

# ASMLibrary

## 5.0

Generated by Doxygen 1.6.1

Wed May 19 18:16:29 2010



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	asm_edge Struct Reference . . . . .	5
3.2	asm_model Class Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.2.2	Constructor & Destructor Documentation . . . . .	6
3.2.2.1	asm_model . . . . .	6
3.2.2.2	~asm_model . . . . .	6
3.2.3	Member Function Documentation . . . . .	6
3.2.3.1	Build . . . . .	6
3.2.3.2	Fit . . . . .	7
3.2.3.3	GetMeanShape . . . . .	7
3.2.3.4	GetModesOfModel . . . . .	7
3.2.3.5	GetReferenceWidthOfFace . . . . .	7
3.2.3.6	ReadModel . . . . .	7
3.2.3.7	WriteModel . . . . .	7
3.2.4	Member Data Documentation . . . . .	8
3.2.4.1	classical_tdm . . . . .	8
3.2.4.2	lbp_tdm . . . . .	8
3.3	asm_profile Class Reference . . . . .	9
3.3.1	Detailed Description . . . . .	9
3.3.2	Constructor & Destructor Documentation . . . . .	10
3.3.2.1	asm_profile . . . . .	10
3.3.2.2	asm_profile . . . . .	10

3.3.2.3	asm_profile	10
3.3.3	Member Function Documentation	10
3.3.3.1	CalcProfile1D	10
3.3.3.2	CalcProfileLBP	10
3.3.3.3	Clear	11
3.3.3.4	CopyFrom	11
3.3.3.5	CopyFrom	11
3.3.3.6	CopyTo	11
3.3.3.7	GetData	11
3.3.3.8	GetProfile	12
3.3.3.9	NLength	12
3.3.3.10	Normalize	12
3.3.3.11	operator*	12
3.3.3.12	operator*=	12
3.3.3.13	operator+	12
3.3.3.14	operator+=	12
3.3.3.15	operator-	12
3.3.3.16	operator-=	12
3.3.3.17	operator/	13
3.3.3.18	operator/=	13
3.3.3.19	operator=	13
3.3.3.20	operator=	13
3.3.3.21	operator[]	13
3.3.3.22	operator[]	13
3.3.3.23	Read	13
3.3.3.24	Resize	14
3.3.3.25	Write	14
3.4	asm_shape Class Reference	15
3.4.1	Detailed Description	16
3.4.2	Constructor & Destructor Documentation	16
3.4.2.1	asm_shape	16
3.4.2.2	asm_shape	16
3.4.2.3	~asm_shape	16
3.4.3	Member Function Documentation	16
3.4.3.1	AlignTo	16
3.4.3.2	AlignTransformation	16

3.4.3.3	CalcBisector	17
3.4.3.4	CalcNormalVector	17
3.4.3.5	Centralize	17
3.4.3.6	Clear	17
3.4.3.7	COG	17
3.4.3.8	CopyFrom	18
3.4.3.9	CopyTo	18
3.4.3.10	GetHeight	18
3.4.3.11	GetLeftRight	18
3.4.3.12	GetNorm2	18
3.4.3.13	GetWidth	18
3.4.3.14	MaxX	18
3.4.3.15	MaxY	19
3.4.3.16	MinX	19
3.4.3.17	MinY	19
3.4.3.18	Normalize	19
3.4.3.19	NPoints	19
3.4.3.20	operator*	19
3.4.3.21	operator*	19
3.4.3.22	operator*=	19
3.4.3.23	operator+	19
3.4.3.24	operator+=	19
3.4.3.25	operator-	20
3.4.3.26	operator-=	20
3.4.3.27	operator/	20
3.4.3.28	operator/=	20
3.4.3.29	operator=	20
3.4.3.30	operator=	20
3.4.3.31	operator[]	20
3.4.3.32	operator[]	20
3.4.3.33	Read	21
3.4.3.34	ReadAnnotations	21
3.4.3.35	ReadFromASF	21
3.4.3.36	ReadFromPTS	21
3.4.3.37	Resize	21
3.4.3.38	Rotate	21

3.4.3.39	Scale	22
3.4.3.40	ScaleXY	22
3.4.3.41	TransformPose	22
3.4.3.42	Translate	22
3.4.3.43	Write	22
3.5	asmbuilding Class Reference	23
3.5.1	Detailed Description	23
3.5.2	Constructor & Destructor Documentation	23
3.5.2.1	asmbuilding	23
3.5.2.2	~asmbuilding	23
3.5.3	Member Function Documentation	23
3.5.3.1	BuildDetectMapping	23
3.5.3.2	Train	24
3.5.3.3	Write	24
3.6	asmfitting Class Reference	25
3.6.1	Detailed Description	25
3.6.2	Constructor & Destructor Documentation	25
3.6.2.1	asmfitting	25
3.6.2.2	~asmfitting	25
3.6.3	Member Function Documentation	25
3.6.3.1	ASMSeqSearch	25
3.6.3.2	Draw	26
3.6.3.3	Fitting	26
3.6.3.4	Fitting2	26
3.6.3.5	GetMappingDetShape	26
3.6.3.6	GetMeanFaceWidth	26
3.6.3.7	Read	26
3.7	lbp_circle_table Struct Reference	28
3.7.1	Detailed Description	28
3.7.2	Member Data Documentation	28
3.7.2.1	multipliers	28
3.7.2.2	nsamples	28
3.7.2.3	offsets	28
3.7.2.4	points	28
3.8	profile_lbp_model Struct Reference	29
3.8.1	Detailed Description	29

3.8.2	Member Data Documentation	29
3.8.2.1	m_asm_meanprofile	29
3.8.2.2	mapping	29
3.8.2.3	nbins	29
3.8.2.4	nblocklength	29
3.8.2.5	nlevels	29
3.8.2.6	nsamples	29
3.8.2.7	predicate	30
3.8.2.8	table	30
3.8.2.9	type	30
3.9	profile_Nd_model Struct Reference	31
3.9.1	Detailed Description	31
3.9.2	Member Data Documentation	31
3.9.2.1	m_asm_meanprofile	31
3.9.2.2	m_buffer	31
3.9.2.3	m_G	31
3.9.2.4	m_P	31
3.10	scale_param Struct Reference	32
3.10.1	Detailed Description	32
3.10.2	Member Data Documentation	32
3.10.2.1	left	32
3.10.2.2	right	32
<b>4</b>	<b>File Documentation</b>	<b>33</b>
4.1	D:/asmlibrary-4.0/src/asmbuilding.h File Reference	33
4.1.1	Detailed Description	33
4.2	D:/asmlibrary-4.0/src/asmfitting.h File Reference	34
4.2.1	Detailed Description	34
4.3	D:/asmlibrary-4.0/src/asmlibrary.h File Reference	35
4.3.1	Detailed Description	36
4.3.2	Typedef Documentation	36
4.3.2.1	detect_func	36
4.3.3	Enumeration Type Documentation	36
4.3.3.1	ASM_PROFILE_TYPE	36
4.3.3.2	LBP_MAPPING_TYPE	37
4.3.4	Function Documentation	37
4.3.4.1	CalcChiSquareDist	37

4.3.4.2	CalcMahalanobisDist	37
4.3.4.3	DrawEdges	37
4.3.4.4	DrawPoints	38
4.3.4.5	GetBilinearPixel	38
4.3.4.6	GetOriPixel	38
4.3.4.7	GetX	38
4.3.4.8	GetY	39
4.3.4.9	InitShapeFromDetBox	39
4.3.4.10	LBP_CalcFeatureVector	39
4.3.4.11	LBP_CalcTransformedImage	40
4.3.4.12	LBP_FreeMapping	40
4.3.4.13	LBP_FreeTable	40
4.3.4.14	LBP_GetMapSize	40
4.3.4.15	LBP_InitMapping	41
4.3.4.16	LBP_InitTable	41
4.3.4.17	LBP_onecount	41
4.3.4.18	LBP_rotmin	41
4.3.4.19	LBP_transitions	42
4.3.4.20	ReadAllShapes	42
4.3.4.21	ReadCvMat	42
4.3.4.22	WriteCvMat	42
4.4	D:/asmlibrary-4.0/src/demo_build.cpp File Reference	43
4.4.1	Detailed Description	43
4.5	D:/asmlibrary-4.0/src/demo_fit.cpp File Reference	44
4.5.1	Detailed Description	44
4.6	D:/asmlibrary-4.0/src/video_camera.cpp File Reference	45
4.6.1	Detailed Description	45
4.6.2	Function Documentation	45
4.6.2.1	close_camera	45
4.6.2.2	close_video	45
4.6.2.3	open_camera	45
4.6.2.4	open_video	46
4.6.2.5	read_from_camera	46
4.6.2.6	read_from_video	46
4.7	D:/asmlibrary-4.0/src/video_camera.h File Reference	47
4.7.1	Detailed Description	47



4.7.2	Function Documentation . . . . .	47
4.7.2.1	close_camera . . . . .	47
4.7.2.2	close_video . . . . .	47
4.7.2.3	open_camera . . . . .	47
4.7.2.4	open_video . . . . .	48
4.7.2.5	read_from_camera . . . . .	48
4.7.2.6	read_from_video . . . . .	48
4.8	D:/asmlibrary-4.0/src/vjfacetect.cpp File Reference . . . . .	49
4.8.1	Detailed Description . . . . .	49
4.8.2	Function Documentation . . . . .	49
4.8.2.1	destory_detect_cascade . . . . .	49
4.8.2.2	detect_all_faces . . . . .	49
4.8.2.3	detect_one_face . . . . .	50
4.8.2.4	free_shape_memeory . . . . .	50
4.8.2.5	init_detect_cascade . . . . .	50
4.9	D:/asmlibrary-4.0/src/vjfacetect.h File Reference . . . . .	51
4.9.1	Detailed Description . . . . .	51
4.9.2	Function Documentation . . . . .	51
4.9.2.1	destory_detect_cascade . . . . .	51
4.9.2.2	detect_all_faces . . . . .	51
4.9.2.3	detect_one_face . . . . .	52
4.9.2.4	free_shape_memeory . . . . .	52
4.9.2.5	init_detect_cascade . . . . .	52



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">asm_edge</a>	5
<a href="#">asm_model</a>	6
<a href="#">asm_profile</a>	9
<a href="#">asm_shape</a>	15
<a href="#">asmbuilding</a>	23
<a href="#">asmfitting</a>	25
<a href="#">lbp_circle_table</a>	28
<a href="#">profile_lbp_model</a>	29
<a href="#">profile_Nd_model</a>	31
<a href="#">scale_param</a>	32



# Chapter 2

## File Index

### 2.1 File List

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

D:/asmlibrary-4.0/src/ <a href="#">asmbuilding.h</a>	33
D:/asmlibrary-4.0/src/ <a href="#">asmfitting.h</a>	34
D:/asmlibrary-4.0/src/ <a href="#">asmlibrary.h</a>	35
D:/asmlibrary-4.0/src/ <a href="#">demo_build.cpp</a>	43
D:/asmlibrary-4.0/src/ <a href="#">demo_fit.cpp</a>	44
D:/asmlibrary-4.0/src/ <b>resource.h</b>	<b>??</b>
D:/asmlibrary-4.0/src/ <a href="#">video_camera.cpp</a>	45
D:/asmlibrary-4.0/src/ <a href="#">video_camera.h</a>	47
D:/asmlibrary-4.0/src/ <a href="#">vjfacedetect.cpp</a>	49
D:/asmlibrary-4.0/src/ <a href="#">vjfacedetect.h</a>	51



## 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 IpImage \* *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:

$$T(a, b, tx, ty) = \begin{bmatrix} a & -b & Tx & 0 \\ b & a & Ty & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

**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  $P_i - P_j$  and  $P_j - P_k$ .

##### 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 bin-terpolate=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](mailto: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](mailto: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](mailto: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  $d(H', H) = \sum_{i=1, \dots, n} (H'(k) - H(k))^2 / (H'(k) + H(k))$ .

**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\_vffacedetect::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 (always 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  $2^{\text{samples}} * \text{sizeof}(\text{int})$  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](mailto: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](mailto: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

Implementation for handling camera and avi-video

Copyright (c) 2008-2010 by Yao Wei <[njustyw@gmail.com](mailto: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](mailto: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/vjfacetect.cpp File Reference

```
#include "vjfacetect.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

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

Copyright (c) 2008-2010 by Yao Wei <[njustyw@gmail.com](mailto: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/vjfacetect.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](mailto: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\_-\nfrontalface\_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

- ~asm\_model
  - asm\_model, [6](#)
- ~asm\_shape
  - asm\_shape, [16](#)
- ~asmbuilding
  - asmbuilding, [23](#)
- ~asmfitting
  - asmfitting, [25](#)
- AlignTo
  - asm\_shape, [16](#)
- AlignTransformation
  - asm\_shape, [16](#)
- asm\_edge, [5](#)
- asm\_model, [6](#)
  - ~asm\_model, [6](#)
  - asm\_model, [6](#)
  - asm\_model, [6](#)
  - Build, [6](#)
  - classical\_tdm, [8](#)
  - Fit, [7](#)
  - GetMeanShape, [7](#)
  - GetModesOfModel, [7](#)
  - GetReferenceWidthOfFace, [7](#)
  - lbp\_tdm, [8](#)
  - ReadModel, [7](#)
  - WriteModel, [7](#)
- asm\_profile, [9](#)
  - asm\_profile, [10](#)
  - asm\_profile, [10](#)
  - CalcProfile1D, [10](#)
  - CalcProfileLBP, [10](#)
  - Clear, [11](#)
  - CopyFrom, [11](#)
  - CopyTo, [11](#)
  - GetData, [11](#)
  - GetProfile, [11](#)
  - NLength, [12](#)
  - Normalize, [12](#)
  - operator\*, [12](#)
  - operator\*=[12](#)
  - operator+, [12](#)
  - operator+=, [12](#)
  - operator-, [12](#)
  - operator-=, [12](#)
  - operator/, [12](#)
  - operator/=, [13](#)
  - operator=, [13](#)
  - Read, [13](#)
  - Resize, [13](#)
  - Write, [14](#)
- ASM\_PROFILE\_TYPE
  - asmlibrary.h, [36](#)
- asm\_shape, [15](#)
  - ~asm\_shape, [16](#)
  - AlignTo, [16](#)
  - AlignTransformation, [16](#)
  - asm\_shape, [16](#)
  - asm\_shape, [16](#)
  - CalcBisector, [17](#)
  - CalcNormalVector, [17](#)
  - Centralize, [17](#)
  - Clear, [17](#)
  - COG, [17](#)
  - CopyFrom, [17](#)
  - CopyTo, [18](#)
  - GetHeight, [18](#)
  - GetLeftRight, [18](#)
  - GetNorm2, [18](#)
  - GetWidth, [18](#)
  - MaxX, [18](#)
  - MaxY, [18](#)
  - MinX, [19](#)
  - MinY, [19](#)
  - Normalize, [19](#)
  - NPoints, [19](#)
  - operator\*, [19](#)
  - operator\*=[19](#)
  - operator+, [19](#)
  - operator+=, [19](#)
  - operator-, [19](#)
  - operator-=, [20](#)
  - operator/, [20](#)
  - operator/=, [20](#)
  - operator=, [20](#)
  - Read, [20](#)
  - ReadAnnotations, [21](#)
  - ReadFromASF, [21](#)
  - ReadFromPTS, [21](#)
  - Resize, [21](#)

- Rotate, 21
- Scale, 21
- ScaleXY, 22
- TransformPose, 22
- Translate, 22
- Write, 22
- asmbuilding, 23
  - ~asmbuilding, 23
  - asmbuilding, 23
  - BuildDetectMapping, 23
  - Train, 23
  - Write, 24
- asmfitting, 25
  - ~asmfitting, 25
  - asmfitting, 25
  - ASMSeqSearch, 25
  - Draw, 25
  - Fitting, 26
  - Fitting2, 26
  - GetMappingDetShape, 26
  - GetMeanFaceWidth, 26
  - Read, 26
- asmlibrary.h
  - ASM\_PROFILE\_TYPE, 36
  - CalcChiSquareDist, 37
  - CalcMahalanobisDist, 37
  - detect\_func, 36
  - DrawEdges, 37
  - DrawPoints, 37
  - GetBilinearPixel, 38
  - GetOriPixel, 38
  - GetX, 38
  - GetY, 39
  - InitShapeFromDetBox, 39
  - LBP\_CalcFeatureVector, 39
  - LBP\_CalcTransformedImage, 39
  - LBP\_FreeMapping, 40
  - LBP\_FreeTable, 40
  - LBP\_GetMapSize, 40
  - LBP\_InitMapping, 40
  - LBP\_InitTable, 41
  - LBP\_MAPPING\_TYPE, 36
  - LBP\_onecount, 41
  - LBP\_rotmin, 41
  - LBP\_transitions, 41
  - ReadAllShapes, 42
  - ReadCvMat, 42
  - WriteCvMat, 42
- ASMSeqSearch
  - asmfitting, 25
- Build
  - asm\_model, 6
- BuildDetectMapping
  - asmbuilding, 23
- CalcBisector
  - asm\_shape, 17
- CalcChiSquareDist
  - asmlibrary.h, 37
- CalcMahalanobisDist
  - asmlibrary.h, 37
- CalcNormalVector
  - asm\_shape, 17
- CalcProfile1D
  - asm\_profile, 10
- CalcProfileLBP
  - asm\_profile, 10
- Centralize
  - asm\_shape, 17
- classical\_tdm
  - asm\_model, 8
- Clear
  - asm\_profile, 11
  - asm\_shape, 17
- close\_camera
  - video\_camera.cpp, 45
  - video\_camera.h, 47
- close\_video
  - video\_camera.cpp, 45
  - video\_camera.h, 47
- COG
  - asm\_shape, 17
- CopyFrom
  - asm\_profile, 11
  - asm\_shape, 17
- CopyTo
  - asm\_profile, 11
  - asm\_shape, 18
- D:/asmlibrary-4.0/src/asmbuilding.h, 33
- D:/asmlibrary-4.0/src/asmfitting.h, 34
- D:/asmlibrary-4.0/src/asmlibrary.h, 35
- D:/asmlibrary-4.0/src/demo\_build.cpp, 43
- D:/asmlibrary-4.0/src/demo\_fit.cpp, 44
- D:/asmlibrary-4.0/src/video\_camera.cpp, 45
- D:/asmlibrary-4.0/src/video\_camera.h, 47
- D:/asmlibrary-4.0/src/vjfacetect.cpp, 49
- D:/asmlibrary-4.0/src/vjfacetect.h, 51
- destory\_detect\_cascade
  - vjfacetect.cpp, 49
  - vjfacetect.h, 51
- detect\_all\_faces
  - vjfacetect.cpp, 49
  - vjfacetect.h, 51
- detect\_func
  - asmlibrary.h, 36
- detect\_one\_face



- vjfacetect.cpp, [49](#)
  - vjfacetect.h, [51](#)
- Draw
  - asmfitting, [25](#)
- DrawEdges
  - asmlibrary.h, [37](#)
- DrawPoints
  - asmlibrary.h, [37](#)
- Fit
  - asm\_model, [7](#)
- Fitting
  - asmfitting, [26](#)
- Fitting2
  - asmfitting, [26](#)
- free\_shape\_memeory
  - vjfacetect.cpp, [50](#)
  - vjfacetect.h, [52](#)
- GetBilinearPixel
  - asmlibrary.h, [38](#)
- GetData
  - asm\_profile, [11](#)
- GetHeight
  - asm\_shape, [18](#)
- GetLeftRight
  - asm\_shape, [18](#)
- GetMappingDetShape
  - asmfitting, [26](#)
- GetMeanFaceWidth
  - asmfitting, [26](#)
- GetMeanShape
  - asm\_model, [7](#)
- GetModesOfModel
  - asm\_model, [7](#)
- GetNorm2
  - asm\_shape, [18](#)
- GetOriPixel
  - asmlibrary.h, [38](#)
- GetProfile
  - asm\_profile, [11](#)
- GetReferenceWidthOfFace
  - asm\_model, [7](#)
- GetWidth
  - asm\_shape, [18](#)
- GetX
  - asmlibrary.h, [38](#)
- GetY
  - asmlibrary.h, [39](#)
- init\_detect\_cascade
  - vjfacetect.cpp, [50](#)
  - vjfacetect.h, [52](#)
- InitShapeFromDetBox
  - asmlibrary.h, [39](#)
- LBP\_CalcFeatureVector
  - asmlibrary.h, [39](#)
- LBP\_CalcTransformedImage
  - asmlibrary.h, [39](#)
- lbp\_circle\_table, [28](#)
  - multipliers, [28](#)
  - nsamples, [28](#)
  - offsets, [28](#)
  - points, [28](#)
- LBP\_FreeMapping
  - asmlibrary.h, [40](#)
- LBP\_FreeTable
  - asmlibrary.h, [40](#)
- LBP\_GetMapSize
  - asmlibrary.h, [40](#)
- LBP\_InitMapping
  - asmlibrary.h, [40](#)
- LBP\_InitTable
  - asmlibrary.h, [41](#)
- LBP\_MAPPING\_TYPE
  - asmlibrary.h, [36](#)
- LBP\_onecount
  - asmlibrary.h, [41](#)
- LBP\_rotmin
  - asmlibrary.h, [41](#)
- lbp\_tdm
  - asm\_model, [8](#)
- LBP\_transitions
  - asmlibrary.h, [41](#)
- left
  - scale\_param, [32](#)
- m\_asm\_meanprofile
  - profile\_lbp\_model, [29](#)
  - profile\_Nd\_model, [31](#)
- m\_buffer
  - profile\_Nd\_model, [31](#)
- m\_G
  - profile\_Nd\_model, [31](#)
- m\_P
  - profile\_Nd\_model, [31](#)
- mapping
  - profile\_lbp\_model, [29](#)
- MaxX
  - asm\_shape, [18](#)
- MaxY
  - asm\_shape, [18](#)
- MinX
  - asm\_shape, [19](#)
- MinY
  - asm\_shape, [19](#)
- multipliers

- lbp\_circle\_table, 28
- nbins
  - profile\_lbp\_model, 29
- nblocklength
  - profile\_lbp\_model, 29
- NLength
  - asm\_profile, 12
- nlevels
  - profile\_lbp\_model, 29
- Normalize
  - asm\_profile, 12
  - asm\_shape, 19
- NPoints
  - asm\_shape, 19
- nsamples
  - lbp\_circle\_table, 28
  - profile\_lbp\_model, 29
- offsets
  - lbp\_circle\_table, 28
- open\_camera
  - video\_camera.cpp, 45
  - video\_camera.h, 47
- open\_video
  - video\_camera.cpp, 45
  - video\_camera.h, 47
- operator\*
  - asm\_profile, 12
  - asm\_shape, 19
- operator\*=
  - asm\_profile, 12
  - asm\_shape, 19
- operator+
  - asm\_profile, 12
  - asm\_shape, 19
- operator+=
  - asm\_profile, 12
  - asm\_shape, 19
- operator-
  - asm\_profile, 12
  - asm\_shape, 19
- operator-=
  - asm\_profile, 12
  - asm\_shape, 20
- operator/
  - asm\_profile, 12
  - asm\_shape, 20
- operator/=
  - asm\_profile, 13
  - asm\_shape, 20
- operator=
  - asm\_profile, 13
  - asm\_shape, 20
- points
  - lbp\_circle\_table, 28
- predicate
  - profile\_lbp\_model, 29
- profile\_lbp\_model, 29
  - m\_asm\_meanprofile, 29
  - mapping, 29
  - nbins, 29
  - nblocklength, 29
  - nlevels, 29
  - nsamples, 29
  - predicate, 29
  - table, 30
  - type, 30
- profile\_Nd\_model, 31
  - m\_asm\_meanprofile, 31
  - m\_buffer, 31
  - m\_G, 31
  - m\_P, 31
- Read
  - asm\_profile, 13
  - asm\_shape, 20
  - asmfitting, 26
- read\_from\_camera
  - video\_camera.cpp, 46
  - video\_camera.h, 48
- read\_from\_video
  - video\_camera.cpp, 46
  - video\_camera.h, 48
- ReadAllShapes
  - asmlibrary.h, 42
- ReadAnnotations
  - asm\_shape, 21
- ReadCvMat
  - asmlibrary.h, 42
- ReadFromASF
  - asm\_shape, 21
- ReadFromPTS
  - asm\_shape, 21
- ReadModel
  - asm\_model, 7
- Resize
  - asm\_profile, 13
  - asm\_shape, 21
- right
  - scale\_param, 32
- Rotate
  - asm\_shape, 21
- Scale
  - asm\_shape, 21
- scale\_param, 32
  - left, 32

- right, [32](#)
- ScaleXY
  - asm\_shape, [22](#)
- table
  - profile\_lbp\_model, [30](#)
- Train
  - asmbuilding, [23](#)
- TransformPose
  - asm\_shape, [22](#)
- Translate
  - asm\_shape, [22](#)
- type
  - profile\_lbp\_model, [30](#)
- video\_camera.cpp
  - close\_camera, [45](#)
  - close\_video, [45](#)
  - open\_camera, [45](#)
  - open\_video, [45](#)
  - read\_from\_camera, [46](#)
  - read\_from\_video, [46](#)
- video\_camera.h
  - close\_camera, [47](#)
  - close\_video, [47](#)
  - open\_camera, [47](#)
  - open\_video, [47](#)
  - read\_from\_camera, [48](#)
  - read\_from\_video, [48](#)
- vjfacedetect.cpp
  - destory\_detect\_cascade, [49](#)
  - detect\_all\_faces, [49](#)
  - detect\_one\_face, [49](#)
  - free\_shape\_memeory, [50](#)
  - init\_detect\_cascade, [50](#)
- vjfacedetect.h
  - destory\_detect\_cascade, [51](#)
  - detect\_all\_faces, [51](#)
  - detect\_one\_face, [51](#)
  - free\_shape\_memeory, [52](#)
  - init\_detect\_cascade, [52](#)
- Write
  - asm\_profile, [14](#)
  - asm\_shape, [22](#)
  - asmbuilding, [24](#)
- WriteCvMat
  - asmlibrary.h, [42](#)
- WriteModel
  - asm\_model, [7](#)