ASMLibrary
5.0

Generated by Doxygen 1.6.1

Wed May 19 18:16:29 2010

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 asm_edge Struct Reference

Public Attributes

- int **ind1**
- int **ind2**

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

- D:/asmlibrary-4.0/src/asmbuilding.cpp

3.2 asm_model Class Reference

```
#include <asmlibrary.h>
```

Public Member Functions

- asm_model ()
- ∼asm_model ()
- bool Build (const char ∗∗image_lists, int n_images, const asm_shape ∗shape_datas, int n_shapes, bool binterpolate, int halfwidth, double percentage, int level_no, ASM_PROFILE_TYPE type)
- bool Fit (asm_shape &shape, const IplImage ∗grayimage, int max_iter=30, const scale_param ∗param=NULL)
- void WriteModel (FILE ∗f)
- void ReadModel (FILE ∗f)
- const asm_shape & GetMeanShape () const
- const int GetModesOfModel () const
- const double GetReferenceWidthOfFace () const

3.2.1 Detailed Description

Class for active shape model.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 asm_model::asm_model ()

Constructor

3.2.2.2 asm_model::∼asm_model ()

Destructor

3.2.3 Member Function Documentation

3.2.3.1 bool asm_model::Build (const char ∗∗ *image_lists*, int *n_images*, const asm_shape ∗ *shape_datas*, int *n_shapes*, bool *binterpolate*, int *halfwidth*, double *percentage*, int *level_no*, ASM_PROFILE_TYPE *type*)

Build active shape model.

Parameters:

image_lists the lists of image files

n_images the number of image files

shape_datas the lists of shape point data

n_shapes the number of shape data

binterpolate will sample pixel by bilinear interpolate or not?

halfwidth the half-side width of profile

percentage the fraction of shape variation to retain during PCA

level_no the number of pyramid level

type the type of sampling profile

Returns:

false on failure, true otherwise

3.2.3.2 bool asm_model::Fit (asm_shape & *shape*, const IplImage ∗ *grayimage*, int *max_iter* = `30`, const scale_param ∗ *param* = `NULL`)

Image alignment/fitting with an initial shape.

Parameters:

shape the point features that carries initial shape and also restores result after fitting

grayimage the gray image resource

max_iter the number of iteration

param the left and right index for x-direction in the shape (Always set *NULL*)

Returns:

false on failure, true otherwise

3.2.3.3 const asm_shape& asm_model::GetMeanShape () const `[inline]`

Get mean shape of model.

3.2.3.4 const int asm_model::GetModesOfModel () const `[inline]`

Get modes of shape distribution model (Will be calculated in shape's PCA)

3.2.3.5 const double asm_model::GetReferenceWidthOfFace () const `[inline]`

Get the width of mean shape [Identical to *m_asm_meanshape.GetWidth()*].

3.2.3.6 void asm_model::ReadModel (FILE ∗ *f*)

Read model data from file stream.

Parameters:

f stream to read from

3.2.3.7 void asm_model::WriteModel (FILE ∗ *f*)

Write model data to file stream.

Parameters:

f stream to write to

3.2.4 Member Data Documentation

3.2.4.1 struct profile_Nd_model∗ asm_model::classical_tdm `[read]`

1d/2d profile model

3.2.4.2 struct profile_lbp_model∗ asm_model::lbp_tdm `[read]`

lbp profile model

The documentation for this class was generated from the following files:

- D:/asmlibrary-4.0/src/asmlibrary.h
- D:/asmlibrary-4.0/src/asm_model.cpp

3.3 asm_profile Class Reference

```
#include <asmlibrary.h>
```

Public Member Functions

- asm_profile ()
- asm_profile (int length)
- asm_profile (const asm_profile &v)
- const double operator[ ] (int i) const
- double & operator[ ] (int i)
- const double ∗ GetData () const
- asm_profile & operator= (const asm_profile &p)
- asm_profile & operator= (double value)
- const asm_profile operator+ (const asm_profile &p) const
- asm_profile & operator+= (const asm_profile &p)
- const asm_profile operator- (const asm_profile &p) const
- asm_profile & operator-= (const asm_profile &p)
- const asm_profile operator∗ (double value) const
- asm_profile & operator∗= (double value)
- const asm_profile operator/ (double value) const
- asm_profile & operator/= (double value)
- void Clear ()
- void Resize (int length)
- void Write (FILE ∗f)
- void Read (FILE ∗f)
- const int NLength () const
- void GetProfile (const IplImage ∗image, const asm_shape &shape, int ipoint, void ∗whole_profile, int offset=0)
- void CalcProfileLBP (const asm_shape &shape, int ipoint, const int ∗lbp_img, int nrows, int ncols, int nblocklength, int xoffset, int yoffset, const int ∗mapping)
- void Normalize ()
- void CopyFrom (const CvMat ∗mat)
- void CopyFrom (const int ∗hist, int nbins)
- void CopyTo (CvMat ∗mat) const

Static Public Member Functions

- static void CalcProfile1D (const IplImage ∗image, const asm_shape &shape, int ipoint, int nwidth, bool binterpolate, int displace_offset, void ∗whole_profile, double ∗cos_alpha=NULL, double ∗sin_alpha=NULL)

3.3.1 Detailed Description

Class for profile.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 asm_profile::asm_profile ()

Null Constructor

3.3.2.2 asm_profile::asm_profile (int *length*)

Constructor

Parameters:

length Width of profile

3.3.2.3 asm_profile::asm_profile (const asm_profile & *v*)

Copy Constructor

3.3.3 Member Function Documentation

3.3.3.1 void asm_profile::CalcProfile1D (const IplImage $*$ *image*, const asm_shape & *shape*, int *ipoint*, int *nwidth*, bool *binterpolate*, int *displace_offset*, void $*$ *whole_profile*, double $*$ *cos_alpha* = NULL, double $*$ *sin_alpha* = NULL) [static]

Pre-Calculate 1D-profiles of all possible locations at one certain point vertex. Note: Use this before calling *GetProfile()*.

Parameters:

image the image resource

shape the shape information

ipoint the index of point vertex

nwidth the width of profile

binterpolate will sampling pixel by bilinear interpolate or not?

displace_offset how long will the profile be calculate?

whole_profile the buffer that store the entire profile (actually its length is $width + 2 * displace\_offset$)

cos_alpha the normal vector in x-direction

sin_alpha the normal vector in y-direction

3.3.3.2 void asm_profile::CalcProfileLBP (const asm_shape & *shape*, int *ipoint*, const int $*$ *lbp_img*, int *nrows*, int *ncols*, int *nblocklength*, int *xoffset*, int *yoffset*, const int $*$ *mapping*)

Calculate LBP-profiles of all possible locations at one certain point vertex. Note: Use this before calling *GetProfile()*.

Parameters:

shape the shape information

ipoint the index of point vertex

lbp_img the target image processed with LBP

nrows the height of *lbp_img*

ncols the width of *lbp_img*

nblocklength the width/height of recentage for sampling profile

xoffset the offset in x-direction away from the center *shape*[*ipoint*]

yoffset the offset in y-direction away from the center *shape*[*ipoint*]

mapping the mapping look-up table initialized by *LBP_InitMapping()*

3.3.3.3 void asm_profile::Clear ()

Release memory.

3.3.3.4 void asm_profile::CopyFrom (const int $*$ *hist*, int *nbins*)

Convert from LBP histogram to class asm_profile.

Parameters:

hist the histogram

nbins the dimension of histogram

3.3.3.5 void asm_profile::CopyFrom (const CvMat $*$ *mat*)

Convert from OpenCV's CvMat to class asm_profile.

Parameters:

mat CvMat that converted from

3.3.3.6 void asm_profile::CopyTo (CvMat $*$ *mat*) const

Convert from class asm_profile to OpenCV's CvMat.

Parameters:

mat CvMat that converted to

3.3.3.7 const double$*$ asm_profile::GetData () const `[inline]`

Access raw ptr of profile data.

Returns:

Raw ptr of profile data

3.3.3.8 void asm_profile::GetProfile (const IplImage $*$ *image*, const asm_shape & *shape*, int *ipoint*, void $*$ *whole_profile*, int *offset* = `0`)

Get the profile for one certain point vertex at the offset

Parameters:

image the image resource

shape the shape point information

ipoint the index of point vertex

whole_profile the buffer that store the entire profile (actually its length is $width + 2 * displace\_offset$)

offset the offset bias from the point $Shape[iPoint]$

3.3.3.9 const int asm_profile::NLength () const `[inline]`

Get the width of profile.

3.3.3.10 void asm_profile::Normalize ()

Normalize the profile so that its $L1$-norm is 1.

3.3.3.11 const asm_profile asm_profile::operator$*$ (double *value*) const

Override of operator $*$

3.3.3.12 asm_profile & asm_profile::operator$*$= (double *value*)

Override of operator $*=$

3.3.3.13 const asm_profile asm_profile::operator+ (const asm_profile & *p*) const

Override of operator +

3.3.3.14 asm_profile & asm_profile::operator+= (const asm_profile & *p*)

Override of operator +=

3.3.3.15 const asm_profile asm_profile::operator- (const asm_profile & *p*) const

Override of operator -

3.3.3.16 asm_profile & asm_profile::operator-= (const asm_profile & *p*)

Override of operator -=

3.3.3.17 const asm_profile asm_profile::operator/ (double *value*) const

Override of operator /

3.3.3.18 asm_profile & asm_profile::operator/= (double *value*)

Override of operator /=

3.3.3.19 asm_profile & asm_profile::operator= (double *value*)

Override of operator =

3.3.3.20 asm_profile & asm_profile::operator= (const asm_profile & *p*)

Override of operator =

3.3.3.21 double& asm_profile::operator[ ] (int *i*) `[inline]`

Access profile elements.

Parameters:

i Index of profile

Returns:

Value at the certain index

3.3.3.22 const double asm_profile::operator[ ] (int *i*) const `[inline]`

Access profile elements.

Parameters:

i Index of profile

Returns:

Value at the certain index

3.3.3.23 void asm_profile::Read (FILE ∗ *f*)

Read profile data from file stream.

Parameters:

f stream to read from

3.3.3.24 void asm_profile::Resize (int *length*)

Allocate memory.

Parameters:

length Width of profile

3.3.3.25 void asm_profile::Write (FILE ∗ *f*)

Write profile data into file stream.

Parameters:

f stream to write to

The documentation for this class was generated from the following files:

- D:/asmlibrary-4.0/src/asmlibrary.h
- D:/asmlibrary-4.0/src/asm_profile.cpp

3.4 asm_shape Class Reference

```
#include <asmlibrary.h>
```

Public Types

- enum { **LU**, **SVD**, **Direct** }

Public Member Functions

- asm_shape ()
- asm_shape (const asm_shape &v)
- ∼asm_shape ()
- const CvPoint2D32f operator[ ] (int i) const
- CvPoint2D32f & operator[ ] (int i)
- const int NPoints () const
- asm_shape & operator= (const asm_shape &s)
- asm_shape & operator= (double value)
- const asm_shape operator+ (const asm_shape &s) const
- asm_shape & operator+= (const asm_shape &s)
- const asm_shape operator- (const asm_shape &s) const
- asm_shape & operator-= (const asm_shape &s)
- const asm_shape operator∗ (double value) const
- asm_shape & operator∗= (double value)
- double operator∗ (const asm_shape &s) const
- const asm_shape operator/ (double value) const
- asm_shape & operator/= (double value)
- void Clear ()
- void Resize (int length)
- bool ReadAnnotations (const char ∗filename)
- void ReadFromASF (const char ∗filename)
- void ReadFromPTS (const char ∗filename)
- void Write (FILE ∗f)
- void Read (FILE ∗f)
- const double MinX () const
- const double MinY () const
- const double MaxX () const
- const double MaxY () const
- void GetLeftRight (int &ileft, int &iright) const
- const double GetWidth (int ileft=-1, int iright=-1) const
- const double GetHeight () const
- void COG (double &x, double &y) const
- void Centralize ()
- void Translate (double x, double y)
- void Scale (double s)
- void Rotate (double theta)
- void ScaleXY (double sx, double sy)
- double Normalize ()

- void AlignTransformation (const asm_shape &ref_shape, double &a, double &b, double &tx, double &ty, int method=SVD) const
- void AlignTo (const asm_shape &ref_shape, int method=SVD)
- void TransformPose (double a, double b, double tx, double ty)
- CvPoint2D32f CalcBisector (int i, int j, int k) const
- double GetNorm2 () const
- void CalcNormalVector (double &cos_alpha, double &sin_alpha, int i) const
- void CopyFrom (const CvMat ∗mat)
- void CopyTo (CvMat ∗mat) const

3.4.1 Detailed Description

Class for 2d shape data.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 asm_shape::asm_shape ()

Constructor

3.4.2.2 asm_shape::asm_shape (const asm_shape & *v*)

Copy Constructor

3.4.2.3 asm_shape::∼asm_shape ()

Destructor

3.4.3 Member Function Documentation

3.4.3.1 void asm_shape::AlignTo (const asm_shape & *ref_shape*, int *method* = SVD)

Align the shape to the reference shape.

Parameters:

ref_shape the reference shape

method method of similarity transform

3.4.3.2 void asm_shape::AlignTransformation (const asm_shape & *ref_shape*, double & *a*, double & *b*, double & *tx*, double & *ty*, int *method* = SVD) const

Calculate the similarity transform between one shape and another reference shape. Where the similarity transform is:

$T(a, b, tx, ty) = [a \ -b \ Tx; b \ a \ Ty; 0 \ 0 \ 1]$.

Parameters:

ref_shape the reference shape

a will return $s \times cos(theta)$ in form of similarity transform

b will return $s \times sin(theta)$ in form of similarity transform

tx will return Tx in form of similarity transform

ty will return Ty in form of similarity transform

method Method of similarity transform

3.4.3.3 CvPoint2D32f asm_shape::CalcBisector (int *i*, int *j*, int *k*) const

Calculate the angular bisector between two lines $Pi - Pj$ and $Pj - Pk$.

Parameters:

i the index of point vertex

j the index of point vertex

k the index of point vertex

Returns:

Angular bisector vector in form of $(cos(x), sin(x))^T$

3.4.3.4 void asm_shape::CalcNormalVector (double & *cos_alpha*, double & *sin_alpha*, int *i*) const

Calculate the normal vector at certain vertex around the shape contour.

Parameters:

cos_alpha the normal vector in x-direction

sin_alpha the normal vector in y-direction

i the index of point vertex

3.4.3.5 void asm_shape::Centralize ()

Translate the shape to make its center locate at (0, 0).

3.4.3.6 void asm_shape::Clear ()

Release memory.

3.4.3.7 void asm_shape::COG (double & *x*, double & *y*) const

Calculate center of gravity for shape.

Parameters:

x Value of center in x-direction

y Value of center in y-direction

3.4.3.8 void asm_shape::CopyFrom (const CvMat $*$ *mat*)

Convert from OpenCV's *CvMat* to class asm_shape

Parameters:

mat *CvMat* that converted from

3.4.3.9 void asm_shape::CopyTo (CvMat $*$ *mat*) const

Convert from class asm_shape to OpenCV's CvMat.

Parameters:

mat CvMat that converted to

3.4.3.10 const double asm_shape::GetHeight () const `[inline]`

Calculate height of shape.

3.4.3.11 void asm_shape::GetLeftRight (int & *ileft*, int & *iright*) const

Calculate the left and right index for x-direction in the shape.

Parameters:

ileft the index of points in x-direction which has the minimum x

iright the index of points in x-direction which has the maximum x

3.4.3.12 double asm_shape::GetNorm2 () const

Calculate the Euclidean norm ($L2$-norm).

Returns:

Euclidean norm

3.4.3.13 const double asm_shape::GetWidth (int *ileft* = `-1`, int *iright* = `-1`) const

Calculate width of shape.

Parameters:

ileft Index of points in x-direction which has the minimum x

iright Index of points in x-direction which has the maximum x

3.4.3.14 const double asm_shape::MaxX () const

Calculate maximum x-direction value of shape.

3.4.3.15 const double asm_shape::MaxY () const

Calculate maximum y-direction value of shape.

3.4.3.16 const double asm_shape::MinX () const

Calculate minimum x-direction value of shape.

3.4.3.17 const double asm_shape::MinY () const

Calculate minimum y-direction value of shape.

3.4.3.18 double asm_shape::Normalize ()

Normalize shape (zero_mean_unit_length) so that its center locates at (0, 0) and its $L2$-norm is 1.

Returns:

the $L2$-norm of original shape

3.4.3.19 const int asm_shape::NPoints () const `[inline]`

Get the number of points.

Returns:

Number of points

3.4.3.20 double asm_shape::operator* (const asm_shape & *s*) const

Override of operator $*$

3.4.3.21 const asm_shape asm_shape::operator* (double *value*) const

Override of operator $*$

3.4.3.22 asm_shape & asm_shape::operator*= (double *value*)

Override of operator $*=$

3.4.3.23 const asm_shape asm_shape::operator+ (const asm_shape & *s*) const

Override of operator +

3.4.3.24 asm_shape & asm_shape::operator+= (const asm_shape & *s*)

Override of operator +=

3.4.3.25 const asm_shape asm_shape::operator- (const asm_shape & *s*) const

Override of operator -

3.4.3.26 asm_shape & asm_shape::operator-= (const asm_shape & *s*)

Override of operator -=

3.4.3.27 const asm_shape asm_shape::operator/ (double *value*) const

Override of operator /

3.4.3.28 asm_shape & asm_shape::operator/= (double *value*)

Override of operator /=

3.4.3.29 asm_shape & asm_shape::operator= (double *value*)

Override of operator =.

3.4.3.30 asm_shape & asm_shape::operator= (const asm_shape & *s*)

Override of operator =

3.4.3.31 CvPoint2D32f& asm_shape::operator[ ] (int *i*) `[inline]`

Access elements by *CvPoint2D32f pt* = *shape*[*i*] to get *i-th* point in the shape.

Parameters:

i Index of points

Returns:

Point at the certain index

3.4.3.32 const CvPoint2D32f asm_shape::operator[ ] (int *i*) const `[inline]`

Access elements by *CvPoint2D32f pt* = *shape*[*i*] to get *i-th* point in the shape.

Parameters:

i Index of points

Returns:

Point at the certain index

3.4.3.33 void asm_shape::Read (FILE ∗ *f*)

Read shape data from file stream.

Parameters:

f stream to read from

3.4.3.34 bool asm_shape::ReadAnnotations (const char ∗ *filename*)

Read points from file.

Parameters:

filename the filename the stored shape data

Returns:

true on pts format, false on asf format, exit otherwise

3.4.3.35 void asm_shape::ReadFromASF (const char ∗ *filename*)

Read points from asf format file.

Parameters:

filename the filename the stored shape data

3.4.3.36 void asm_shape::ReadFromPTS (const char ∗ *filename*)

Read points from pts format file.

Parameters:

filename the filename the stored shape data

3.4.3.37 void asm_shape::Resize (int *length*)

Allocate memory.

Parameters:

length Number of of shape points

3.4.3.38 void asm_shape::Rotate (double *theta*)

Rotate shape by anti clock-wise.

Parameters:

theta Angle to be rotated

3.4.3.39 void asm_shape::Scale (double *s*)

Scale shape by an uniform factor.

Parameters:

s Scaling factor

3.4.3.40 void asm_shape::ScaleXY (double *sx*, double *sy*)

Scale shape in x and y direction respectively.

Parameters:

sx Scaling factor in x-direction

sy Scaling factor in y-direction

3.4.3.41 void asm_shape::TransformPose (double *a*, double *b*, double *tx*, double *ty*)

Transform Shape using the similarity transform $T(a, b, tx, ty)$.

3.4.3.42 void asm_shape::Translate (double *x*, double *y*)

Translate the shape.

Parameters:

x Value of translation factor in x-direction

y Value of translation factor in y-direction

3.4.3.43 void asm_shape::Write (FILE $*$ *f*)

Write shape data into file stream.

Parameters:

f stream to write to

The documentation for this class was generated from the following files:

- D:/asmlibrary-4.0/src/asmlibrary.h
- D:/asmlibrary-4.0/src/asm_shape.cpp

3.5 asmbuilding Class Reference

```
#include <asmbuilding.h>
```

Public Member Functions

- asmbuilding ()
- ∼asmbuilding ()
- bool Train (const char ∗∗imagelists, int n_images, const char ∗∗shapelists, int n_shapes, bool binterpolate=true, int halfwidth=8, double percentage=0.975, int level_no=4, ASM_PROFILE_TYPE type=PROFILE_1D)
- void BuildDetectMapping (const char ∗∗imagelists, int n_images, const char ∗∗shapelists, int n_shapes, detect_func my_func)
- bool Write (const char ∗filename)
- const asm_model ∗ **GetModel** () const

3.5.1 Detailed Description

Wrapped Class for building of active shape face model

3.5.2 Constructor & Destructor Documentation

3.5.2.1 asmbuilding::asmbuilding ()

Constructor

3.5.2.2 asmbuilding::∼asmbuilding ()

Destructor

3.5.3 Member Function Documentation

3.5.3.1 void asmbuilding::BuildDetectMapping (const char ∗∗ *imagelists*, int *n_images*, const char ∗∗ *shapelists*, int *n_shapes*, detect_func *my_func*)

Generate map relation between the face box and shape data groundtruth.

Parameters:

imagelists the lists of image files

n_images the number of image files

shapelists the lists of shape point files

n_shapes the number of shape data

my_func your implementing function for detecting only one object

3.5.3.2 bool asmbuilding::Train (const char ∗∗ *imagelists*, int *n_images*, const char ∗∗ *shapelists*, int *n_shapes*, bool *binterpolate* = `true`, int *halfwidth* = `8`, double *percentage* = `0.975`, int *level_no* = `4`, ASM_PROFILE_TYPE *type* = `PROFILE_1D`)

Build active shape model for human face.

Parameters:

imagelists the lists of image files

n_images the number of image files

shapelists the lists of shape point files

n_shapes the number of shape data

binterpolate will sample pixel by bilinear interpolate or not?

halfwidth the halfside width of profile

percentage the fraction of shape variation to retain during PCA

level_no the number of pyramid level

type the type of sampling profile

Returns:

false on failure, true otherwise

3.5.3.3 bool asmbuilding::Write (const char ∗ *filename*)

Write active shape model for human face to file.

Parameters:

filename the filename the model writes to

Returns:

false on failure, true otherwise Get raw ptr of asm_model.

The documentation for this class was generated from the following files:

- D:/asmlibrary-4.0/src/asmbuilding.h
- D:/asmlibrary-4.0/src/asmbuilding.cpp

3.6 asmfitting Class Reference

```
#include <asmfitting.h>
```

Public Member Functions

- asmfitting ()
- ~asmfitting ()
- void Fitting (asm_shape &shape, const IplImage ∗image, int n_iteration=30)
- void Fitting2 (asm_shape ∗shapes, int n_shapes, const IplImage ∗image, int n_iteration=30)
- bool ASMSeqSearch (asm_shape &shape, const IplImage ∗image, int frame_no=0, bool bopticalflow=false, int n_iteration=30)
- const asm_shape GetMappingDetShape () const
- const double GetMeanFaceWidth () const
- const asm_model ∗ **GetModel** () const
- bool Read (const char ∗filename)
- void Draw (IplImage ∗image, const asm_shape &shape)

3.6.1 Detailed Description

Wrapped Class for face alignment/tracking using active shape model

3.6.2 Constructor & Destructor Documentation

3.6.2.1 asmfitting::asmfitting ()

Constructor

3.6.2.2 asmfitting::~asmfitting ()

Destructor

3.6.3 Member Function Documentation

3.6.3.1 bool asmfitting::ASMSeqSearch (asm_shape & *shape*, const IplImage ∗ *image*, int *frame_no* = `0`, bool *bopticalflow* = `false`, int *n_iteration* = `30`)

Process face tracking on video/camera.

Parameters:

shape the point features that carries initial shape and also restores result after fitting

image the image resource

frame_no one certain frame number of video/camera

bopticalflow whether to use optical flow or not?

n_iteration the number of iteration during fitting

Returns:

false on failure, true otherwise. Get the Average Viola-Jone Box.

3.6.3.2 void asmfitting::Draw (IplImage ∗ *image*, const asm_shape & *shape*)

Draw point and edge on the image.

Parameters:

image the image resource

shape the shape after fitting

3.6.3.3 void asmfitting::Fitting (asm_shape & *shape*, const IplImage ∗ *image*, int *n_iteration* = `30`)

Process face alignment on image. (Only for one face box)

Parameters:

shape the point features that carries initial shape and also restores result after fitting

image the image resource

n_iteration the number of iteration during fitting

3.6.3.4 void asmfitting::Fitting2 (asm_shape ∗ *shapes*, int *n_shapes*, const IplImage ∗ *image*, int *n_iteration* = `30`)

Process face alignment on image. (For multi-face boxes)

Parameters:

shapes all shape datas that carry the fitting result

n_shapes the number of human face

image the image resource

n_iteration the number of iteration during fitting

3.6.3.5 const asm_shape asmfitting::GetMappingDetShape () const `[inline]`

Get the width of mean face.

3.6.3.6 const double asmfitting::GetMeanFaceWidth () const `[inline]`

Get raw ptr of asm_model.

3.6.3.7 bool asmfitting::Read (const char ∗ *filename*)

Read model data from file.

Parameters:

filename the filename that stores the model

Returns:

false on failure, true otherwise

The documentation for this class was generated from the following files:

- D:/asmlibrary-4.0/src/asmfitting.h
- D:/asmlibrary-4.0/src/asmfitting.cpp

3.7 lbp_circle_table Struct Reference

```
#include <asmlibrary.h>
```

Public Attributes

- int nsamples
- CvPoint ∗ points
- CvPoint2D32f ∗ offsets
- double ∗ multipliers

3.7.1 Detailed Description

"Circular neighborhood" is used to denote a situation where, instead of the traditional rectangular one, neighborhood pixels are defined to be the ones that lie at a certain distance from the center. The distance is also called "predicate". The number of samples at this distance and the predicate itself can be dynamically changed. In digital images, all pixels in a circular neighborhood do not necessarily match the pixel grid. Pixel values at these positions are obtained with bilinear interpolation or, if the interpolation flag is set to false, from the pixel nearest to the exact position.

3.7.2 Member Data Documentation

3.7.2.1 double∗ lbp_circle_table::multipliers

Precalculated values for interpolation multiplication.

3.7.2.2 int lbp_circle_table::nsamples

Number of neighborhood samples

3.7.2.3 CvPoint2D32f∗ lbp_circle_table::offsets

A precalculated table of interpolation offsets.

3.7.2.4 CvPoint∗ lbp_circle_table::points

A precalculated table of interpolation points.

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

- D:/asmlibrary-4.0/src/asmlibrary.h

3.8 profile_lbp_model Struct Reference

```
#include <asmlibrary.h>
```

Public Attributes

- asm_profile ∗∗ m_asm_meanprofile
- int nsamples
- int predicate
- int nblocklength
- LBP_MAPPING_TYPE type
- int ∗ mapping
- struct lbp_circle_table ∗ table
- int nbins
- int nlevels

3.8.1 Detailed Description

Profile distribution model for ASM_PROFILE_1D and ASM_PROFILE_2D

3.8.2 Member Data Documentation

3.8.2.1 asm_profile∗∗ profile_lbp_model::m_asm_meanprofile

the mean histogram for all landmark

3.8.2.2 int∗ profile_lbp_model::mapping

the look-up table

3.8.2.3 int profile_lbp_model::nbins

the dimension of feature vector

3.8.2.4 int profile_lbp_model::nblocklength

the width/height of block that for sampling profile

3.8.2.5 int profile_lbp_model::nlevels

the pyramid level

3.8.2.6 int profile_lbp_model::nsamples

the number of neighborhood samples

3.8.2.7 int profile_lbp_model::predicate

the radius of the neighborhood

3.8.2.8 struct lbp_circle_table∗ profile_lbp_model::table `[read]`

the precalculated circular local sampler instance

3.8.2.9 LBP_MAPPING_TYPE profile_lbp_model::type

the type of LBP mapping

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

- D:/asmlibrary-4.0/src/asmlibrary.h

3.9 profile_Nd_model Struct Reference

```
#include <asmlibrary.h>
```

Public Attributes

- CvMat *** m_P
- asm_profile ** m_asm_meanprofile
- CvMat *** m_G
- double * m_buffer

3.9.1 Detailed Description

Profile distribution model for ASM_PROFILE_1D and ASM_PROFILE_2D

3.9.2 Member Data Documentation

3.9.2.1 asm_profile** profile_Nd_model::m_asm_meanprofile

mean of profile data

3.9.2.2 double* profile_Nd_model::m_buffer

pre-allocated buffer for calculate profile

3.9.2.3 CvMat*** profile_Nd_model::m_G

inverted covariance matrix of profile data

3.9.2.4 CvMat*** profile_Nd_model::m_P

mean of profile data

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

- D:/asmlibrary-4.0/src/asmlibrary.h

3.10 scale_param Struct Reference

```
#include <asmlibrary.h>
```

Public Attributes

- int left
- int right

3.10.1 Detailed Description

Left and Right index in x-direction of shape

3.10.2 Member Data Documentation

3.10.2.1 int scale_param::left

Index of points in x-direction which has the minimum x

3.10.2.2 int scale_param::right

Index of points in x-direction which has the maximum x

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

- D:/asmlibrary-4.0/src/asmlibrary.h

Chapter 4

File Documentation

4.1 D:/asmlibrary-4.0/src/asmbuilding.h File Reference

```
#include "asmlibrary.h"
```

Classes

- class asmbuilding

4.1.1 Detailed Description

Classes for implementing building active shape model for face alignment/tracking.

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Version:

5.0-2010-5-20

4.2 D:/asmlibrary-4.0/src/asmfitting.h File Reference

```
#include "asmlibrary.h"
```

Classes

- class asmfitting

4.2.1 Detailed Description

Classes for implementing face alignment/tracking using active shape model.

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Version:

5.0-2010-5-20

4.3 D:/asmlibrary-4.0/src/asmlibrary.h File Reference

```
#include <stdio.h>
#include <cv.h>
#include <highgui.h>
```

Classes

- struct lbp_circle_table
- class asm_shape
- class asm_profile
- struct scale_param
- struct profile_Nd_model
- struct profile_lbp_model
- class asm_model

Typedefs

- typedef unsigned char **uchar**
- typedef bool(∗ detect_func)(asm_shape &shape, const IplImage ∗image)

Enumerations

- enum ASM_PROFILE_TYPE { **PROFILE_1D**, **PROFILE_2D**, **PROFILE_LBP** }
- enum LBP_MAPPING_TYPE { **MAP_UNIFORM**, **MAP_ROTMIN**, **MAP_UNIFORM_ROTMIN**, **MAP_NONE** }

Functions

- ASMLIB double GetX (double x, int offset, double cos_alpha)
- ASMLIB double GetY (double y, int offset, double sin_alpha)
- ASMLIB void WriteCvMat (FILE ∗f, const CvMat ∗mat)
- ASMLIB void ReadCvMat (FILE ∗f, CvMat ∗mat)
- ASMLIB uchar GetBilinearPixel (const IplImage ∗image, double x, double y, int width, int height)
- ASMLIB uchar GetOriPixel (const IplImage ∗image, double x, double y, int width, int height)
- ASMLIB double CalcMahalanobisDist (const CvMat ∗M, const double ∗x)
- ASMLIB double CalcChiSquareDist (const double ∗h, const double ∗H, int nbins)
- ASMLIB void InitShapeFromDetBox (asm_shape &shape, const asm_shape &det_shape, const asm_shape &ref_shape, double refwidth)
- ASMLIB void DrawPoints (IplImage ∗image, const asm_shape &shape)
- ASMLIB void DrawEdges (IplImage ∗image, const asm_shape &shape, int ∗edge_start, int ∗edge_end, int n_edges)
- ASMLIB bool ReadAllShapes (asm_shape ∗shapes, int n_shapes, const char ∗∗shape_lists, const char ∗∗image_lists)
- ASMLIB int LBP_onecount (unsigned int c, int bits=8)
- ASMLIB int LBP_transitions (unsigned int c, int bits=8)
- ASMLIB unsigned int LBP_rotmin (unsigned int c, int bits=8)
- ASMLIB int ∗ LBP_InitMapping (int nsamples, LBP_MAPPING_TYPE type)

- ASMLIB void LBP_FreeMapping (int ∗mapping)
- ASMLIB int LBP_GetMapSize (int nsamples, LBP_MAPPING_TYPE type)
- ASMLIB struct lbp_circle_table ∗ LBP_InitTable (int nsamples, double predicate)
- ASMLIB void LBP_FreeTable (struct lbp_circle_table ∗table)
- ASMLIB int LBP_CalcTransformedImage (const IplImage ∗grayimage, const struct lbp_circle_table ∗table, int nsamples, int predicate, LBP_MAPPING_TYPE type, int ∗result, CvRect ∗rect=NULL)
- ASMLIB void LBP_CalcFeatureVector (const int ∗result, int nrows, int ncols, const int ∗mapping, int ∗hist, int nbins)

4.3.1 Detailed Description

Functions, structures, classes for implementing active shape model.

Copyright (c) 2008-2010 by Yao Wei <`njustyw@gmail.com`>, all rights reserved.

Please cite the following or equivalent reference in any publicly available text that uses asmlibrary:

YAO Wei. Research on Facial Expression Recognition and Synthesis. *Master Thesis, Department of Computer Science and Technology, Nanjing University*, Feb 2009. `http://code.google.com/p/asmlibrary`

Version:

5.0-2010-5-20

4.3.2 Typedef Documentation

4.3.2.1 typedef bool(∗ detect_func)(asm_shape &shape, const IplImage ∗image)

You can define your own face detector function here

Parameters:

shapes Returned face detected box which stores the Top-Left and Bottom-Right points, so its *NPoints()* = 2 here.

image Image resource.

Returns:

false on no face exists in image, true otherwise.

4.3.3 Enumeration Type Documentation

4.3.3.1 enum ASM_PROFILE_TYPE

Predefined local texture (profile) types.

- PROFILE_1D: use only the pixels along the normal vector in the contour.
- PROFILE_2D: use the pixels located at the recentage.
- PROFILE_LBP: use the pixels processed with LBP-operator.

4.3.3.2 enum LBP_MAPPING_TYPE

Predefined mapping types.

- MAP_UNIFORM: use only patterns that have at most two 1-to-0 or 0-to-1 transitions. Junk the rest in one value.
- MAP_ROTMIN: rotate patterns to their minimum values.
- MAP_UNIFORM_ROTMIN: use only uniform patterns and rotate them to their minimum values.
- MAP_NONE: no mapping

4.3.4 Function Documentation

4.3.4.1 ASMLIB double CalcChiSquareDist (const double $*$ *h*, const double $*$ *H*, int *nbins*)

Calculate Chi square measure $d(H', H) = \sum i = 1, ..., n(H'(k) - H(k))^2/(H'(k) + H(k))$.

Parameters:

h the testing point's histogram

H the mean histogram

nbins the dimension of histogram

Returns:

Chi square measure

4.3.4.2 ASMLIB double CalcMahalanobisDist (const CvMat $*$ *M*, const double $*$ *x*)

Calculate Mahalanobis distance $d(x, M) = x' \times M \times x$.

Parameters:

M the covariance matrix

x the vector used to calculate the M-distance

Returns:

Mahalanobis distance

4.3.4.3 ASMLIB void DrawEdges (IplImage $*$ *image*, const asm_shape & *shape*, int $*$ *edge_start*, int $*$ *edge_end*, int *n_edges*)

Draw the fitting shape edge onto the image.

Parameters:

image the image resource

shape the shape data

edge_start the starting index of edges

edge_end the ending index of edges

n_edges the number of edges

4.3.4.4 ASMLIB void DrawPoints (IplImage $*$ *image*, const asm_shape & *shape*)

Draw the fitting shape points onto the image.

Parameters:

image the image resource

shape the shape data

4.3.4.5 ASMLIB uchar GetBilinearPixel (const IplImage $*$ *image*, double *x*, double *y*, int *width*, int *height*)

Image pixel at the location (x, y) using bilinear interpolate.

Parameters:

image the image resource

x the grid value in x-direction

y the grid value in y-direction

width the width of image

height the height of image

Returns:

the pixel value at (x, y)

$*$

4.3.4.6 ASMLIB uchar GetOriPixel (const IplImage $*$ *image*, double *x*, double *y*, int *width*, int *height*)

Image pixel at the location (x, y) using no interpolation.

Parameters:

image Image resource

x the grid value in x-direction

y the grid value in y-direction

width the width of image

height the height of image

Returns:

the pixel value (x, y)

4.3.4.7 ASMLIB double GetX (double *x*, int *offset*, double *cos_alpha*) `[inline]`

Parameters:

x the coordinate in x-direction of source object.

offset the length from the source object to the target object

cos_alpha the value of cosine angle between the horizontal line and source-target line

Returns:

the coordinate in x-direction of target object

4.3.4.8 ASMLIB double GetY (double *y*, int *offset*, double *sin_alpha*) `[inline]`

Parameters:

y the coordinate in y-direction of source object.

offset the length from the source object to the target object

sin_alpha the value of sine angle between the horizontal line and source-target line

Returns:

the coordinate in y-direction of target object

4.3.4.9 ASMLIB void InitShapeFromDetBox (asm_shape & *shape*, const asm_shape & *det_shape*, const asm_shape & *ref_shape*, double *refwidth*)

Initialize shape from the detected box.

Parameters:

shape the returned initial shape

det_shape the detected box calling by *asm_vjfacedetect::Detect()*

ref_shape the average mean shape

refwidth the width of average mean shape

4.3.4.10 ASMLIB void LBP_CalcFeatureVector (const int $*$ *result*, int *nrows*, int *ncols*, const int $*$ *mapping*, int $*$ *hist*, int *nbins*)

Calculate the feature vector from the image that has been transformed by LBP-operator.

Parameters:

result the transformed image with LBP

nrows the height of image

ncols the width of image

mapping the mapping look-up table initialized by *LBP_InitMapping()*

hist the histogram of feature vector

nbins the dimension of feature vector (identity to *LBP_GetMapSize*(*nsamples*, *type*))

4.3.4.11 ASMLIB int LBP_CalcTransformedImage (const IplImage ∗ *grayimage*, const struct lbp_circle_table ∗ *table*, int *nsamples*, int *predicate*, LBP_MAPPING_TYPE *type*, int ∗ *result*, CvRect ∗ *rect* = NULL)

Transform the source image using the LBP-operator .

Parameters:

grayimage the source image resource that must be 8-depth

table the precalculated circular local sampler instance

nsamples the number of neighborhood samples (e.g. 8u)

predicate the radius of the neighborhood (e.g. 1.5)

type the type of mapping

result the target image processed with LBP

rect the recentange for the masked image (if possible)

Returns:

-1 on failure, 0 otherwise

4.3.4.12 ASMLIB void LBP_FreeMapping (int ∗ *mapping*)

Free memory of the mapping look-up table.

Parameters:

mapping the ptr of mapping look-up table

4.3.4.13 ASMLIB void LBP_FreeTable (struct lbp_circle_table ∗ *table*)

Release the memory of the precalculated point value tables.

Parameters:

table the ptr of precalculated point value tables

4.3.4.14 ASMLIB int LBP_GetMapSize (int *nsamples*, LBP_MAPPING_TYPE *type*)

Get the number of distinct values (bins) the given mapping type can produce. This is useful in determining the length of the resulting feature vector when the mapping is in use.

Parameters:

nsamples the number of neighborhood samples

type the type of mapping

Returns:

the maximum value of the mapping (alway plus one for handling the case of outside or on the boundary)

4.3.4.15 ASMLIB int$*$ LBP_InitMapping (int *nsamples*, LBP_MAPPING_TYPE *type*)

Calculate a mapping look-up table for the given mapping type. The returned value is a newly allocated array of integers in which each item represents the index the particular LBP code should be mapped to. The returned array will take up 2^{samples} $*$sizeof(int) bytes of memory.

Parameters:

nsamples the number of neighborhood samples

type the type of mapping

Returns:

a look-up table

4.3.4.16 ASMLIB struct lbp_circle_table$*$ LBP_InitTable (int *nsamples*, double *predicate*) `[read]`

Update the precalculated point value tables.

Parameters:

nsamples the number of neighborhood samples (e.g. 8u)

predicate the radius of the neighborhood (e.g. 1.5)

Returns:

A new Circular Local Sampler instance

4.3.4.17 ASMLIB int LBP_onecount (unsigned int *c*, int *bits* = 8)

Get the number of ones in a binary number.

Parameters:

c the number

bits the number of bits to consider

4.3.4.18 ASMLIB unsigned int LBP_rotmin (unsigned int *c*, int *bits* = 8)

Rotate a binary number to its minimum value.

Parameters:

c the number

bits the number of bits to consider

4.3.4.19 ASMLIB int LBP_transitions (unsigned int *c*, int *bits* = 8)

Get the number of 0-to-1 or 1-to-0 transitions in a binary number.

Parameters:

c the number

bits the number of bits to consider

4.3.4.20 ASMLIB bool ReadAllShapes (asm_shape ∗ *shapes*, int *n_shapes*, const char ∗∗ *shape_lists*, const char ∗∗ *image_lists*)

Read the whole shape datas from the file lists

Parameters:

shapes all shape datas

n_shapes the number of shape data

shape_lists the lists of shape point files

image_lists the lists of image files

Returns:

false on failure, true otherwise

4.3.4.21 ASMLIB void ReadCvMat (FILE ∗ *f*, CvMat ∗ *mat*)

Read *CvMat* data from file stream.

Parameters:

f the stream to read from.

mat the *CvMat* that will be read.

4.3.4.22 ASMLIB void WriteCvMat (FILE ∗ *f*, const CvMat ∗ *mat*)

Write *CvMat* data to file stream.

Parameters:

f the stream to write to.

mat the *CvMat* that will be wrote.

4.4 D:/asmlibrary-4.0/src/demo_build.cpp File Reference

```
#include "asmbuilding.h"
#include "vjfacedetect.h"
#include <iostream>
#include <string>
#include <vector>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
```

Typedefs

- typedef vector< string > **filelists**

4.4.1 Detailed Description

A demo show how to build a active shape model

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Please cite the following or equivalent reference in any publicly available text that uses asmlibrary:

YAO Wei. Research on Facial Expression Recognition and Synthesis. *Master Thesis, Department of Computer Science and Technology, Nanjing University*, Feb 2009. http://code.google.com/p/asmlibrary

Version:

5.0-2010-5-20

4.5 D:/asmlibrary-4.0/src/demo_fit.cpp File Reference

```
#include <vector>
#include <string>
#include <iostream>
#include "asmfitting.h"
#include "vjfacedetect.h"
#include "video_camera.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

4.5.1 Detailed Description

A demo show how to do image alignment (face tracking) using active shape model

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Please cite the following or equivalent reference in any publicly available text that uses asmlibrary:

YAO Wei. Research on Facial Expression Recognition and Synthesis. *Master Thesis, Department of Computer Science and Technology, Nanjing University*, Feb 2009. http://code.google.com/p/asmlibrary

Version:

5.0-2010-5-20

4.6 D:/asmlibrary-4.0/src/video_camera.cpp File Reference

```
#include "video_camera.h"
#include <stdio.h>
```

Functions

- int open_video (const char ∗filename)
- void close_video ()
- IplImage ∗ read_from_video (int frame_no)
- bool open_camera (int index)
- void close_camera ()
- IplImage ∗ read_from_camera ()

4.6.1 Detailed Description

Implemention for handling camera and avi-video

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Please cite the following or equivalent reference in any publicly available text that uses asmlibrary:

YAO Wei. Research on Facial Expression Recognition and Synthesis. *Master Thesis, Department of Computer Science and Technology, Nanjing University*, Feb 2009. http://code.google.com/p/asmlibrary

Version:

5.0-2010-5-20

4.6.2 Function Documentation

4.6.2.1 void close_camera ()

Close camara and release memory.

4.6.2.2 void close_video ()

Close avi and release memory.

4.6.2.3 bool open_camera (int *index*)

Capture from live camera.

Parameters:

index camara index

Returns:

false on failure, true otherwise

4.6.2.4 int open_video (const char ∗ *filename*)

Open an AVI file.

Parameters:

filename the video file located in

Returns:

-1 on failure, frame count of the video otherwise

4.6.2.5 IplImage∗ read_from_camera ()

Get one certain frame of live camera.

Returns:

Internal IplImage ptr

4.6.2.6 IplImage∗ read_from_video (int *frame_no*)

Get one certain frame of video.

Parameters:

frame_no which frame

Returns:

Internal IplImage ptr

4.7 D:/asmlibrary-4.0/src/video_camera.h File Reference

```
#include "asmlibrary.h"
```

Functions

- int open_video (const char ∗filename)
- IplImage ∗ read_from_video (int frame_no)
- void close_video ()
- bool open_camera (int index)
- IplImage ∗ read_from_camera ()
- void close_camera ()

4.7.1 Detailed Description

Routines for handling camera and avi-video

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Please cite the following or equivalent reference in any publicly available text that uses asmlibrary:

YAO Wei. Research on Facial Expression Recognition and Synthesis. *Master Thesis, Department of Computer Science and Technology, Nanjing University*, Feb 2009. http://code.google.com/p/asmlibrary

Version:

5.0-2010-5-20

4.7.2 Function Documentation

4.7.2.1 void close_camera ()

Close camara and release memory.

4.7.2.2 void close_video ()

Close avi and release memory.

4.7.2.3 bool open_camera (int *index*)

Capture from live camera.

Parameters:

index camara index

Returns:

false on failure, true otherwise

4.7.2.4 int open_video (const char ∗ *filename*)

Open an AVI file.

Parameters:

filename the video file located in

Returns:

-1 on failure, frame count of the video otherwise

4.7.2.5 IplImage∗ read_from_camera ()

Get one certain frame of live camera.

Returns:

Internal IplImage ptr

4.7.2.6 IplImage∗ read_from_video (int *frame_no*)

Get one certain frame of video.

Parameters:

frame_no which frame

Returns:

Internal IplImage ptr

4.8 D:/asmlibrary-4.0/src/vjfacedetect.cpp File Reference

```
#include "vjfacedetect.h"
```

Functions

- bool init_detect_cascade (const char ∗cascade_name)
- void destory_detect_cascade ()
- bool detect_all_faces (asm_shape ∗∗Shapes, int &n_shapes, const IplImage ∗image)
- void free_shape_memeory (asm_shape ∗∗shapes)
- bool detect_one_face (asm_shape &Shape, const IplImage ∗image)

4.8.1 Detailed Description

Implemention for Viola and Jones's AdaBoost Haar-like Face Detector

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Please cite the following or equivalent reference in any publicly available text that uses asmlibrary:

YAO Wei. Research on Facial Expression Recognition and Synthesis. *Master Thesis, Department of Computer Science and Technology, Nanjing University*, Feb 2009. http://code.google.com/p/asmlibrary

Version:

5.0-2010-5-20

4.8.2 Function Documentation

4.8.2.1 void destory_detect_cascade ()

Release the memory of adaboost cascade face detector

4.8.2.2 bool detect_all_faces (asm_shape ∗∗ *shapes*, int & *n_shapes*, const IplImage ∗ *image*)

Detect all human face from image.

Parameters:

shapes return face detected box which stores the Top-Left and Bottom-Right points, so its *NPoints()* = 2 here

n_shapes the numbers of faces to return

image the image resource

Returns:

false on no face exists in image, true otherwise

4.8.2.3 bool detect_one_face (asm_shape & *shape*, const IplImage $*$ *image*)

Detect only one face from image, and this human face is located as close as to the center of image

Parameters:

shape return face detected box which stores the Top-Left and Bottom-Right points, so its *NPoints()* = 2 here

image the image resource

Returns:

false on no face exists in image, true otherwise

4.8.2.4 void free_shape_memeory (asm_shape $**$ *shapes*)

Release the shape resource allocated by detect_all_faces().

Parameters:

shapes the ptr of asm_shape []

4.8.2.5 bool init_detect_cascade (const char $*$ *cascade_name* = `"haarcascade_frontalface_alt2.xml"`)

Load adaboost cascade file for detect face.

Parameters:

cascade_name Filename the cascade detector located in

Returns:

false on failure, true otherwise

4.9 D:/asmlibrary-4.0/src/vjfacedetect.h File Reference

```
#include "asmlibrary.h"
```

Functions

- bool init_detect_cascade (const char ∗cascade_name="haarcascade_frontalface_alt2.xml")
- void destory_detect_cascade ()
- bool detect_one_face (asm_shape &shape, const IplImage ∗image)
- bool detect_all_faces (asm_shape ∗∗shapes, int &n_shapes, const IplImage ∗image)
- void free_shape_memeory (asm_shape ∗∗shapes)

4.9.1 Detailed Description

Routines for Viola and Jones's AdaBoost Haar-like Face Detector

Copyright (c) 2008-2010 by Yao Wei <njustyw@gmail.com>, all rights reserved.

Please cite the following or equivalent reference in any publicly available text that uses asmlibrary:

YAO Wei. Research on Facial Expression Recognition and Synthesis. *Master Thesis, Department of Computer Science and Technology, Nanjing University*, Feb 2009. http://code.google.com/p/asmlibrary

Version:

5.0-2010-5-20

4.9.2 Function Documentation

4.9.2.1 void destory_detect_cascade ()

Release the memory of adaboost cascade face detector

4.9.2.2 bool detect_all_faces (asm_shape ∗∗ *shapes*, int & *n_shapes*, const IplImage ∗ *image*)

Detect all human face from image.

Parameters:

shapes return face detected box which stores the Top-Left and Bottom-Right points, so its *NPoints()* = 2 here

n_shapes the numbers of faces to return

image the image resource

Returns:

false on no face exists in image, true otherwise

4.9.2.3 bool detect_one_face (asm_shape & *shape*, const IplImage ∗ *image*)

Detect only one face from image, and this human face is located as close as to the center of image

Parameters:

shape return face detected box which stores the Top-Left and Bottom-Right points, so its *NPoints()* = 2 here

image the image resource

Returns:

false on no face exists in image, true otherwise

4.9.2.4 void free_shape_memeory (asm_shape ∗∗ *shapes*)

Release the shape resource allocated by detect_all_faces().

Parameters:

shapes the ptr of asm_shape []

4.9.2.5 bool init_detect_cascade (const char ∗ *cascade_name* = `"haarcascade_frontalface_alt2.xml"`)

Load adaboost cascade file for detect face.

Parameters:

cascade_name Filename the cascade detector located in

Returns:

false on failure, true otherwise

Index