections
thrs_pose	threshold used for pruning pose detections

Returns

array of detected faces (together with part locations)

5.3.2.4 vector<bbox_t> facemodel_detect (const facemodel_t * *facemodel*, const posemodel_t * *posemodel*, const image_ptr *img*)

Perform face detection with help of body detection using thresholds inside models.

Parameters

facemodel	face detection model
posemodel	body pose detection model
img	where to find faces from

Returns

array of detected faces (together with part locations)

5.3.2.5 facemodel_t* facemodel_parseXml (char * *xmlstr*)

Parse face model from xml style string.

Note

xmlstr is modified during parsing, this can be avoided by using Non-Destrutive Mode of rapidxml

5.3.2.6 facemodel_t* facemodel_readFromFile (const char * *filepath*)

Read face model from file.

See Also

facemodel_parseXml()

5.4 eHfeatpyramid.h File Reference

Feature pyramid data type and calculation.

```
#include "eHmatrix.h"
#include "eHimage.h"
```

Classes

- struct eHfeatpyramid

  Image feature pyramid.

Typedefs

- typedef struct eHfeatpyramid featpyra_t

  Pointer to a feature pyramid.

Functions

- featpyra_t * featpyra_create (const image_ptr im, int interval, int sbin, const int *maxsize, bool hallucinate=true)

  Allocate and compute a feature pyramid from an image.
- void featpyra_delete (featpyra_t *pyra)

  Delete a feature pyramid.

5.4.1 Detailed Description

Feature pyramid data type and calculation.

Author

Hang Su

Date

2012-08-14

5.4.2 Function Documentation

5.4.2.1 featpyra_t* featpyra_create (const image_ptr *im*, int *interval*, int *sbin*, const int * *maxsize*, bool *hallucinate* = `true`)

Allocate and compute a feature pyramid from an image.

Parameters

hallucinate	whether hallucinate a higher resolution interval

5.5 eHfilter.h File Reference

Filters applied on image features.

```
#include "eHmatrix.h"
#include <vector>
```

Classes

- struct eHfilter

  Image feature filter.

Typedefs

- typedef struct eHfilter filter_t

  Image feature filter.

Functions

- mat3d_ptr filterv_apply (const std::vector< filter_t > filters, const mat3d_ptr feats, int start, int end)

  Convolve a feature map with a set of filters - Multithreaded version.
- mat3d_ptr filterv_apply_ST (const std::vector< filter_t > filters, const mat3d_ptr feats, int start, int end)

  Convolve a feature map with a set of filters - Singlethreaded version.

5.5.1 Detailed Description

Filters applied on image features.

Author

Hang Su

Date

2012-08

5.5.2 Function Documentation

5.5.2.1 mat3d_ptr filterv_apply (const std::vector< filter_t > *filters*, const mat3d_ptr *feats*, int *start*, int *end*)

Convolve a feature map with a set of filters - Multithreaded version.

Parameters

filters	a set of part filters
feats	feature map
start	range of filters used - first one
end	range of filters used - last one

Returns

filter responses

Note

filter responses is allocated inside, proper delete is necessary after use
cblas library is required

See Also

filterv_apply_ST()

5.5.2.2 mat3d_ptr filterv_apply_ST (const std::vector< filter_t > *filters*, const mat3d_ptr *feats*, int *start*, int *end*)

Convolve a feature map with a set of filters - Singlethreaded version.

Parameters

filters	a set of part filters
feats	feature map
start	range of filters used - first one
end	range of filters used - last one

Returns

filter responses

Note

filter responses is allocated inside, proper delete is necessary after use

See Also

filterv_apply()

5.6 eHimage.h File Reference

Basic image type and operations.

```
#include <stdlib.h>
#include <string>
#include "eHbox.h"
#include "eHbbox.h"
```

Classes

- struct eHimage

  Basic image data structure.

Typedefs

- typedef struct eHimage image_t

  Basic image data structure.
- typedef image_t * image_ptr

  Pointer to image.

Functions

- image_ptr image_alloc (size_t sizy, size_t sizx, size_t nch=3)

  Allocate a new image of size [sizy, sizx, nch].
- image_ptr image_alloc (size_t sizy, size_t sizx, size_t nch, const double *fillval)

  Allocate a new image of size [sizy, sizx, nch], and initialize all pixel values to fill.
- void image_delete (image_ptr img)

  Delete image and associated memory.
- void image_fill (image_ptr img, const double *val)

  Fill all pixels with same values.
- image_ptr image_readJPG (const char *filename)

  Read Jpeg image file.
- void image_display (const image_ptr img, const std::string &winname)

Display an image.

- image_ptr image_subsample (const image_ptr img, double scale)

  Fast image subsampling.
- image_ptr image_reduce (const image_ptr img)

  Get an image half the size of input one.
- image_ptr image_resize (const image_ptr img, double scale)

  Resize an image using bilateral interpolation.
- image_ptr image_crop (const image_ptr img, fbox_t crop, int *offset=NULL, bool shared=true)

  Crop image This function can be used in two ways, either get shared data from original image, or allocate a new image, which is more expensive.
- void image_showDetection (const image_ptr img, const vector< bbox_t > boxes, const std::string &winname)

  Show detection results on image and wait.
- void image_showFaces (const image_ptr img, const vector< bbox_t > boxes, const std::string &winname)

  Show face detection results: face region, eyes, nose, mouth.

5.6.1 Detailed Description

Basic image type and operations.

Note

images are stored using column major style

Author

Hang Su

Date

2012-08-13

5.6.2 Function Documentation

5.6.2.1 image_ptr image_alloc (size_t *sizy*, size_t *sizx*, size_t *nch* = 3)

Allocate a new image of size [sizy, sizx, nch].

Returns

pointer to the allocated image

Note

Returned image is not initialized

5.6.2.2 image_ptr image_alloc (size_t *sizy*, size_t *sizx*, size_t *nch*, const double * *fillval*)

Allocate a new image of size [sizy, sizx, nch], and initialize all pixel values to fill.

Returns

pointer to the allocated image

5.6.2.3 image_ptr image_crop (const image_ptr *img*, fbox_t *crop*, int * *offset* = NULL, bool *shared* = true)

Crop image This function can be used in two ways, either get shared data from original image, or allocate a new image, which is more expensive.

Parameters

img	original image
crop	crop area within img
store	offset [offy offx] of the cropped patch inside image if not NULL
shared	indicate whether the result shares data with original image

Returns

cropped image patch, NULL if allocation failed

5.6.2.4 void image_delete (image_ptr *img*)

Delete image and associated memory.

Note

If it's a shared image(the "child"), no data is destroyed; if the image that owns the data is deleted, all descendants are not accessible anymore

5.6.2.5 void image_display (const image_ptr *img,* const std::string & *winname*)

Display an image.

Parameters

img	the image to be displayed
winname	window name, also serves as the identifier of the window

Note

Requires opencv library: libopencv_core, libopencv_highgui
If a window with the same name already exists, no new window is created
windows need to be destroyed later, using cv::destroyWindow()

5.6.2.6 void image_fill (image_ptr *img,* const double * *val*)

Fill all pixels with same values.

Parameters

img	target
val	value to be filled to each pixel, it should be at least the same length as img->nchannel

5.6.2.7 image_ptr image_readJPG (const char * *filename*)

Read Jpeg image file.

Returns

pointer to allocatd image, NULL if failed

Note

Requires opencv library: libopencv_core, libopencv_highgui

5.6.2.8 image_ptr image_reduce (const image_ptr *img*)

Get an image half the size of input one.

Parameters

img	the image to be reduced

Returns

reduced image

Note

input image remains alive and unchanged

5.6.2.9 image_ptr image_resize (const image_ptr *img*, double *scale*)

Resize an image using bilateral interpolation.

Parameters

image	the image to be resized
scale	resizing scale

Returns

resized image

See Also

image_subsample()

Note

input image remains alive and unchanged

5.6.2.10 void image_showDetection (const image_ptr *img*, const vector< bbox_t > *boxes*, const std::string & *winname*)

Show detection results on image and wait.

Parameters

img	detection target
boxes	detection results
winname	display window name, also serves as an identifier

See Also

image_showFaces()

5.6.2.11 void image_showFaces (const image_ptr *img*, const vector< bbox_t > *boxes*, const std::string & *winname*)

Show face detection results: face region, eyes, nose, mouth.

See Also

image_showDetection()

5.6.2.12 image_ptr image_subsample (const image_ptr *img*, double *scale*)

Fast image subsampling.

Unlike image_resize(), this function can only be used to down-scale an image, and focus more on anti-aliasing when building pyramid

Parameters

img	the image to be subsampled
scale	subsample scale (<1)

Returns

subsampleded image, or NULL if scale>1

See Also

image_resize()

Note

input image remains alive and unchanged
src image is not destroyed

5.7 eHimageFeature.h File Reference

Compute image features.

```
#include "eHimage.h"
#include "eHmatrix.h"
```

Functions

- mat3d_ptr eHhog (const image_ptr img, int sbin)
  Compute HOG feature of a color image.

5.7.1 Detailed Description

Compute image features.

Author

Hang Su

Date

2012-08

5.7.2 Function Documentation

5.7.2.1 mat3d_ptr eHhog (const image_ptr *img*, int *sbin*)

Compute HOG feature of a color image.

Parameters

img	3 channel double precision image
sbin	bin size

Returns

HOG feature matrix

Note

feature matrix is allocated inside, proper delete is necessary after use

5.8 eHmatrix.h File Reference

Basic matrix types and operations.

```
#include <stdlib.h>
```

Classes

- struct eHmatrix2d
  2D matrix
- struct eHmatrix3d
  3D matrix
- struct eHmatrixkd
  k-dimension matrix

Typedefs

- typedef struct eHmatrix2d mat2d_t
  2D matrix
- typedef mat2d_t * mat2d_ptr
  pointer to a 2D matrix
- typedef struct eHmatrix3d mat3d_t
  3D matrix
- typedef mat3d_t * mat3d_ptr
  pointer to a 3D matrix
- typedef struct eHmatrixkd matkd_t
  k-dimension matrix
- typedef matkd_t * matkd_ptr
  pointer to a k-dimension matrix

Functions

- mat2d_ptr mat2d_alloc (size_t sizy, size_t sizx)
  Allocate a 2D matrix.
- void mat2d_delete (mat2d_ptr)
  Destruct a 2D matrix.
- mat3d_ptr mat3d_alloc (size_t sizy, size_t sizx, size_t sizz)

Allocate a 3D matrix.

- void mat3d_delete (mat3d_ptr)

  Destruct a 3D matrix.

- void mat3d_pad (mat3d_ptr mat, const size_t *pad, double pad_val)

  Pad 3d matrix.

- void mat3d_fill (mat3d_ptr mat, const size_t *start, const size_t *width, double fill_val)

  Fill continous region of a 3D matrix with fill_val.

- matkd_ptr matkd_alloc (size_t k, size_t *sizs)

  Allocate a kD matrix.

- void matkd_delete (matkd_ptr)

  Delete a kD matrix.

5.8.1 Detailed Description

Basic matrix types and operations.

Author

Hang Su

Date

2012-07

5.8.2 Function Documentation

5.8.2.1 mat2d_ptr mat2d_alloc (size_t *sizy*, size_t *sizx*)

Allocate a 2D matrix.

See Also

mat2d_delete()

5.8.2.2 void mat2d_delete (mat2d_ptr)

Destruct a 2D matrix.

See Also

mat2d_alloc()

5.8.2.3 mat3d_ptr mat3d_alloc (size_t *sizy*, size_t *sizx*, size_t *sizz*)

Allocate a 3D matrix.

See Also

mat3d_delete()

5.8.2.4 void mat3d_delete (mat3d_ptr)

Destruct a 3D matrix.

See Also

mat3d_alloc()

5.8.2.5 void mat3d_fill (mat3d_ptr *mat*, const size_t * *start*, const size_t * *width*, double *fill_val*)

Fill continous region of a 3D matrix with fill_val.

Parameters

start	starting point
width	width in each dimension
fill_val	the value to be filled in

5.8.2.6 void mat3d_pad (mat3d_ptr *mat*, const size_t * *pad*, double *pad_val*)

Pad 3d matrix.

Parameters

pad	width of padding along each dimension
pad_val	values to be padded in each dimension

5.8.2.7 matkd_ptr matkd_alloc (size_t *k*, size_t * *sizs*)

Allocate a kD matrix.

See Also

matkd_delete()

5.8.2.8 void matkd_delete (matkd_ptr)

Delete a kD matrix.

See Also

matkd_alloc()

5.9 eHposemodel.h File Reference

Human body/pose detection model and operations.

```
#include <vector>
#include "eHimage.h"
#include "eHfilter.h"
#include "eHbbox.h"
```

Classes

- struct eHposemodel

  Human body/pose mdoel.

Typedefs

- typedef struct deformation_pose **posedef_t**
- typedef struct part_pose **posepart_t**
- typedef struct bias_pose **posebias_t**
- typedef struct eHposemodel **posemodel_t**

Functions

- posemodel_t ∗ posemodel_parseXml (char ∗xmlstr)

  Parse body/pose model from xml style string.
- posemodel_t ∗ posemodel_readFromFile (const char ∗filepath)

  Read body/pose model from file.
- vector< bbox_t > posemodel_detect (const posemodel_t ∗model, const image_ptr img, double thrs)

  Perform body/pose detection.
- vector< bbox_t > posemodel_detect (const posemodel_t ∗model, const image_ptr img)

  Perform body/pose detection using default threshold.
- void posemodel_delete (posemodel_t ∗model)

  Delete a pose model, release related memory.

5.9.1 Detailed Description

Human body/pose detection model and operations.

See Also

Y. Yang, D. Ramanan, "Articulated Pose Estimation using Flexible Mixtures of Parts". In CVPR 2011.

Author

Hang Su

Date

2012-08

5.9.2 Function Documentation

5.9.2.1 vector<bbox_t> posemodel_detect (const posemodel_t ∗ *model,* const image_ptr *img,* double *thrs*)

Perform body/pose detection.

Parameters

model	pose detection model
img	where to find human poses from
thrs	threshold used for pruning results

Returns

array of detected poses (together with part locations)

5.9.2.2 vector<bbox_t> posemodel_detect (const posemodel_t * *model*, const image_ptr *img*)

Perform body/pose detection using default threshold.

Parameters

model	pose detection model
img	where to find human poses from

Returns

array of detected poses (together with part locations)

5.9.2.3 posemodel_t* posemodel_parseXml (char * *xmlstr*)

Parse body/pose model from xml style string.

Note

xmlstr is modified during parsing, this can be avoided by using Non-Destrutive Mode of rapidxml

5.9.2.4 posemodel_t* posemodel_readFromFile (const char * *filepath*)

Read body/pose model from file.

See Also

posemodel_parseXml()

5.10 eHshiftdt.h File Reference

Generalized Distance Transform.

Functions

- void eHshiftdt (double *M, int *Ix, int *Iy, int lenx, int leny, int offx, int offy, int dstep, const double *vals, int sizx, int sizy, const double *w)

  Perform generalized distance transform.
- void **eHshiftdt** (double *M, int *Ix, int *Iy, const double *vals, int sizx, int sizy, const double *w)

5.10.1 Detailed Description

Generalized Distance Transform.

See Also

P. F. Felzenszwalb and D. P. Huttenlocher, "Distance Transforms of Sampled Functions". 2004.

Author

Hang Su

Date

2012-07

5.10.2 Function Documentation

5.10.2.1 void eHshiftdt (double * *M,* int * *lx,* int * *ly,* int *lenx,* int *leny,* int *offx,* int *offy,* int *dstep,* const double * *vals,* int *sizx,* int *sizy,* const double * *w*)

Perform generalized distance transform.

This applies computes a min convolution of a quadratic function ax^2+bx This outputs results on a shifted(offy, offx), subsampled(dstep) grid

Note

M, lx, ly should be properly allocated before passed in, they are then modified as output results

5.11 eHutils.h File Reference

Some useful stuff (string parsing etc.)

Functions

- int * parseCSVstr2int (const char *csvstr, int *siz, int offset=0)

  Parse given string(e.g. "10, 5\0") to integer array.
- double * parseCSVstr2double (const char *csvstr, int *siz)

  Parse given string(e.g. "1.2, 3.4\0") to double precision array.

5.11.1 Detailed Description

Some useful stuff (string parsing etc.)

Author

Hang Su

Date

2012-08

5.11.2 Function Documentation

5.11.2.1 double* parseCSVstr2double (const char * *csvstr,* int * *siz*)

Parse given string(e.g. "1.2, 3.4\0") to double precision array.

Parameters

csvstr	null-terminated c string
siz	amount of numbers inside the string; if -1 is passed, actural size will be calculated and stored in siz

Returns

array of numbers, memory is allocated for it

5.11.2.2 int* parseCSVstr2int (const char * *csvstr,* int * *siz,* int *offset* = 0)

Parse given string(e.g. "10, 5\0") to integer array.

Parameters

csvstr	null-terminated c string
siz	number of integers inside the string; if -1 is passed, actural size will be calculated and stored in siz

Returns

array of integers, memory is allocated for it

Index