Active Appearance Models Application Programmers Interface

AAM-API

Mikkel B. Stegmann

http://www.imm.dtu.dk/~aam/

Informatics and Mathematical Modelling
Technical University of Denmark

April 24, 2003

Contents

1 class **CAAMAnalyzeSynthesize**

Abstract base class for the analyze/synthesize classes.

Inheritance

- 1 CAAMAnalyzeSynthesize
  - 3 CAAMAnalyzeSynthesizeSoftware
  - 2 CAAMAnalyzeSynthesizeOpenGL

Public Members

This class serves as an abstract base class for the analyze/synthesize classes so that both the software andhardware implementation will have a common interface.

To core task of such classes is to sample pixels inside a shape placed on an image (the analyze part) and render a texture vector into a shape given in image coordinates (the synthesize part).

Author: Mikkel B. Stegmann
Version: 6-7-2002

1.1 void **FromFile** (FILE* fh)

Reads an object from disk.

Reads an object form disk. Don't use this function directly. Use AAMLoadAnalyzerSynthesizer.

Return Value: `Nothing.`
Parameters: `fh` Open file handle.
See Also: AAMLoadAnalyzerSynthesizer
Author: Mikkel B. Stegmann
Version: 6-13-2002

1.2

void **ToFile** (FILE* fh) const

Writes an object to disk.

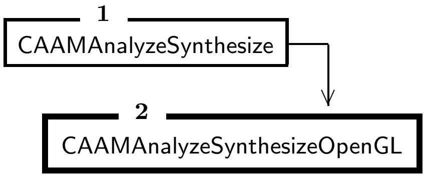
Writes an object to disk.

Return Value: `Nothing.`
Parameters: `fh` Open file handle.
See Also: AAMLoadAnalyzerSynthesizer
Author: Mikkel B. Stegmann
Version: 6-13-2002

2 class **CAAMAnalyzeSynthesizeOpenGL** : public CAAMAnalyzeSynthesize

Hardware implementation of the analysis and synthesis functions.

Inheritance



Public Members

This class utilises the presence of an OpenGL compliant graphics board to carry out analysis and synthesis.

The board should supprot certain OpenGL extensions. However, this requirement can easily be switched of inside the methods. Refer to the actual code to do this.

However, a reasonable new GPU will support the used extensions, e.g. a GeForce 2 MX will do. And to be honest, turning of the need for extentions can easily make the hardware warping *slower* than the software warping.

If the OpenGL drivers are really crappy it can actually be slower even without turning the extensions off. So remember to do some benchmarking on your system before permenantly switching to OpenGL warping.

See Also: CAAMAnalyzeSynthesize
Author: Mikkel B. Stegmann
Version: 6-12-2002

2.1 CAAMAnalyzeSynthesizeOpenGL (const CAAMReferenceFrame &rf)

Constructor.

Constructor.

Return Value: `Nothing.`
Parameters: `rf` The reference frame to analyze and synthesize through.
Author: Mikkel B. Stegmann
Version: 6-7-2002

2.2 ~CAAMAnalyzeSynthesizeOpenGL ()

Destructor.

Destructor.

Return Value:	`Nothing.`
Author:	Mikkel B. Stegmann
Version:	6-7-2002

2.3

bool Analyze (const CAAMShape &shape, CDMultiBand<TAAMPixel> &refImg, const bool useInterpolation) const

Samples the texture under a given shape.

This method samples the image intensities under a user- supplied shape into a texture vector.

The image is set using the SetAnalyzeImage call.

If a texture vector of the shape is needed instead call the alternative form of Analyze.

Return Value:	`True` is the shape is inside the image.	
Parameters:	`shape`	A shape in mage coordinates.
	`refImg`	Output reference image.
	`useInterpolation`	If true bilinear interpolation is used (default).Otherwise the faster nearest neighbor interpolationis used.NOTE: This flag is actually ignored currently inthis OpenGL implementation.
See Also:	SetAnalyzeImage	
Author:	Mikkel B. Stegmann	
Version:	6-7-2002	

2.4

bool Analyze (const CAAMShape &shape, CDVector &texture, const bool useInterpolation) const

Samples the texture under a given shape.

This method samples the image intensities under a user- supplied shape into a texture vector.

The image is set using the SetAnalyzeImage call.

If a reference image of the shape is needed instead call the alternative form of Analyze.

Return Value:	`True` is the shape is inside the image.	
Parameters:	`shape`	A shape in mage coordinates.
	`texture`	Output texture vector.
	`useInterpolation`	If true bilinear interpolation is used (default).Otherwise the faster nearest neighbor interpolationis used.NOTE: This flag is actually ignored currently inthis OpenGL implementation.
See Also:	SetAnalyzeImage	
Author:	Mikkel B. Stegmann	
Version:	6-7-2002	

2.5

void **SetupPBuffer** (const int w, const int h, PBuffer** pPBuffer)

Constructs a p-buffer and sets up various OpenGL stuff.

This method constructs a p-buffer and sets up various OpenGL stuff. Credit goes to NVidea for the p-buffer code.

Return Value:	`Nothing.`	
Parameters:	`w`	P-buffer width.
	`h`	P-buffer height.
	`pPBuffer`	Pointer to a pointer of the newly created p-buffer.
Author:	Mikkel B. Stegmann	
Version:	6-7-2002	

2.6

void **ClockReadPixels** ()

Benchmarks the glReadPixels call.

This function tries to find the fastest form of the glReadPixels(). This is typically *very* driver/card-specific.

Results are written to stdout.

Return Value:	`Nothing.`
Author:	Mikkel B. Stegmann
Version:	6-11-2002

2.7

void **ExtractChannel** (const int pixelSize, const int channelNo, unsigned char* pPixels, CDMultiBand<TAAMPixel> &dest)

Extracts a channel (band) from an RGB/RGBA image.

Extracts a channel (band) from an RGB/RGBA image.

Return Value: `Nothing.`

Parameters:

`pixelSize`	The size of a pixel in bytes.
`channelNo`	The channel to extract (zero is the first).
`pPixels`	Pointer to a continuous pixel-array(i.e. *no* row-padding etc.) of a reference image.
`dest`	Pre-allocated destination image.

Author: Mikkel B. Stegmann
Version: 6-11-2002

2.8

bool **Synthesize** (const CAAMShape &shape, const CDVector &texture, CDMultiBand<TAAMPixel> &destImage, bool renderOntoImage) const

Renders a texture vector into a shape.

This method renders a texture vector into a shape defined in image coordinates.

Return Value: `True` on success.

Parameters:

`shape`	The shape to synthesize into.
`texture`	The input texture vector in byte range [0;255].
`destImage`	Destination image
`renderOntoImage`	If true the synthesization is done on top of the existing image.

Author: Mikkel B. Stegmann
Version: 6-7-2002

2.9

void **LoadTexture** (const CDMultiBand<TAAMPixel> &textureImage, unsigned int* pTextureId)

Loads an image into the texture memory.

Loads an image into the texture memory. Expensive call.

Return Value:	`Nothing.`	
Parameters:	`textureImage`	Image to upload to the GPU.
	`pTextureId`	Destination texture id.
Author:	Mikkel B. Stegmann	
Version:	6-7-2002	

2.10

void **SetAnalyzeImage** (const CDMultiBand<TAAMPixel> &img)

Sets the image to be analyzed.

Sets the image to be analyzed. Recognize that this means that the image must be uploaded to the GPU.

Hence, during multiple calls to Analyze on the same image *avoid* calling SetAnalyze image every time!

Return Value:	`Nothing.`	
Parameters:	`img`	Image to be analyzed.
Author:	Mikkel B. Stegmann	
Version:	6-7-2002	

2.11

CAAMAnalyzeSynthesize* **Clone** (const CAAMReferenceFrame &rf) const

Clones ifself (smart way to convey type info).

Clones ifself (smart way to convey type info).

Return Value:	`A`	cloned object created on the heap.
Parameters:	`rf`	The reference frame of the cloned object.
Author:	Mikkel B. Stegmann	
Version:	6-7-2002	

2.12

bool **EnsureSize** (const int w, const int h, PBuffer** pPBuffer)

Call this to ensure that the p-buffer is large enough.

The method ensures that the p-buffer is large enough to hold and image of w * h.

Return Value: `True` the the size of 'pPBuffer' has changed.

Parameters:
- `w` Desired minimum p-buffer width.
- `h` Desired minimum p-buffer height.
- `pPBuffer` Pointer to a p-buffer pointer.

Author: Mikkel B. Stegmann

Version: 7-4-2002

2.13

void **SetImage** (const CDMultiBand<TAAMPixel> &img, CDMultiBand<TAAMPixel>** destImage, unsigned int* pTextureId)

Uploads an image to the GPU.

Uploads an image to the GPU. Don't call this directly.

This method is called by SetAnalyzeImage.

Return Value: `Nothing.`

Parameters:
- `img` Input image.
- `destImage` Pointer to an image pointer of the image actually being uploaded.
- `pTextureId` Input texture id.

See Also: SetAnalyzeImage

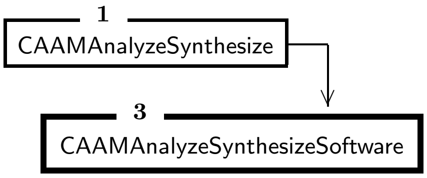
Author: Mikkel B. Stegmann

Version: 7-4-2002

3 class **CAAMAnalyzeSynthesizeSoftware** : public CAAMAnalyzeSynthesize

Software implementation of the analysis and synthesis functions.

Inheritance



Public Members

3.1		**CAAMAnalyzeSynthesizeSoftware** (const CAAMReferenceFrame &rf)	*Constructor.*	13
3.2		**~CAAMAnalyzeSynthesizeSoftware** ()	*Destructor.*	13
3.3	void	**BuildWarpTable** ()	*Cache method, that caches triangle info.*	13
3.4	void	**SetAnalyzeImage** (const CDMultiBand<TAAMPixel> &img)	*Sets the image to be analyzed.*	14
3.5	bool	**Analyze** (const CAAMShape &shape, CDVector &texture, const bool useInterpolation) const	*Samples the texture under a given shape.*	14
3.6	bool	**Analyze** (const CAAMShape &shape, CDMultiBand<TAAMPixel> &refImg, const bool useInterpolation) const	*Samples the texture under a given shape.*	15
3.7	bool	**Synthesize** (const CAAMShape &shape, const CDVector &texture, CDMultiBand<TAAMPixel> &destImage, bool renderOntoImage) const	*Renders a texture vector into a shape.*	15
3.8	CAAMAnalyzeSynthesize*	**Clone** (const CAAMReferenceFrame &rf) const	*Clones ifself (smart way to convey type info).*	15

This class contains a fairly fast software implementation of the analyze/synthesize interface.

See Also: CAAMAnalyzeSynthesize
Author: Mikkel B. Stegmann
Version: 6-12-2002

3.1

CAAMAnalyzeSynthesizeSoftware (const CAAMReferenceFrame &rf)

Constructor.

Constructor.

Return Value: `Nothing.`
Parameters: `rf` The reference frame to analyze and synthesize through.
Author: Mikkel B. Stegmann
Version: 6-12-2002

3.2

˜**CAAMAnalyzeSynthesizeSoftware** ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 6-12-2002

3.3

void **BuildWarpTable** ()

Cache method, that caches triangle info.

This method cache triangle information that does not change.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 6-12-2002

3.4

void **SetAnalyzeImage** (const CDMultiBand<TAAMPixel> &img)

Sets the image to be analyzed.

Sets the image to be analyzed. That reason for this, at first sight, somewhat clumpsy design is due to the OpenGL implementation.

Return Value:	`Nothing.`	
Parameters:	`img`	Image to be analyzed.
Author:	Mikkel B. Stegmann	
Version:	6-12-2002	

3.5

bool **Analyze** (const CAAMShape &shape, CDVector &texture, const bool useInterpolation) const

Samples the texture under a given shape.

This method samples the image intensities under a user- supplied shape into a texture vector.

The image is set using the SetAnalyzeImage call.

If a reference image of the shape is needed instead call the alternative form of Analyze.

Return Value:	`True`	is the shape is inside the image.
Parameters:	`shape`	A shape in mage coordinates.
	`texture`	Output texture vector.
	`useInterpolation`	If true bilinear interpolation is used (default).Otherwise the faster nearest neighbor interpolationis used.
See Also:	SetAnalyzeImage	
Author:	Mikkel B. Stegmann	
Version:	6-12-2002	

3.6

bool **Analyze** (const CAAMShape &shape, CDMultiBand<TAAMPixel> &refImg, const bool useInterpolation) const

Samples the texture under a given shape.

This method samples the image intensities under a user- supplied shape into a texture vector.

The image is set using the SetAnalyzeImage call.

If a texture vector of the shape is needed instead call the alternative form of Analyze.

Return Value: `True` is the shape is inside the image.

Parameters:

`shape`	A shape in mage coordinates.
`refImg`	Output reference image.
`useInterpolation`	If true bilinear interpolation is used (default).Otherwise the faster nearest neighbor interpolationis used.NOTE: This flag is actually ignored currently inthis OpenGL implementation.

See Also: SetAnalyzeImage

Author: Mikkel B. Stegmann

Version: 6-12-2002

3.7

bool **Synthesize** (const CAAMShape &shape, const CDVector &texture, CD-MultiBand<TAAMPixel> &destImage, bool renderOntoImage) const

Renders a texture vector into a shape.

This method renders a texture vector into a shape defined in image coordinates.

Return Value: `True` on success.

Parameters:

`shape`	The shape to synthesize into.
`texture`	The input texture vector in byte range [0;255].
`destImage`	Destination image
`renderOntoImage`	If true the synthesization is done on top of the existing image.

Author: Mikkel B. Stegmann

Version: 6-12-2002

3.8

CAAMAnalyzeSynthesize* **Clone** (const CAAMReferenceFrame &rf) const

Clones ifself (smart way to convey type info).

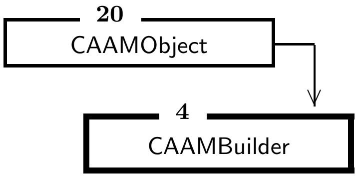
Clones ifself (smart way to convey type info).

Return Value: `A` cloned object created on the heap.
Parameters: `rf` The reference frame of the cloned object.
Author: Mikkel B. Stegmann
Version: 6-7-2002

4 class **CAAMBuilder** : public CAAMObject

Factory object that produces CAAMModel objects.

Inheritance



Public Members

4.5	void	**BuildFromFiles** (CAAMModel &model, const CString &inDir, const CString &acf, const int modelReduction, const int excludeShape)	*Diver method for model generation.*	19
4.6	void	**BuildFromFiles** (CAAMModel &model, const std::vector<CString> &asfFiles, const CString &acf, const int modelReduction, const int excludeShape)	*Diver method for model generation.*	20
4.7	bool	**LoadShapes** (const std::vector<CString> &asfFiles, CAAMShapeCollection &destination, int modelReduction, bool addCompleteImage, double addExtents, int excludeShape)	*Loads (and preprocess) all training shapes.*	21
4.8	void	**MapTextures** ()	*Performs a mapping of all textures in 'm_vTexture'.*	21
4.9	void	**DumpModelDoc** (std::vector<CDVector> bVectors)	*Write additional documentation output.*	21
4.10	void	**EstRegressionMatrices** (const std::vector<CDVector> &bVectors, const int ts_subsampling)	*Build the regression matrices for pose and parameter prediction.*	22
4.11	void	**DoPoseExperiments** (const std::vector<CDVector> &vPoseDisps, const std::vector<CDVector> &cVectors, CDMatrix &X, CDMatrix &C, const int ts_subsampling) const	*Performs a set of pose parameter displacement experiments.*	22
4.12	void	**DoCParamExperiments** (const std::vector<CDVector> &vCDisps, const std::vector<CDVector> &cVectors, CDMatrix &X, CDMatrix &C, const int ts_subsampling) const		

Protected Members

Factory object that produces CAAMModel objects. Main tasks are the estimation of parameter update matrices and verbose dumping of model information. Most other tasks are simple calls into CAAMModel, CAAMShape etc.

See Also: CAAMModel
Author: Mikkel B. Stegmann
Version: 10-26-2000

4.5

void **BuildFromFiles** (CAAMModel &model, const CString &inDir, const CString &acf, const int modelReduction, const int excludeShape)

Diver method for model generation.

This method automates the model generation as much as possible by using the various class methods for all the sequences in the task of producing a model.

Return Value:	`Nothing.`	
Parameters:	`model`	The generated model.
	`inDir`	Input directory where annotations (.asf) resides.
	`acf`	Filename of an AAM configuration file. If omitted defaults are used.
	`modelReduction`	Model reduction multiplier. Default off == 1. Useful when building multi-scale AAMs.
	`excludeShape`	Excludes one shape number 'excludeShape'from the input directory. Default -1, i.e.no shapes are removed.Used to perform leave-one-out testing.
Author:	Mikkel B. Stegmann	
Version:	4-14-2000	

4.6

void **BuildFromFiles** (CAAMModel &model, const std::vector<CString> &asfFiles, const CString &acf, const int modelReduction, const int excludeShape)

Diver method for model generation.

This method automates the model generation as much as possible by using the various class methods for all the sequences in the task of producing a model.

Return Value:	`Nothing.`	
Parameters:	`model`	The generated model.
	`asfFiles`	Vector of asf filenames.
	`acf`	Filename of an AAM configuration file. If omitted defaults are used.
	`modelReduction`	Model reduction multiplier. Default off == 1. Useful when building multi-scale AAMs.
	`excludeShape`	Excludes one shape number 'excludeShape'from the input directory. Default -1, i.e.no shapes are removed.Used to perform leave-one-out testing.
Author:	Mikkel B. Stegmann	
Version:	4-14-2000	

4.7

bool **LoadShapes** (const std::vector<CString> &asfFiles, CAAMShapeCollection &destination, int modelReduction, bool addCompleteImage, double addExtents, int excludeShape)

Loads (and preprocess) all training shapes.

Loads (and preprocess) all training shapes into a CAAMShapeCollection. This could as well be placed in CAAMUtil actually.

Return Value: `Nothing.`

Parameters:

`asfFiles`	An array of asf filenames.
`destination`	Output shape collection.
`modelReduction`	Optional size reduction. Default 1, i.e. no reduction.
`addCompleteImage`	Set this to true if you (for wird reasons) would likeadd the corners of the image to the shape. Default false.
`addExtents`	Simple and somewhat hacked way to add a shape neighborhood(will be removed in later versions).
`excludeShape`	If != -1 the the 'excludeShape'-th shape will be excluded.Zero is the first shape. Used for leave-one-out evaluation.

Author: Mikkel B. Stegmann
Version: 11-4-2002

4.8

void **MapTextures** ()

Performs a mapping of all textures in 'm_vTexture'.

Performs a mapping of all textures in 'm_vTexture' using the texture transfer class of the model.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 1-28-2002

4.9

void **DumpModelDoc** (std::vector<CDVector> bVectors)

Write additional documentation output.

Write additional documentation output.

Return Value: `Nothing.`

Parameters:

`bVectors`	Concatenated and weighted shape and texture vectorsover the training set.

Author: Mikkel B. Stegmann
Version: 1-28-2002

4.10

void **EstRegressionMatrices** (const std::vector<CDVector> &bVectors, const int ts_subsampling)

Build the regression matrices for pose and parameter prediction.

Build the regression matrices for pose and parameter prediction. I.e. calculates the member variables 'regR_c', 'regR_t'. using principal component regression.

Return Value:	`Nothing.`	
Parameters:	`bVectors`	The b-vectors for the training set the current AAM is built upon.[Can be optained from the DoCombined-PCA() call]
	`ts_subsampling`	Controls the sub sampling of the training set - i.e. to use every fifth shape to build the regression matrices upon, setshape_subsampling = 5;The motivation for doing this is reduction of model building time – and perhaps most importantly – conservation of memory resources.
Author:	Mikkel B. Stegmann	
Version:	3-14-2000	

4.11

void **DoPoseExperiments** (const std::vector<CDVector> &vPoseDisps, const std::vector<CDVector> &cVectors, CDMatrix &X, CDMatrix &C, const int ts_subsampling) const

Performs a set of pose parameter displacement experiments.

Performs a set of pose parameter displacement experiments on the training set given a set of displacment vectors.

Return Value:	`Nothing.`	
Parameters:	`vPoseDisps`	A vector of displacement vectors as obtained fromPoseDispVectors() or DisplacementSets().
	`cVectors`	The set of optimum c vectors for the training examples.
	`X`	Output matrix containing the texture difference vectorsobtained from the displacements.
	`Y`	Output matrix containing the model parameter displacementscarried out.
	`ts_subsamplingSubsampling`	factor, i.e. ts_subsampling==n will carry outdisplacements on every n-th example in the training set.
See Also:	DoPoseExperiments	
Author:	Mikkel B. Stegmann	

Version: 5-13-2002

4.12

void **DoCParamExperiments** (const std::vector<CDVector> &vCDisps, const std::vector<CDVector> &cVectors, CDMatrix &X, CDMatrix &C, const int ts_subsampling) const

Performs a set of model parameter displacement experiments.

Performs a set of model parameter displacement experiments on the training set given a set of displacment vectors.

Return Value: `Nothing.`

Parameters:

`vCDisps`	A vector of displacement vectors as obtained fromC-ParamDispVectors() or DisplacementSets().
`cVectors`	The set of optimum c vectors for the training examples.
`X`	Output matrix containing the texture difference vector-sobtained from the displacements.
`Y`	Output matrix containing the model parameter dis-placementscarried out.
`ts_subsamplingSubsampling`	factor, i.e. ts_subsampling==n will carry outdisplace-ments on every n-th example in the training set.

See Also: DoPoseExperiments
Author: Mikkel B. Stegmann
Version: 5-13-2002

4.13

void **DisplacementSets** (std::vector<CDVector> &vCDisps, std::vector<CDVector> &vPoseDisps, const std::vector<CDVector> &cVectors) const

Generates model parameter and pose displacement sets.

Generates model parameter and pose displacement sets.

Return Value: `Nothing.`

Parameters:

`vCDisps`	Resulting model parameter displacement set.
`vPoseDisps`	Resulting pose parameter displacement set.
`cVectors`	The set of c vectors over the training set.

See Also: CParamDispVectors, PoseDispVectors

Author: Mikkel B. Stegmann
Version: 3-14-2000

4.14

std::vector<CDVector> **CParamDispVectors** (const CDVector &vStdDisp, const std::vector<CDVector> &cVectors, const int pStart, const int pLen) const

Generates a set combined model parameter displacement vectors.

Generates a set combined model parameter displacement vectors where each parameter is displaced at a time according to the values in vStdDisp.

Return Value: A vector of displacement vectors.

Parameters:

`vStdDisp`	A vector of parameter displacements in standard deviations of the correspondingparameter.
`cVectors`	The set of c vectors over the training set.
`pStart`	The first parameter to displace.(default 0).
`pLen`	The number of parameters to displace.(default 0, which means all parameters).

See Also: CParamDispVectors
Author: Mikkel B. Stegmann
Version: 3-14-2000

4.15

std::vector<CDVector> **PoseDispVectors** (const CDVector &vXDisp, const CDVector &vYDisp, const CDVector &vScaleDisp, const CDVector &vRotDisp) const

Generates a set pose displacement vectors.

Generates a set pose displacement vectors.

Return Value: A vector of displacement vectors.

Parameters:

`vXDisp`	A vector of x displacements in pixels.
`vYDisp`	A vector of y displacements in pixels.
`vScaleDisp`	A vector of scale displacements (1.0=no scaling).
`vRotDisp`	A vector of rotation displacements in degrees.

See Also: CParamDispVectors

Author: Mikkel B. Stegmann
Version: 3-14-2000

4.16

void **BuildTextureVectors ()**

Samples all shapes in the training set and build the corresponding texture vectores.

Initializes the private member: 'm_vTexture' by sampling all shapes using a warp function. Note that this method calculates the mean texture.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 2-21-2000

4.17

CAAMBuilder ()

Constructor.

Constructor. Sets up default values for the settings usally given by an acf file.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 10-26-2000

4.18

~CAAMBuilder ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 10-26-2000

4.19

void CalcPixel2ShapeWeights ()

Calculates the pixel-to-shape weights.

Calculates the pixel-to-shape weights used in the combined PCA. Currently the simple 'split even' strategy is employed, i.e. normalise shape and texture variance to be equal.

Return Value:	`Nothing.`
Author:	Mikkel B. Stegmann
Version:	3-6-2000

4.20

void PoseDisplacement (const CDMultiBand<TAAMPixel> &image, const CAAMShape &shape, const CDVector &c0, const CDVector &t, CDVector &pixel_diff) const

Performs one pose regression experiment.

Performs one pose regression experiment.

Return Value:	`Nothing.`	
Parameters:	`image`	The image corresponding to the equibrilium shape.
	`shape`	The equibrilium shape.
	`c0`	The equibrilium model parameters.
	`t`	The pose displacement parameters.
	`pixel_diff`	The normalized pixel differences resulting from the pose displacement.
See Also:	ModelDisplacement	
Author:	Mikkel B. Stegmann	
Version:	3-14-2000	

4.21

void ModelDisplacement (const CDMultiBand<TAAMPixel> &image, const CAAMShape &shape, const CDVector &c0, const CDVector &delta_c, CDVector &pixel_diff) const

Performs one model parameter regression experiment.

Performs one model parameter regression experiment.

Return Value: `Nothing.`
Parameters:

`image`	The image corresponding to the equibrilium shape.
`shape`	The equibrilium shape.
`c0`	The equibrilium model parameters.
`delta_c`	The model parameter displacements.
`pixel_diff`	The normalized pixel differences resulting from the pose displacement.

See Also: PoseDisplacement
Author: Mikkel B. Stegmann
Version: 3-14-2000

4.22

void **DoShapeAlignment** (const bool fUseTangentSpace)

Aligns shapes and calc mean- and reference-shape.

Alignes shapes and calc mean- and reference-shape. I.e. initializes 'm_AlignedShapes', 'm_sMeanAShape' and 'm_sReferenceShape'.

Return Value: `Noting.`
Parameters: `fUseTangentSpace` Use the tangent space projection (bool).
Author: Mikkel B. Stegmann
Version: 10-24-2000

4.23

void **DumpPCA** (const std::vector<CDVector> &bVectors)

Dumps the PC scores of the shape, texture and combined PCA.

Dumps the PC scores of the shape, texture and combined PCA. These are written to the current directory in Matlab format as shape_pc.m, texture_pc.m and combined_pc, respectively.

param bVectors The b-parameters for all training examples. As obtained from DoCombinedPCA().

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 5-17-2000

4.24

void NormalizeTextureVectors ()

Iterative normalization of the texture samples.

Performs normalization of the texture vectors as described by Cootes et al. in "Active Appearance Models" sec. 2. Recalculates the mean texture.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 2-22-2000

4.25

bool ReadACF (const CString &filename)

Reads and parses an ACF file.

Reads and parses an ACF file. In this way the AAM can be configured using different setting for model generation. Note that all parsing is very primitive and in no way robust :-(So be careful about the configuration files.

Return Value: `true` on success, false on file errors.
Parameters: `filename` The acf file to open.
Author: Mikkel B. Stegmann
Version: 4-17-2000

4.26

void RecalcMeanTexture ( )

Recalculates the mean texture vector 'm_vMeanTexture'.

Recalculates the mean texture vector 'm_vMeanTexture'.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 2-22-2000

4.27

std::vector<CDVector> **DoCombinedPCA** ()

Performs PCA on shape and texture data.

Performs principal component analysis on the shape and the texture data. Uses the Eckhart-Young theorem described in appendix A of "Statistical Models of Appearance of Computer Vision" by T.F. Cootes et al. if we have fewer samples than dimensions which is typically the case.

Return Value: `The` b-parameters for all training examples. The format isa vector of b-parameter vectors. This vector is used in theBuildRegressionMatrices() call.
Author: Mikkel B. Stegmann
Version: 2-23-2000

4.28

void **EstPredictionMatrices** (const std::vector<CDVector> &bVectors, const int ts_subsampling)

Build the prediction matrices for pose and parameter prediction.

Build the prediction matrices for pose and parameter prediction. I.e. calculates the member variables 'regR_c', 'regR_t'. using estimates of the gradient matrices.

Return Value: `Nothing.`
Parameters:

`bVectors`	The b-vectors for the training set the current AAM is built upon.[Can be optained from the DoCombined-PCA() call]
`ts_subsampling`	Controls the sub sampling of the training set - i.e. to use every fifth shape to build the regression matrices upon, setshape_subsampling = 5;The motivation for doing this is reduction of model building time.

Author: Mikkel B. Stegmann
Version: 7-18-2002

4.29

void **EstPoseGradientMatrix** (const std::vector<CDVector> &vPoseDisps, const std::vector<CDVector> &cVectors, CDMatrix &Gpose, const int ts_subsampling) const

Estimates the Jacobian of the pose parameters.

Estimates the Jacobian of the pose parameters given a set of displacement vectors and the optimum model parameters for the training set.

Return Value:	`Nothing.`	
Parameters:	`vPoseDisps`	A vector of displacement vectors as obtained fromPoseDispVectors() or DisplacementSets().
	`cVectors`	The set of optimum c vectors for the training examples.
	`Gpose`	The output Jacobian matrix (or gradient matrix if you like).
	`ts_subsamplingSubsampling`	factor, i.e. ts_subsampling==n will carry outdisplacements on every n-th example in the training set.
See Also:	EstCParamGradientMatrix	
Author:	Mikkel B. Stegmann	
Version:	7-18-2002	

4.30

void **EstCParamGradientMatrix** (const std::vector<CDVector> &vCDisps, const std::vector<CDVector> &cVectors, CDMatrix &Gparam, const int ts_subsampling) const

Estimates the Jacobian of the model parameters.

Estimates the Jacobian of the model parameters given a set of displacement vectors and the optimum model parameters for the training set.

Return Value:	`Nothing.`	
Parameters:	`vCDisps`	A vector of displacement vectors as obtained fromCParamDispVectors() or DisplacementSets().
	`cVectors`	The set of optimum c vectors for the training examples.
	`Gparam`	The output Jacobian matrix (or gradient matrix if you like).
	`ts_subsamplingSubsampling`	factor, i.e. ts_subsampling==n will carry outdisplacements on every n-th example in the training set.
See Also:	EstPoseGradientMatrix	
Author:	Mikkel B. Stegmann	
Version:	7-18-2002	

4.1

CAAMShapeCollection **m_Shapes**

The raw (unaligned) shapes

The raw (unaligned) shapes

4.2

CAAMShapeCollection **m_AlignedShapes**

The aligned shapes

The aligned shapes

4.3

std::vector<CDVector> **m_vTexture**

The texture samples

The texture samples

4.4

CAAMModel* **m_pModel**

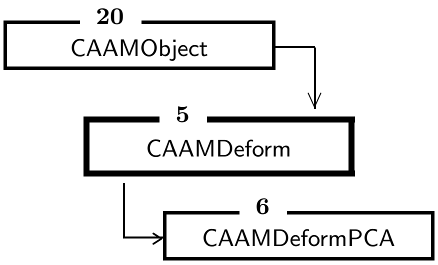
Pointer to the model we're building

Pointer to the model we're building

5 class **CAAMDeform** : public CAAMObject

Abstract deformation basis.

Inheritance



Protected Members

Abstract deformation basis.

Author: Mikkel B. Stegmann
Version: 10-17-2000

5.1 bool **m_bValid**

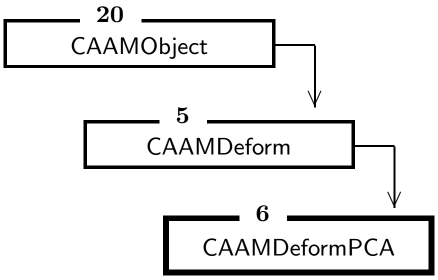
Flags that signals everything has been setup correctly

Flags that signals everything has been setup correctly

6 class **CAAMDeformPCA** : public CAAMDeform

Performs Principal Component Analysis on a set of data vectors.

Inheritance



Public Members

6.1	inline CDVector&	**EigenValues** ()	*Returns the eigen values in vector form*	34
6.2	inline const CDVector&	**EigenValues** () const	*Returns the eigen values in const vector form*	35
6.3	inline const CDVector&	**EigenValuesOrg** () const	*Returns the orignal eigen values in const vector form*	35
6.4	inline CDMatrix&	**EigenVectors** ()	*Returns the eigen vectors in matrix form*	35
6.5	inline const CDMatrix&	**EigenVectors** () const	*Returns the eigen vectors in const matrix form*	35
6.6	inline int	**NParameters** () const	*Returns the number of princal parameters*	36
6.7	inline int	**NParametersOrg** () const	*Returns the number of princal parameters before any truncation*	36
6.8	void	**ClearDataItems** ()	*Deletes the data matrix*	36
6.9	const int	**NDataItems** () const	*Returns the number of data items this basis is based on*	36
6.10	double	**ParameterWeight** (const int i, bool asPercentage) const	*Returns the eigenvalue of the i-th parameter.*	37
6.11	double	**ParameterWeightOrg** (const int i, bool asPercentage) const	*Returns the eigenvalue of the i-th parameter.*	37
6.12	void	**InsertDataItem** (const CDVector &v)	*Inserts a data vector.*	37
6.13	void	**UseIdentityTransformation** ()	*Makes the object use an identity basis instead of PCA basis.*	38
6.14	void	**DoPCA** (bool bWriteCoVarMarix)		

Performs Principal Component Analysis on a set of data vectors. The PCA basis can then be used for e.g. shape deformation.

See Also: CAAMDeform
Author: Mikkel B. Stegmann
Version: 10-17-2000

6.1

inline CDVector& **EigenValues** ()

Returns the eigen values in vector form

Returns the eigen values in vector form

6.2

inline const CDVector& **EigenValues** () const

Returns the eigen values in const vector form

Returns the eigen values in const vector form

6.3

inline const CDVector& **EigenValuesOrg** () const

Returns the orignal eigen values in const vector form

Returns the orignal eigen values in const vector form

6.4

inline CDMatrix& **EigenVectors** ()

Returns the eigen vectors in matrix form

Returns the eigen vectors in matrix form

6.5

inline const CDMatrix& **EigenVectors** () const

Returns the eigen vectors in const matrix form

Returns the eigen vectors in const matrix form

6.6

inline int **NParameters** () const

Returns the number of princal parameters

Returns the number of princal parameters

6.7

inline int **NParametersOrg** () const

Returns the number of princal parameters before any truncation

Returns the number of princal parameters before any truncation

6.8

void **ClearDataItems** ()

Deletes the data matrix

Deletes the data matrix

6.9

const int **NDataItems** () const

Returns the number of data items this basis is based on

Returns the number of data items this basis is based on

6.10

double **ParameterWeight** (const int i, bool asPercentage) const

Returns the eigenvalue of the i-th parameter.

Returns the eigenvalue of the i-th parameter in absolute numbers or as percentage.

Return Value:	`The` eigenvalue.
See Also:	ParameterWeightOrg
Author:	Mikkel B. Stegmann
Version:	10-17-2000

6.11

double **ParameterWeightOrg** (const int i, bool asPercentage) const

Returns the eigenvalue of the i-th parameter.

Returns the eigenvalue of the i-th parameter in absolute numbers or as percentage.

Return Value:	`The` eigenvalue.
See Also:	ParameterWeightOrg
Author:	Mikkel B. Stegmann
Version:	10-17-2000

6.12

void **InsertDataItem** (const CDVector &v)

Inserts a data vector.

Inserts a data vector.

Return Value:	`Nothing.`
Parameters:	`v` A data vector.
Author:	Mikkel B. Stegmann
Version:	10-17-2000

6.13

void **UseIdentityTransformation** ()

Makes the object use an identity basis instead of PCA basis.

Makes the object use an identity basis instead of PCA basis, i.e. essentially a by-pass of this object.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 10-30-2002

6.14

void **DoPCA** (bool bWriteCoVarMarix)

Performs the principal component analysis.

Performs the principal component analysis on the data items. Uses the Eckhart-Young theorem if necessary for reduced memory and computational requirements.

Return Value: `Nothing.`
Parameters: `bWriteCoVarMarix` If true the covariance matrix is writtento the current dir.NOTE: Only in the case of more data itemsthan dimensions (samples).
Author: Mikkel B. Stegmann
Version: 10-17-2000

6.15

const CDVector& **MeanDataItem** () const

Returns the sample mean.

Returns the mean of all sample vectors.

Return Value: `A` mean vector.
Author: Mikkel B. Stegmann
Version: 10-17-2000

6.16

void ToFile (FILE* fh) const

Writes a PCA object to file.

Writes a PCA object to a binary file.

Return Value:	`Nothing.`
Parameters:	`fh` File handle to binary file open for writing.
See Also:	FromFile
Author:	Mikkel B. Stegmann
Version:	10-18-2000

6.17

void FromFile (FILE* fh)

Reads a PCA object from file.

Reads a PCA object from a binary file.

Return Value:	`Nothing.`
Parameters:	`fh` File handle to binary file open for reading.
See Also:	ToFile
Author:	Mikkel B. Stegmann
Version:	10-18-2000

6.18

void Dump (const char* szPath) const

Writes eigen vectors and eigen values to matlab text files: pca_eigenvectors.m pca_eigenvalues.m pca_eigenvalues_org.m

Debug method that provides a human readable file dump of the object data.

Return Value:	`Nothing.` @throws VisError
Parameters:	`szPath` Path including terminating backslash where the data is dumped.
Author:	Mikkel B. Stegmann
Version:	10-18-2000

6.19

void **Deform** (const CDVector &params, CDVector &object) const

Projects a set of PC scores to the original space.

Projects a set of PC scores to the original space.

Return Value: `Nothing.`

Parameters:
`params` PC scores, i.e. the model paramerisation.
`object` Resulting projection into the original space, e.g. a shape.

Author: Mikkel B. Stegmann

Version: 10-18-2000

6.20

double **MahalanobisDistance** (const CDVector &params) const

Returns the Mahalanobis distance of a set of PCA parameters.

Returns the Mahalanobis distance of a set of PCA parameters.

Return Value: `The` Mahalanobis distance.

Parameters:
`params` A set of PCA parameters.

Author: Mikkel B. Stegmann

Version: 6-11-2000

6.21

void **ShuffleData** ()

Shuffle each dimension in all data items.

Shuffle each dimension in all data items over all observations.

Return Value: `Nothing.`

Author: Mikkel B. Stegmann

Version: 7-17-2002

6.22

int **TruncateParallel** ()

Truncate eigenvectors and eigenvalues using parallel analysis.

Truncate eigenvectors and eigenvalues using parallel analysis.

Also known as Horn's parallel analysis or Humphrey-Ilgen parallel analysis.

Assumes the data items have not been cleared.

Return Value:	`The` number of parameters in the truncated basis.
See Also:	TruncateVar
Author:	Mikkel B. Stegmann
Version:	7-15-2002

6.23

int **TruncateVar** (const double retained_variance)

Truncate eigenvectors and eigenvalues to retain 'retained_variance'.

Truncate eigenvectors and eigenvalues to retain 'retained_variance'.

I.e. if retained_variance = .95 this function calculates how many eigenvectors we need to explain 95% of the total variation in the PCA training set.

Return Value:	`The` number of parameters in the truncated basis.
Parameters:	`retained_variance` Amount of variance to retain [0<x<1].
Author:	Mikkel B. Stegmann
Version:	6-11-2000

6.24

int **CalcNParam** (const double retained_variance) const

Calulates the needed number of parameters.

Calulates the needed number of parameters to retain 'retained_variance'. I.e. if retained_variance = .95 then this function calculates how many eigenvectors we need to explain 95% of the total variation in the PCA training set.

Assumes that no eigenvalue cutoff has been done prior to the call.

Return Value: `The` number of model parameters.
Parameters: `retained_variance` Amount of variance to retain [0<x<1].
Author: Mikkel B. Stegmann
Version: 6-5-2000

6.25

void **Project** (const CDVector &obs, CDVector &param) const

Projects an observation into the PCA space.

Projects an observation into the PCA space.

Notice: Costly, due to the non-cached transpose of the eigenvectors.

Return Value: `Nothing.`
Parameters: `obs` Input observation.
`param` Output model parameters of the (possibly truncated)basis.
See Also: BackProject, Filter
Author: Mikkel B. Stegmann
Version: 3-6-2003

6.26

void **BackProject** (const CDVector &param, CDVector &synth_obs) const

Back projects a set of PCA model parameters into the original space.

Back projects a set of PCA model parameters into the original space.

Return Value: `Nothing.`
Parameters: `param` Input PCA model parameters.
`synth_obs` Synthesized output observation.
See Also: Project, Filter
Author: Mikkel B. Stegmann
Version: 3-6-2003

6.27

void **Filter** (CDVector &obs) const

Projects an observation into the PCA space and back.

Projects an observation into the PCA space and back.

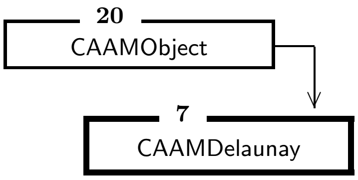
Notice: Costly, due to the non-cached transpose of the eigenvectors in the project call.

Return Value:	`Nothing.`
Parameters:	`obs` Input observation.
See Also:	Project, BackProject
Author:	Mikkel B. Stegmann
Version:	3-6-2003

7 class **CAAMDelaunay** : public CAAMObject

Delaunay Triangulator.

Inheritance



Public Members

This class act as a wrapper for the delaunay triangulator by Steven Fortune. Copyright (c) 1994 by AT&T Bell Laboratories. AT&T, Bell Laboratories.

The full diclaimer for the delaunay triangulator is:

The author of this software is Steven Fortune. Copyright (c) 1994 by AT&T Bell Laboratories. Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.

THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHORS NOR AT&T MAKE ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

See Also: CAAMMesh
Author: Mikkel B. Stegmann
Version: 02-11-2000

7.1 ~CAAMDelaunay ()

Destructor.

Return Value:	`Nothing.`
Author:	Mikkel B. Stegmann
Version:	7-27-2000

7.2

void **MakeMesh** (const CAAMShape &s, CAAMMesh &m, bool bConcaveCleanUp)

Generates a triangular mesh.

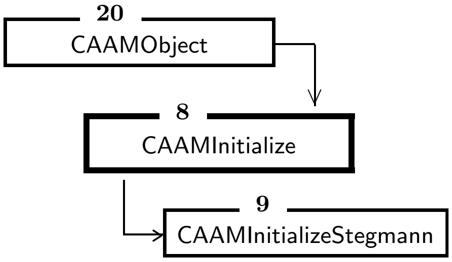
Generates a triangular mesh, where all triangles satisfies the Delaunay property.

Return Value:	`Nothing.`	
Parameters:	`s`	The input shape one wants to triangulate.
	`m`	The output Delaunay mesh.
	`bConcaveCleanUp`	If true convex triangles of concave shapesare removed (default false).
Author:	Mikkel B. Stegmann	
Version:	7-27-2000	

8 class **CAAMInitialize** : public CAAMObject

Abstract base class for all AAM initialization classes.

Inheritance



Public Members

Abstract base class for all AAM initialization classes.

Author: Mikkel B. Stegmann
Version: 7-22-2000

8.1 CAAMInitialize ()

Default constructor

Default constructor

8.2

virtual int **Initialize** (const CDMultiBand<TAAMPixel> &image, CAAMShape &s, CDVector &c)

General initialization function

General initialization function

8.3

CAAMInitialize (const CAAMModel &aammodel)

Constructor.

Constructor.

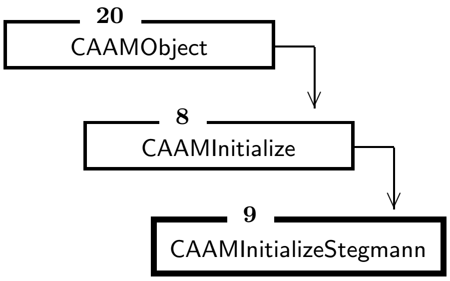
Return Value: `Nothing.`
Parameters: `model` Reference to a model.
Author: Mikkel B. Stegmann
Version: 7-22-2000

9

class **CAAMInitializeStegmann** : public CAAMInitialize

Implementation of a simple intialization method.

Inheritance



Public Members

9.3	void	**SetIterations1stPass** (const int i)	*Sets the number of interations in the coarse search*	49
9.4	void	**SetIterations2ndPass** (const int i)	*Sets the number of interations in the refinement search*	49
9.5	void	**SetXYSpacing** (const double xs, const double ys)	*Sets the spacing of the search grid in x and y*	49
9.6	void	**SetScaleSteps** (const CDVector &v)	*Sets the number of scale steps*	49
9.7	void	**SetRotationSteps** (const CDVector &v)	*Sets the number of rotation steps*	50
9.8	void	**SetModelParameterSteps** (const std::vector<CDVector> &av)	*Sets the number of model parameter steps*	50
9.9		**CAAMInitializeStegmann** (const CAAMModel &aammodel)	*Constructor.*	50
9.10	int	**Initialize** (const CDMultiBand<TAAMPixel> &image, CAAMShape &s, CDVector &c)	*Performs a somewhat brute force initialization of an AAM.*	50

Private Members

9.1	class	**CAAMInitEntry**	*Container for initialization results.*	51
9.2	class	**CAAMInitCandidates**	*Initialization candidate containiner.*	52

Implementation of a simple intialization method.

See Also: CAAMInitCandidates, CAAMInitEntry.
Author: Mikkel B. Stegmann
Version: 7-22-2000

9.3

void **SetIterations1stPass** (const int i)

Sets the number of interations in the coarse search

Sets the number of interations in the coarse search

9.4

void **SetIterations2ndPass** (const int i)

Sets the number of interations in the refinement search

Sets the number of interations in the refinement search

9.5

void **SetXYSpacing** (const double xs, const double ys)

Sets the spacing of the search grid in x and y

Sets the spacing of the search grid in x and y

9.6

void **SetScaleSteps** (const CDVector &v)

Sets the number of scale steps

Sets the number of scale steps

9.7

void **SetRotationSteps** (const CDVector &v)

Sets the number of rotation steps

Sets the number of rotation steps

9.8

void **SetModelParameterSteps** (const std::vector<CDVector> &av)

Sets the number of model parameter steps

Sets the number of model parameter steps

9.9

CAAMInitializeStegmann (const CAAMModel &aammodel)

Constructor.

Constructor.

Return Value: `Nothing.`
Parameters: `aammodel` The model to initialize.
Author: Mikkel B. Stegmann
Version: 7-22-2000

9.10

int **Initialize** (const CDMultiBand<TAAMPixel> &image, CAAMShape &s, CDVector &c)

Performs a somewhat brute force initialization of an AAM.

Performs a somewhat brute force initialization of an AAM.

Assumes one object per image as of now.

Return Value: `0` on succes, non-zero on errors.

Parameters:

`image`	The image beeing searched in.
`s`	The output shape after initialization.
`c`	The output model parameters after initialization.

Author: Mikkel B. Stegmann

Version: 7-22-2000

9.1 class **CAAMInitEntry**

Container for initialization results.

Public Members

Container for initialization results.

See Also: CAAMInitialize, CAAMInitCandidates

Author: Mikkel B. Stegmann

Version: 7-19-2000

9.1.1 **CAAMInitEntry** (const double e_fit, const CAAMShape s, const CDVector &c)

Constructor

Constructor

9.1.2

inline double **E_fit** () const

Returns the fit measure

Returns the fit measure

9.1.3

inline CAAMShape **Shape** () const

Returns the corresponding shape

Returns the corresponding shape

9.1.4

inline CDVector **C** () const

Returns the corresponding model parameters

Returns the corresponding model parameters

9.2

class **CAAMInitCandidates**

Initialization candidate containiner.

Public Members

Initialization candidate containiner. Holds the 'n' best candidates that has applieed for acceptance.

See Also: CAAMInitEntry, CAAMInitialize
Author: Mikkel B. Stegmann
Version: 7-19-2000

9.2.1

int **NCandidates** () const

Returns the number of candidates

Returns the number of candidates

9.2.2

CAAMInitEntry& **Candidate** (int i)

Returns the i-th candidate

Returns the i-th candidate

9.2.3

CAAMInitCandidates (int n)

Allocates a pool with room for 'n' candidates.

Allocates a pool with room for 'n' candidates.

Return Value: `Nothing.`
Parameters: `n` The number of candidates.
Author: Mikkel B. Stegmann
Version: 7-19-2000

9.2.4

bool **ApplyForAcceptance** (const CAAMInitEntry e)

Apply for acceptance of a new initialization hypothesis.

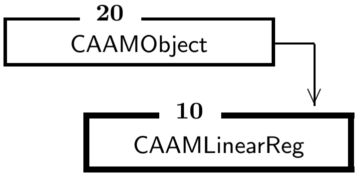
Apply for acceptance of a new initialization hypothesis. The entry are accepted if it is better than the worst candiate, or if there is less than 'n' candidates in the set.

Return Value: `True` if it is accepted, false if not.
Parameters: `e` The seach result, which is applieing for acceptance.
Author: Mikkel B. Stegmann
Version: 7-19-2000

10 class **CAAMLinearReg** : public CAAMObject

Performs multi-variate linear regression on a set of experiments.

Inheritance



Public Members

Performs multi-variate linear regression on a set of experiments.

Author: Mikkel B. Stegmann
Version: 3-15-2000

10.1 **CAAMLinearReg** ()

Constructor.

Constructor.

Return Value: `Nothning.`
Author: Mikkel B. Stegmann
Version: 3-16-2001

10.2

~CAAMLinearReg ()

Destructor.

Destructor.

Return Value:	`Nothning.`
Author:	Mikkel B. Stegmann
Version:	3-16-2001

10.3

int **EstimateK** (const CDMatrix &C, CDMatrix &EigenVec_k, CDMatrix &EigenVal_k)

Performs k-estimation.

Performs k-estimation.

Return Value:	`k`	
Parameters:	`C`	The parameter matrix.
	`EigenVec_k`	Output k eigenvectors.
	`EigenVal_k`	Output k eigenvalues.
Author:	Mikkel B. Stegmann	
Version:	3-15-2000	

10.4

int **DoRegression** (const CDMatrix &C, const CDMatrix &X, CDMatrix &R)

Calculates the regression matrix, R.

Calculates the regression matrix, R from the set of experiments in C and X, obtaining the relationship: C = RX.

NOTE: Destroys X!!!

Return Value: `k`

Parameters:

- `C` In the AAM case: the input parameters displacements.
- `X` In the AAM case: the input normalized pixel differences.
- `R` The output regression matrix.

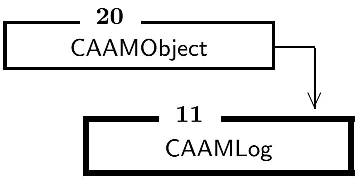
Author: Mikkel B. Stegmann

Version: 3-15-2000

11 class **CAAMLog** : public CAAMObject

Log facility class.

Inheritance



Public Members

11.7		**CAAMLog** (const char* service, const eLogLevel level)	*Default constructor.*	59
11.8	void	**InitLogFile** (const char* filename)	*Basic initialization.*	59
11.9	void	**Printf** (eLogLevel level, char* message, ...)	*Prints a message with the given severity level.*	59
11.10	void	**GetTime** (char* str)	*Returns the current (local) time in the string 'str'.*	60
11.11	void	**SetLevel** (eLogLevel level)	*Sets the minimum severity level.*	60

Private Members

11.1	static char*	**m_logfile**	*Common filname of the logfile.*	60
11.2	static FILE*	**m_file_hndl**	*Common file handle*	60
11.3	char*	**m_service**	*Name identifying the service*	61
11.4	eLogLevel	**m_level**	*Notify level*	61
11.5	bool	**m_bFileLogging**	*Used to set whether file logging should be used or not*	61
11.6	static int	**m_init**	*Keeps track of whether the info service has been initialized*	61

Log facility class. Built directly upon the CInfo class from the Verticle project by Hans P. Palbl.

Author: Hans P. Palbl & Mikkel B. Stegmann
Version: 03-05-2001

11.7

CAAMLog (const char* service, const eLogLevel level)

Default constructor.

Default constructor.

Return Value:	`Nothing.`	
Parameters:	`service`	Name of the calling class or function.
	`level`	The minimum level that will be reported.
Author:	Hans P. Palbl & Mikkel B. Stegmann	
Version:	3-5-2001	

11.8

void **InitLogFile** (const char* filename)

Basic initialization.

Basic initialization. Sets up the optional log file.

Return Value:	`Nothing.`	
Parameters:	`filename`	The log filename. If
Author:	Hans P. Palbl & Mikkel B. Stegmann	
Version:	3-5-2001	

11.9

void **Printf** (eLogLevel level, char* message, ...)

Prints a message with the given severity level.

Prints a message with the given severity level.

Return Value:	`Nothing.`	
Parameters:	`level`	Severity level of the incoming message.
	`message`	Format string in printf() format.
	`...`	Optional arguments for the printf() format string.
Author:	Hans P. Palbl & Mikkel B. Stegmann	
Version:	3-5-2001	

11.10

void **GetTime** (char* str)

Returns the current (local) time in the string 'str'.

Returns the current (local) time in the string 'str'. Format: "hh:mm:ss".

Return Value: `Nothing.`
Parameters: `str` Output string. Must be pre-allocated.
Author: Hans P. Palbl
Version: 3-5-2001

11.11

void **SetLevel** (eLogLevel level)

Sets the minimum severity level.

Sets the minimum severity level. All messages at this level and above are reported.

Return Value: `Nothing.`
Parameters: `str` Output string. Must be pre-allocated.
Author: Hans P. Palbl
Version: 3-5-2001

11.1

static char* **m_logfile**

Common filname of the logfile.

Common filname of the logfile.

11.2

static FILE* **m_file_hndl**

Common file handle

Common file handle

11.3

char* **m_service**

Name identifying the service

Name identifying the service

11.4

eLogLevel **m_level**

Notify level

Notify level

11.5

bool **m_bFileLogging**

Used to set whether file logging should be used or not

Used to set whether file logging should be used or not

11.6

static int **m_init**

Keeps track of whether the info service has been initialized

Keeps track of whether the info service has been initialized

12 class **CAAMMathUtil**

Utility math/statistical methods for the AAM project.

Public Members

This is the mathematical / statistical garbage bin.

It consists of methods that for some reason haven't found its way into other (and more meaningful) classes.

All methods are static, so there is never need for an instantiation of this class. **Author:** Mikkel B. Stegmann

Version: 1-28-2002

12.1

void **Hist** (CDVector &v, const double min, const double max, CDVector &hist, const int nbins, bool normalize, bool transform)

Calculates the histogram.

Calculates the histogram of a vector within [min;max] with optional nomalization.

Return Value: `Nothing.`

Parameters:

`v`	Input vector.
`min`	Value for the first bin (values lower are clamped to min).
`max`	Value for the last bin (values lower are clamped to max).
`hist`	Output histogram vector.
`nbins`	The number of bins. Default 256.
`normalize`	Optional normalization, i.e. make the histogram entries sum to 1.Default true.
`transform`	Optional transformation of v into 'nbins' integer intervals between [min;max].Default false.

Author: Mikkel B. Stegmann
Version: 1-25-2002

12.2

void **CumSum** (const CDVector &v, CDVector &cumsum, bool normalize)

Cumulative sum of elements.

Cumulative sum of elements.

Return Value: `Nothing.`

Parameters:

`v`	Input vector.
`cumsum`	Output vector containing the cumulative sum of elements.
`normalize`	Optional normalization of output, i.e. cumsum[end] = 1.0.Default true.

Author: Mikkel B. Stegmann
Version: 1-25-2002

12.3

double **MutualInformation** (const CDVector &v1, const CDVector &v2, int nbins)

Calulates the Mutual Information (MI) between two vectors.

Calulates the Mutual Information (MI) between two vectors. See e.g. C. Studholme et al. or Viola et al.

Return Value: `The` Mutual Information (MI).

Parameters:

- `v1` Signal 1.
- `v2` Signal 2.
- `nbins` The number of bins in the single and joint histograms.Default 256.

Author: Mikkel B. Stegmann

Version: 1-22-2002

12.4

void **GaussianHistogramMatching** (const CDVector &v, CDVector &out, const int nbins, CDVector* gaussLUT)

Maps the distibution of v into an approximate Gaussian distribution.

Maps the distibution of v into an approximate Gaussian distribution.

Return Value: `Nothing.`

Parameters:

- `v` Input vector.
- `out` Output vector.
- `nbins` Number of bins used to approx the distribution. Default 256.
- `gaussLUT` If not null, the calulated LUT is returned here.

Author: Mikkel B. Stegmann

Version: 1-25-2002

12.5

void **ExpandMatrix2DyadicSize** (const CDMatrix &m, CDMatrix &dyad)

Expands a matrix to have dyadic size.

This method expands a matrix to have dyadic size, i.e. nrows and ncols that are powers of two.

Expansion are done using zero-padding.

Return Value:	`Nothing.`	
Parameters:	`m`	Input matrix.
	`dyad`	Output dyadic version.
Author:	Mikkel B. Stegmann	
Version:	4-24-2002	

12.6

void **CalcElementVar** (const std::vector<CDVector> &vVectors, CDVector &varVec, CDVector* vpMean)

Calculates the variance of each component in a set of vectors.

Return Value:	`Nothing.`	
Parameters:	`cVectors`	Input set of vectors.
	`varVec`	A vector containing the variance of each compomentin cVectors.
	`vpMean`	Optional vector pointer to return the mean vector in.
Author:	Mikkel B. Stegmann	
Version:	3-22-2000	

12.7

void **LinearStretchMinMax** (CDVector &v, const double new_min, const double new_max)

Maps a vector linearly from [min;max] to [new_min;new_max].

Maps a vector linearly from [min;max] to [new_min;new_max].

Return Value:	`Nothing.`	
Parameters:	`v`	Input vector.
	`new_min`	Desired minimum.
	`new_max`	Desired maximum.
See Also:	LinearStretchClamp	
Author:	Mikkel B. Stegmann	
Version:	4-25-2002	

12.8

void **LinearStretchClamp** (CDVector &v, const double x1, const double x2, const double new_min, const double new_max)

Maps a vector linearly from [x1;x2] to [new_min;new_max].

Maps a vector linearly from [x1;x2] to [new_min;new_max]. Values outside [x1;x2] are clamped to x1, x2 respectively.

Return Value: `Nothing.`

Parameters:

`v`	Input vector.
`x1`	Starting point of stretch.
`x2`	Ending point of stretch.
`new_min`	Desired minimum.
`new_max`	Desired maximum.

See Also: LinearStretchMinMax
Author: Mikkel B. Stegmann
Version: 4-25-2002

12.9

void **MeanFilter** (const CDMatrix &in, CDMatrix &out)

Performs a 3x3 mean filtering of a matrix.

Performs a 3x3 mean filtering of a matrix.

Return Value: `Nothing.`

Parameters:

`in`	Input matrix.
`out`	Output mean filtered matrix.

Author: Mikkel B. Stegmann
Version: 5-15-2002

12.10

void **ZeroMeanUnitLength** (CDVector &v)

Normalises a vector to zero mean and unit length.

Normalises a vector to zero mean and unit length.

Return Value: `Nothing.` The result is returned in v.
Parameters: `v` Input vector.
Author: Mikkel B. Stegmann
Version: 6-12-2002

12.11

void **PseudoInv** (const CDMatrix &A, CDMatrix &P)

Calculates the pseudo inverse of a matrix.

Calculates the pseudo inverse of a matrix.

Return Value: `Nothing.`
Parameters:
`A` Input matrix.
`P` Pseudo inverse of A.
Author: Mikkel B. Stegmann
Version: 7-19-2002

13 class **CAAMPoint**

Point container.

Public Members

This class act as a very simple container for the concept 'a real-precision point'. It's merely a struct with a constructor. Used in the mesh and triangle representation.

See Also: CAAMTriangle, CAAMMesh
Author: Mikkel B. Stegmann
Version: 02-10-2000

13.1 double **x**

x-position

x-position

13.2 double **y**

y position

y position

13.3

CAAMPoint ()

Constructor

Constructor

13.4

CAAMPoint (const double _x, const double _y)

Constructor

Constructor

14 class CAAMTriangle

Triangle container with built-in hit test.

Public Members

Private Members

This class is a simple triangle structure defined by three indexes to a list of points. Included functionality is a hit test.

See Also: CAAMMesh, CAAMPoint.
Author: Mikkel B. Stegmann
Version: 02-10-2000

14.5 inline int V1 () const

Returns index to point 1

Returns index to point 1

14.6

inline int **V2** () const

Returns index to point 2

Returns index to point 2

14.7

inline int **V3** () const

Returns index to point 3

Returns index to point 3

14.8

CAAMTriangle (int _v1, int _v2, int _v3, std::vector<CAAMPoint>* pPoints)

Constructor.

Constructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 11-15-2000

14.9

void **Calc_dD** ()

Cache function.

Cache function.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 11-15-2000

14.10

bool **IsInside** (const CAAMPoint &p) const

Returns true if the point 'p' is inside the triangle.

Return Value: `True` if the point is inside, otherwise false.
Parameters: `p` The point to test
Author: Mikkel B. Stegmann
Version: 2-14-2000

14.11

bool **IsInside** (const CAAMPoint &p, double &alpha, double &beta, double &gamma) const

Performs a hit test on the point p. If p is inside – the position of p relative to the triangle is returned.

The relative position is:

p = alpha*p1 + beta*p2 + gamma*p3

Where p1-3 is the three points of the triangle.

Return Value: `True` if the point is inside, otherwise false.
Parameters: `p` The point to test alpha Relative x1 position.beta Relative x2 position.gamma Relative x3 position.
Author: Mikkel B. Stegmann
Version: 2-14-2000

14.12

CAAMPoint **CenterPoint** ( ) const

Returns the center point of the triangle.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-4-2000

14.13

double **Area** () const

Returns the area.

Returns the area.

Return Value: `The` area.
Author: Mikkel B. Stegmann
Version: 11-15-2000

14.1

std::vector<CAAMPoint> * **m_pPoints**

The point vector from which the triangle is defined

The point vector from which the triangle is defined

14.2

int **m_v1**

Index to point 1

Index to point 1

14.3

int **m_v2**

Index to point 2

Index to point 2

14.4

int **m_v3**

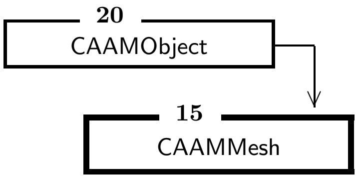
Index to point 3

Index to point 3

15 class **CAAMMesh** : public CAAMObject

2D triangular mesh container.

Inheritance



Public Members

15.3	std::vector<CAAMTriangle> &	**Triangles** ()	*Returns a vector of triangles*	76
15.4	std::vector<CAAMPoint> &	**Points** ()	*Returns a vector of points*	76
15.5	const std::vector<CAAMTriangle> &	**Triangles** () const	*Returns a const vector of triangles*	76
15.6	const std::vector<CAAMPoint> &	**Points** () const	*Returns a const vector of points*	77
15.7	inline int	**NPoints** () const	*Returns the total number of points in the mesh*	77
15.8	inline int	**NTriangles** () const	*Returns the total number of triangles in the mesh*	77
15.9	void	**Clear** ()	*Deletes all points and triangles*	77
15.10	bool	**IsInside** (const CAAMPoint &p) const		77
15.11	bool	**IsInside** (const CAAMPoint &p, int &triangle, double &alpha, double &beta, double &gamma) const	*Performs a hit test on the point p.*	78
15.12		**CAAMMesh** ()	*Constructor.*	78
15.13		**~CAAMMesh** ()	*Destructor.*	78
15.14	void	**Insert** (const CAAMPoint &p)	*Adds a point to the end.*	79
15.15	void	**ReplacePoints** (const CAAMShape &s)	*Replaces the points in the mesh.*	79
15.16	void	**Insert** (const CAAMTriangle &t)	*Adds a triangle to the end.*	79
15.17	int	**ToMatlab** (const CString& sFilename) const		80
15.18	double	**Area** () const	*Returns the total area of all triangles in the mesh.*	80
15.19	CAAMMesh&	**operator=** (const CAAMMesh &m)		

Private Members

This class implements the concept of a 2D triangular mesh. Included functionality is a hit test and matlab export capability.

Author: Mikkel B. Stegmann
Version: 02-10-2000

15.3

std::vector<CAAMTriangle> & **Triangles** ()

Returns a vector of triangles

Returns a vector of triangles

15.4

std::vector<CAAMPoint> & **Points** ()

Returns a vector of points

Returns a vector of points

15.5

const std::vector<CAAMTriangle> & **Triangles** () const

Returns a const vector of triangles

Returns a const vector of triangles

15.6

const std::vector<CAAMPoint> & **Points** () const

Returns a const vector of points

Returns a const vector of points

15.7

inline int **NPoints** () const

Returns the total number of points in the mesh

Returns the total number of points in the mesh

15.8

inline int **NTriangles** () const

Returns the total number of triangles in the mesh

Returns the total number of triangles in the mesh

15.9

void **Clear** ()

Deletes all points and triangles

Deletes all points and triangles

15.10

bool **IsInside** (const CAAMPoint &p) const

Performs a hit test on the point p.

Return Value: `True` if the point is inside, otherwise false.
Parameters: `p` The point to test
Author: Mikkel B. Stegmann
Version: 2-14-2000

15.11

bool **IsInside** (const CAAMPoint &p, int &triangle, double &alpha, double &beta, double &gamma) const

Performs a hit test on the point p.

Performs a hit test on the point p.

Return Value: `True` if the point is inside, otherwise false.

Parameters:

`p`	The point to test
`triangle`	Triangle index.
`alpha`	Relative position on triangle (barycentric coordinate).
`beta`	Relative position on triangle (barycentric coordinate).
`gamme`	Relative position on triangle (barycentric coordinate).

Author: Mikkel B. Stegmann
Version: 2-14-2000

15.12

CAAMMesh ()

Constructor.

Constructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 11-15-2000

15.13

˜CAAMMesh ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 11-15-2000

15.14

void **Insert** (const CAAMPoint &p)

Adds a point to the end.

Adds a point to the end.

Return Value: `Nothing.`
Parameters: `p` Point to add.
Author: Mikkel B. Stegmann
Version: 11-15-2000

15.15

void **ReplacePoints** (const CAAMShape &s)

Replaces the points in the mesh.

Replaces the points in the mesh with the one given by a shape. Preserves the triangles.

Return Value: `Nothing.`
Parameters: `s` Shape to fecth points from.
Author: Mikkel B. Stegmann
Version: 1-14-2003

15.16

void **Insert** (const CAAMTriangle &t)

Adds a triangle to the end.

Adds a triangle to the end.

Return Value: `Nothing.`
Parameters: `t` Triangle to add.
Author: Mikkel B. Stegmann
Version: 11-15-2000

15.17

int **ToMatlab** (const CString& sFilename) const

Writes mesh structure to a matlab file containing three vectors:

xTri X-points. yTri Y-points. Tri Triangles defined as an (ntriangles x 3) matrix. Thus each row defines a triangle using three indices pointing to the point-vectors.

Return Value: `Zero` on succes, non-zero if the mesh is empty.
Parameters: `sFilename` The filename to be written. The file isoverwritten if it already exists.
Author: Mikkel B. Stegmann
Version: 02-11-2000

15.18

double **Area** () const

Returns the total area of all triangles in the mesh.

Returns the total area of all triangles in the mesh.

Return Value: `The` area.
Author: Mikkel B. Stegmann
Version: 11-15-2000

15.19

CAAMMesh& **operator=** (const CAAMMesh &m)

Assignment operator.

Assignment operator.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 2-14-2000

15.20

CAAMMesh (const CAAMMesh &m)

Copy constructor.

Copy constructor.

Return Value: `Nothing.`
Parameters: `m` Mesh to copy.
Author: Mikkel B. Stegmann
Version: 2-14-2000

15.1

std::vector<CAAMTriangle> **m_vTriangles**

A vector of triangles

A vector of triangles

15.2

std::vector<CAAMPoint> **m_vPoints**

A vector of points

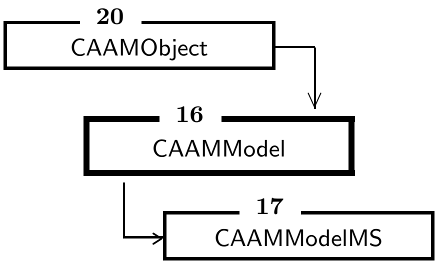
A vector of points

16

class **CAAMModel** : public CAAMObject

The core Active Apperance Model object.

Inheritance



Public Members

16.1	struct	**CAAMOptState**	*Stores interates from the optimization process.*	86
16.2	class	**CAAMOptRes**	*Stores optimization results.*	87
16.18	const CDMatrix&	**Rc** () const	*Texture parameter update prediction matrix*	89
16.19	const CDMatrix&	**Rt** () const	*Pose parameter update prediction matrix*	89
16.20	const CDMatrix&	**Qg** () const	*The texture part of the combined PCA eigenvectors*	90
16.21	const CDMatrix&	**Qs** () const	*The shape part of the combined PCA eigenvectors*	90
16.22	const CAAMReferenceFrame&	**ReferenceFrame** () const	*Returns the reference frame of the model*	90
16.23	inline int	**NBands** () const	*Returns the number of bands in the model*	90
16.24	int	**NTextureSamples** () const	*Returns the number of samples in the texture model*	90
16.25	inline bool	**IsConvexHullUsed** () const	*Returns true if the texture model is based on the convex hull*	91
16.26	inline double	**AddExtents** () const	*Returns the amount shape extents added (warning: shape extents will be remove in later versions)*	91
16.27	inline const CAAMShape&	**MeanShape** () const	*Returns the mean shape*	91
16.28	inline const CDVector&			

			Transforms the concatenated b-parameters into c-parameters.	102
16.57	bool	**WriteModel** (const CString &filename, const bool txt_only) const	*Writes the complete AAMModel to disk as a .txt and an .amf file.*	103
16.58	void	**WriteVarianceMap** (const CString &filename) const	*Plots the pixel variance into the mean shape.*	103
16.59	const CAAMShape&	**ReferenceShape** () const	*Returns the reference shape.*	103
16.60	void	**ShapeTexParam2Combined** (const std::vector<CDVector> &bVectors, std::vector<CDVector> &cVectors) const	*Converts a set of b-vectors to combined model parameters (c-vectors).*	104
16.61		**CAAMModel** (const CAAMModel &m)	*Copy constructor.*	104
16.62	CAAMModel&	**operator=** (const CAAMModel &m)	*Assignment operator.*	104
16.63	void	**Shape2Combined** (const CAAMShape &shape, const CDMultiBand<TAAMPixel> &image, CDVector &c) const	*Projects a shape into a set of c parameters.*	105
16.64	void	**Shape2Param** (const CAAMShape &shape, CDVector &b_s) const	*Projects a shape into a set of shape parameters.*	105
16.65	void	**Combined2ShapeParam** (const CDVector &c, CDVector &b_s) const	*Converts combined model parameters to shape parameters.*	105
16.66	void	**Combined2TexParam** (const CDVector &c, CDVector &b_g) const	*Converts combined model parameters to texture parameters.*	106
16.67	void	**ShapeTexParam2Combined** (const CDVector &b_s, const CDVector &b_g, CDVector &c) const	*Projects shape and texture parameters into the combined eigenspace.*	106

Protected Members

16.3	CAAMDeformPCA	**m_ShapePCA**	*The shape PCA*	106
16.4	CAAMDeformPCA	**m_TexturePCA**	*The texture PCA*	107
16.5	CAAMDeformPCA	**m_CombinedPCA**	*The combined PCA*	107
16.6	CDMatrix	**m_mShape2PixelWeights**	*The shape-to-pixel weights*	107
16.7	double	**m_dAMFVersion**	*The AMF format version*	107
16.8	CDMatrix	**m_mQsEV**	*The shape part of the combined eigenvectors*	107
16.9	CDMatrix	**m_mQgEV**	*The texture part of the combined eigenvectors*	108

The core Active Apperance Model object that hold all eigenmodels, prediction matrices etc. Build by a CAAMBuilder.

See Also: CAAMBuilder
Author: Mikkel B. Stegmann
Version: 10-19-2000

16.1

struct **CAAMOptState**

Stores interates from the optimization process.

Members

Stores interates from the optimization process.

Author: Mikkel B. Stegmann
Version: 4-10-2000

16.1.1

CAAMOptState (double e, const CAAMShape &s, const CDVector &_c, int d)

Constructor

Constructor

16.1.2

CAAMOptState& **operator=** (const CAAMOptState &os)

Assignment operator

Assignment operator

16.2

class **CAAMOptRes**

Stores optimization results.

Public Members

Private Members

Stores optimization results.

Author: Mikkel B. Stegmann
Version: 4-10-2000

16.2.4

CAAMOptRes ()

Constructor

Constructor

16.2.5

CAAMOptRes (const double maha, const int nIterations, const double sm)

Constructor

Constructor

16.2.6

inline int **NIter** () const

Returns the number of iterations

Returns the number of iterations

16.2.7

inline double **Mahalanobis** () const

Returns the Mahalanobis Distance of the model parameters

Returns the Mahalanobis Distance of the model parameters

16.2.8

inline double **SimilarityMeasure** () const

Returns the similarity measure

Returns the similarity measure

16.2.1

int **m_nIterations**

The number of iterations

The number of iterations

16.2.2

double **m_dMaha**

The Mahalanobis Distance of the model parameters

The Mahalanobis Distance of the model parameters

16.2.3

double **m_dSimilarityMeasure**

The similarity measure value at optimum

The similarity measure value at optimum

16.18

const CDMatrix& **Rc** () const

Texture parameter update prediction matrix

Texture parameter update prediction matrix

16.19

const CDMatrix& **Rt** () const

Pose parameter update prediction matrix

Pose parameter update prediction matrix

16.20

const CDMatrix& **Qg** () const

The texture part of the combined PCA eigenvectors

The texture part of the combined PCA eigenvectors

16.21

const CDMatrix& **Qs** () const

The shape part of the combined PCA eigenvectors

The shape part of the combined PCA eigenvectors

16.22

const CAAMReferenceFrame& **ReferenceFrame** () const

Returns the reference frame of the model

Returns the reference frame of the model

16.23

inline int **NBands** () const

Returns the number of bands in the model

Returns the number of bands in the model

16.24

int **NTextureSamples** () const

Returns the number of samples in the texture model

Returns the number of samples in the texture model

16.25

inline bool **IsConvexHullUsed** () const

Returns true if the texture model is based on the convex hull

Returns true if the texture model is based on the convex hull

16.26

inline double **AddExtents** () const

Returns the amount shape extents added (warning: shape extents will be remove in later versions)

Returns the amount shape extents added (warning: shape extents will be remove in later versions)

16.27

inline const CAAMShape& **MeanShape** () const

Returns the mean shape

Returns the mean shape

16.28

inline const CDVector& **MeanTexture** () const

Returns the mean texture

Returns the mean texture

16.29

inline int **ReductionFactor** () const

Returns the reduction factor of the training set that this model was built on

Returns the reduction factor of the training set that this model was built on

16.30

inline const CAAMDeformPCA& **ShapePCA** () const

Returns the shape PCA

Returns the shape PCA

16.31

inline const CAAMDeformPCA& **TexturePCA** () const

Returns the texture PCA

Returns the texture PCA

16.32

inline const CAAMDeformPCA& **CombinedPCA** () const

Returns the combined PCA

Returns the combined PCA

16.33

inline double **MeanShapeSize** () const

Returns the mean shape size, ie.

Returns the mean shape size, ie. the size of the reference shape.

16.34

CAAMModel ()

Constructor. Set up the default settings.

Constructor. Set up the default settings.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 5-16-2000

16.35

˜CAAMModel ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 5-16-2000

16.36

bool **ApproxExample** (const CString &filename, CDMultiBand<TAAMPixel> &outImg) const

Doing model approximation of an (unseen) example.

Synthesizes an unseen example by projecting the shape and (normalized) texture into c-parameter space, generating a model instance and assigning it the appropriate pose (incl. denormalization).

Return Value: `true` on success, false if the image and/or annotation could not be read.
Parameters:
`filename` The base filename of an annotation. Ex. "scan.asf"
`outImg` The output image where the model approximation has been overlaid.
Author: Mikkel B. Stegmann
Version: 3-6-2000

16.37

void **SetPoseParameterUpdateConstraints** (const CDVector &pc)

Sets constraints on the pose parameter updates.

Sets constraints on the pose parameter updates.

NOTICE: Currently, these are *not* saved along with the model and thus must be set each time a model is loaded.

Return Value: `Nothing`
Parameters: `pc` Parameter constraints (in absolute numbers).
See Also: ConstrainSearchParameters
Author: Mikkel B. Stegmann
Version: 10-30-2002

16.38

void **SetShapeParameterUpdateConstraints** (const CDVector &mean, const CDVector &sd)

Sets user-specified shape parameter update constraints.

Sets user-specified shape parameter update constraints. NOTICE: These are currently not saved with the model.

Return Value: `Nothing.`
Parameters: `mean` Some shape parameter configuration.
`sd` The standard diviations of the 'mean'.
Author: Mikkel B. Stegmann
Version: 11-4-2002

16.39

void **ConstrainSearchParameters** (CDVector &c, CDVector &pose) const

Constrain the pose and model parameters.

Constrain the pose and model parameters to be within some reasonable limits.

Return Value:	`Nothing.`	
Parameters:	`c`	The model parameters.
	`pose`	The pose parameters.
Author:	Mikkel B. Stegmann	
Version:	7-20-2000	

16.40

bool **EstimatePose** (const CDMultiBand<TAAMPixel> image, const CAAMShape &shape, CDVector &pose) const

Estimate the pose of a shape using the pose regression matrix.

Estimate the pose of a shape using the pose regression matrix.

Return Value:	`True`	is ok, false if the shape is outside the image.
Parameters:	`image`	The image to search in.
	`shape`	The shape to determine the pose from.
	`pose`	The output pose vector.
See Also:	(→, *page 239*)	
Author:	Mikkel B. Stegmann	
Version:	3-27-2000	

16.41

void **ModelImage** (const CDVector &c, CDMultiBand<TAAMPixel> &outImg, const CAAMShape* matchPose, const bool fitTexture) const

Generates a model image based on the parameters in 'c'.

Generates a model image based on the parameters in 'c' with various options.

Return Value:	`Nothing.`	
Parameters:	`c`	A set of model parameters.
	`outImg`	The output imge.
	`matchPose`	A pointer to a shape. If not NULL the modelgenerated by the set of c-parameters will bealigned wrt. pose to this shape.
	`fitTexture`	If matchPose is not NULL, the de-mapped and de-normalized texture will be fitted in a least squares sense to the texture in outImg given by the shape. Default true.

Author: Mikkel B. Stegmann
Version: 2-29-2000

16.42

void **ModelImageEx** (const CAAMShape &shape, const CDVector &texture, CDMultiBand<TAAMPixel> &outImg, const bool renderImage, const bool fitTexture) const

Generates a synthetic image from the AAM.

Generates a synthetic image from the AAM using the model parameters in 'c'.

Return Value:	`Nothing.`	
Parameters:	`shape`	The shape the texture vector should be mapped to.
	`textureSamples`	The texture vector to be warped into an image.Will be de-mapped and de-normalized.
	`outImg`	The output image (sized correctly inside thismethod, if renderInImage is false).
	`renderImage`	If true:1) outImg is expected to be allocated.2) the model is rendered with it'spose unchanged, thus the shape isexpected to lie within the outImg This option is used for drawing the final optimization into the image or to draw amodel approximation into the image 'outImg'.Default false.
	`fitTexture`	If renderImage is true, the de-mapped and de-normalized texture will be fitted in a least squares sense to the texture in outImg given by the shape. Default true.

See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 2-24-2000

16.43

void **ShapeFreeImage** (const CDVector &textureSamples, CDMultiBand<TAAMPixel> &outImg, const bool deMap) const

Generates a shape free image using a vector of texture samples.

Generates a shape free image (that is; a mean shape image) using a vector of texture samples.

Return Value:	`Nothing.`	
Parameters:	`textureSamples`	The texture vector to be warped into an image.Denormalizes (and demaps) inside this method.
	`outImg`	A reference to an output image. Resize of theimage is done inside this method.
See Also:	ModelImage	
Author:	Mikkel B. Stegmann	
Version:	2-21-2000	

16.44

void **ShapeFreeImage** (const CDVector &textureSamples, CDMatrix &m, const bool deMap, const bool normalize) const

Generates a shape free image using a vector of texture samples.

Generates a shape free image (that is; a mean shape image) using a vector of texture samples.

Return Value:	`Nothing.`	
Parameters:	`textureSamples`	The texture vector to be warped into an image.Denormalizes (and demaps) inside this method.
	`m`	A reference to an output image on matrix form. Resize of the matrix is done inside this method.
See Also:	ModelImage	
Author:	Mikkel B. Stegmann	
Version:	2-21-2000	

16.45

double **ModelEstimateTexDiff** (const CDMultiBand<TAAMPixel> &image, const CDVector &c, CAAMShape &estimate, CDVector &diff, const int similaritym, const bool useInterpolation) const

Calculates the pixel difference from a model instance and an image.

Calculates the pixel difference from a model instance and an image.

Return Value:	`The`	similarity measure.

Parameters:	`image`	The image (input)
	`c`	Model parameters (in/output).
	`estimate`	The shape estimate (in/output).
	`diff`	The pixel difference vector (output) Resized inside.
	`similaritym`	Set the used similarity measure:0 Non-normalised L_2 norm (default)1 The "Mahalanobis" distance (texture samples areregarded independent to increase performance).2 The Lorentzian error norm.3 Absolute auto correlation of the residuals.
Author:	Mikkel B. Stegmann	
Version:	3-27-2000	

16.46

void **NormalizeTexture** (CDVector &texture) const

Normalizes a texture vector.

Normalizes a texture vector.

Return Value:	`Nothing.`	
Parameters:	`texture`	Texture to be normalized.
Author:	Mikkel B. Stegmann	
Version:	3-6-2000	

16.47

CAAMModel::CAAMOptRes **OptimizeModel** (const CDMultiBand<TAAMPixel> &image, CAAMShape &s, CDVector &c, const int maxIterations, std::vector<CAAMOptState>* pOptStates, bool disableDamping) const

Performs AAM optimization of a shape.

Performs AAM optimization of a shape containing initial pose and a set of model parameters (c).

Return Value:	`The`	results of the optimization in the form of a 'CAAMOptRes' instance.

Parameters:	`image`	The image to search in.
	`s`	The initial shape (also containing the inital pose, thus;not a normalized shape). Actually only the pose of 's' is used (to align the reference shape as the initial shape).NOTE: The optimal shape is returned in 's' after execution.
	`c`	The initial model parameters. If this vector is empty, it isresized correctly and set equal to zero, thus the mean model.NOTE: The optimal model parameters are returned in 'c' after execution.
	`maxIterations`	The maximum iterations allowed.
	`pOptStates`	Optional parameter all convergence info can be returned in.See CAAMOptState.
	`disableDamping`	Disables the damping steps (default false).
Author:	Mikkel B. Stegmann	
Version:	5-7-2002	

16.48

CAAMModel::CAAMOptRes **OptimizeModelByFineTuning** (const CDMultiBand<TAAMPixel> &image, CAAMShape &s, CDVector &c, const int maxIterations, const int similaritym, const int optimizer) const

Perform general-purpose optimization of the AAM.

Perform general-purpose optimization of the AAM using simulated annealing, conjugate gradient, steepest descent, BGFS or pattern search.

Return Value:	`The` final fit.	
Parameters:	`image`	The image beeing searched in.
	`s`	The inital shape pose.
	`c`	The initial model parameters.
	`maxIterations`	The maximum allowed number of iterations.
	`similaritym`	The used similarity measure for the optimization:0 Non-normalized L_2 norm (default).1 The "Mahalanobis" distance (texture samples areregarded independent to increase performance).2 The Lorentzian error norm.
	`optimizer`	Sets the optimer to use:1 Steepest Descent (default)2 Conjugate Gradient3 Quasi-Newton, BFGS4 Pattern search5 Simulated annealing
Author:	Mikkel B. Stegmann	
Version:	7-7-2000	

16.49

bool **ReadModel** (const CString &filename)

Reads the complete AAMModel from disk.

Reads the complete AAMModel from disk.

Return Value: `true` on success, false on file errors.

Parameters:

`filename`	Input filename without any extension. E.g. if the files on disk are'model.txt' & 'model.amf' -> filename = 'model'

See Also: WriteModel
Author: Mikkel B. Stegmann
Version: 3-10-2000

16.50

int **SampleShape** (const CDMultiBand<TAAMPixel> &image, const CAAMShape &shape, CDVector &textureSamples, const bool normalize, const bool useInterpolation, const bool map) const

Builds a texture vector from an image and a shape.

Builds a texture vector from an image and a shape.

Return Value: `The` number of samples done (zero if the shape is outside the image).

Parameters:

`image`	The image to sample in.
`textureSamples`	The normalized destination texture vector.
`shape`	The shape to sample from (in image coordinates).
`normalize`	Perform normalization after sampling.Default true.
`map`	Perform mapping after sampling.Default true.

Author: Mikkel B. Stegmann
Version: 2-21-2000

16.51

void **ShapeInstance** (const CDVector &c, CAAMShape &outShape) const

Generates a shape based on a set of model parameters.

Generates a shape based on a set of model parameters.

Return Value: `Nothing.`

Parameters:

`c`	Input model parameters.
`outShape`	The generated shape (resizing are done inside this method).

Author: Mikkel B. Stegmann
Version: 2-24-2000

16.52

void **TextureInstance** (const CDVector &c, CDVector &outTexture) const

Generates a texture based on a set of model parameters.

Generates a texture based on a set of model parameters.

Return Value: `Nothing.`
Parameters:

`c`	Input model parameters.
`outShape`	The generated texture (resizing are done inside this method).

Author: Mikkel B. Stegmann
Version: 2-24-2000

16.53

void **ShapePCAInstance** (const CDVector &b_s, CAAMShape &outShape) const

Generates a shape based on a set of shape b-parameters.

Generates a shape based on a set of shape b-parameters.

Return Value: `Nothing.`
Parameters:

`b_s`	Shape parameters
`outShape`	Output shape (resized inside method).

Author: Mikkel B. Stegmann
Version: 2-28-2000

16.54

void **ShapeTex2Combined** (const CAAMShape &shape, const CDVector &texture, CDVector &c) const

Projects the shape and texture into c-space.

Projects the shape and texture into c-space i.e. the combined model parameters.

Return Value: `Nothing.`

Parameters:

`shape`	The input shape aligned to the (aligned) mean shape.
`texture`	The corresponding normalized texture.
`c`	The resulting model parameters.

Author: Mikkel B. Stegmann

Version: 3-27-2000

16.55

void **ShapeTex2Param** (const CAAMShape &shape, const CDVector &texture, CDVector &b) const

Extract the b-parameters from a shape and corresponding texture.

Extract the b-parameters from a shape and corresponding texture by inverting the shape and texture pca projection.

Assumes that the shape and texture PCA are done beforehand.

Return Value: `Nothing.`

Parameters:

`shape`	The input shape aligned to the (aligned) mean shape.
`texture`	The corresponding normalized texture.
`b`	The resulting concatenated b vector.

Author: Mikkel B. Stegmann

Version: 3-6-2000

16.56

void **ShapeTexParam2Combined** (const CDVector &b, CDVector &c) const

Transforms the concatenated b-parameters into c-parameters.

Transforms the concatenated b-parameters into combined model parameters.

Return Value: `Nothing.`

Parameters:

`b`	Concatenated shape (weighted) and texture parameters
`c`	Combined model parameters (resized inside function).

See Also: ShapeTex2Param

Author: Mikkel B. Stegmann

Version: 2-28-2000

16.57

bool **WriteModel** (const CString &filename, const bool txt_only) const

Writes the complete AAMModel to disk as a .txt and an .amf file.

Writes the complete AAMModel to disk as a .txt and an .amf file.

Return Value:	`true`	on success, false on file errors.
Parameters:	`filename`	Output filename without any extension.
	`txt_only`	If true binary model data is not written.
See Also:	ReadModel	
Author:	Mikkel B. Stegmann	
Version:	3-10-2000	

16.58

void **WriteVarianceMap** (const CString &filename) const

Plots the pixel variance into the mean shape.

Plots the variance of each pixel in the model over the training set into the mean shape and saves the image.

Return Value:	`Nothing.`	
Parameters:	`filename`	Output image filename.
Author:	Mikkel B. Stegmann	
Version:	5-4-2000	

16.59

const CAAMShape& **ReferenceShape** () const

Returns the reference shape.

Returns the reference shape where all texture sampling and comparison should be done.

The reference shape is defined as the mean shape size to mean size and moved to the fourth quadrant.

Return Value:	`The`	reference shape.
Author:	Mikkel B. Stegmann	
Version:	3-27-2000	

16.60

void **ShapeTexParam2Combined** (const std::vector<CDVector> &bVectors, std::vector<CDVector> &cVectors) const

Converts a set of b-vectors to combined model parameters (c-vectors).

Converts a set of b-vectors to combined model parameters (c-vectors).

Return Value: `Nothing.`
Parameters: `bVectors` The input b-vectors.
`cVectors` The output c-vectors.
Author: Mikkel B. Stegmann
Version: 3-22-2000

16.61

CAAMModel (const CAAMModel &m)

Copy constructor.

Copy constructor.

Return Value: `Nothing.`
Parameters: `m` Object to copy from.
Author: Mikkel B. Stegmann
Version: 10-27-2000

16.62

CAAMModel& **operator=** (const CAAMModel &m)

Assignment operator.

Assignment operator.

Return Value: `This;`
Parameters: `m` Object to copy from.
Author: Mikkel B. Stegmann
Version: 10-30-2000

16.63

void **Shape2Combined** (const CAAMShape &shape, const CDMultiBand<TAAMPixel> &image, CDVector &c) const

Projects a shape into a set of c parameters.

Projects a shape into a set of c parameters.

Return Value:	`Nothing.`	
Parameters:	`shape`	The input shape (assumed to be in abs coordinates).
	`image`	The image where the shape is placed.
	`c`	The output combined model parameters.
Author:	Mikkel B. Stegmann	
Version:	12-5-2000	

16.64

void **Shape2Param** (const CAAMShape &shape, CDVector &b_s) const

Projects a shape into a set of shape parameters.

Projects a shape into a set of shape parameters,

Return Value:	`Nothing.`	
Parameters:	`shape`	The input shape (assumed to be in abs parameters).
	`b`	The output shape model parameters.
Author:	Mikkel B. Stegmann	
Version:	12-5-2000	

16.65

void **Combined2ShapeParam** (const CDVector &c, CDVector &b_s) const

Converts combined model parameters to shape parameters.

Converts combined model parameters to shape parameters.

Return Value:	`Nothing.`	
Parameters:	`c`	Combined model parameters.
	`b_s`	Output non-weighted shape parameters.

Author: Mikkel B. Stegmann
Version: 11-4-2002

16.66

void Combined2TexParam (const CDVector &c, CDVector &b_g) const

Converts combined model parameters to texture parameters.

Converts combined model parameters to texture parameters.

Return Value: `Nothing.`
Parameters:
- `c` Combined model parameters.
- `b_g` Output texture parameters.

Author: Mikkel B. Stegmann
Version: 11-4-2002

16.67

void ShapeTexParam2Combined (const CDVector &b_s, const CDVector &b_g, CDVector &c) const

Projects shape and texture parameters into the combined eigenspace.

Projects shape and texture parameters into the combined eigenspace.

Return Value: `Nothing.`
Parameters:
- `b_s` Input shape parameters.
- `b_g` Input texture parameters.
- `c` Resulting combined model parameters.

See Also: Combined2TexParam, Combined2ShapeParam
Author: Mikkel B. Stegmann
Version: 11-4-2002

16.3

CAAMDeformPCA m_ShapePCA

The shape PCA

The shape PCA

16.4

CAAMDeformPCA **m_TexturePCA**

The texture PCA

The texture PCA

16.5

CAAMDeformPCA **m_CombinedPCA**

The combined PCA

The combined PCA

16.6

CDMatrix **m_mShape2PixelWeights**

The shape-to-pixel weights

The shape-to-pixel weights

16.7

double **m_dAMFVersion**

The AMF format version

The AMF format version

16.8

CDMatrix **m_mQsEV**

The shape part of the combined eigenvectors

The shape part of the combined eigenvectors

16.9

CDMatrix **m_mQgEV**

The texture part of the combined eigenvectors

The texture part of the combined eigenvectors

16.10

CDVector **m_vMeanTexture**

The mean texture

The mean texture

16.11

CAAMShape **m_sMeanAShape**

Cached mean shape

Cached mean shape

16.12

CDMatrix **m_mShapeInstance**

Cache matrix

Cache matrix

16.13

CDMatrix **m_mTextureInstance**

Cache matrix

Cache matrix

16.14

CDVector m_vTextureVar

The variance of each normalized pixel in the texture model

The variance of each normalized pixel in the texture model

16.15

int m_iTextureSamples

The number of texture samples in the model

The number of texture samples in the model

16.16

CDMatrix m_R_c

The parameter prediction matricies

The parameter prediction matricies

16.17

CAAMTransferFunction* m_pTextureTF

The texture transfer function

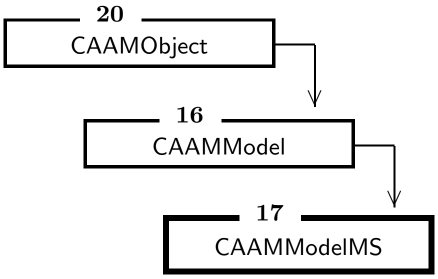
The texture transfer function

17

class **CAAMModelMS** : public CAAMModel

Multi-scale derivation of CAAModel.

Inheritance



Public Members

17.1	inline const CAAMModel&	**Model** (const int i) const	*Get the model at level i*	111
17.2	inline const int	**NLevels** () const	*Get the number of levels*	111
17.3		**CAAMModelMS** ()	*Default multi-scale constructor.*	111
17.4		**˜CAAMModelMS** ()	*multi-scale destructor.*	112
17.5	void	**BuildAllLevels** (const int nLevels, const CString &inDir, const CString &acf, const int modelReduction, const int excludeShape)	*Diver method for model generation.*	112
17.6	void	**BuildAllLevels** (const int nLevels, const std::vector<CString> &asfFiles, const CString &acf, const int modelReduction, const int excludeShape)	*Diver method for model generation.*	113
17.7	bool	**WriteModel** (const CString &filename, const bool txt_only) const	*Writes the complete multi-scale AAMModel to disk as a set of .txt and an .amf files.*	113
17.8	bool	**ReadModel** (const CString &basename)	*Reads the complete AAMModel from disk.* .	113
17.9	CAAMModel::CAAMOptRes	**OptimizeModel** (const CDMultiBand<TAAMPixel> &image, CAAMShape &s, CDVector &c, const int maxIterations, std::vector<CAAMOptState>* pOptStates, bool disableDamping) const	*Performs AAM optimization of a shape (using a model pyramid).*	114
17.10	void	**BuildPyr** (const CDMultiBand<TAAMPixel> &image) const	*Builds an image pyramid (if its not cached beforehand).*	115

Multi-scale derivation of CAAModel. Smaller models are stored in m_vModels.

See Also: CAAMModel
Author: Mikkel B. Stegmann
Version: 12-6-2002

17.1

inline const CAAMModel& **Model** (const int i) const

Get the model at level i

Get the model at level i

17.2

inline const int **NLevels** () const

Get the number of levels

Get the number of levels

17.3

CAAMModelMS ()

Default multi-scale constructor.

Default multi-scale constructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 12-6-2002

17.4

˜CAAMModelMS ()

multi-scale destructor.

multi-scale destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 12-6-2002

17.5

void **BuildAllLevels** (const int nLevels, const CString &inDir, const CString &acf, const int modelReduction, const int excludeShape)

Diver method for model generation.

This method automates the model generation as much as possible by using the various class methods for all the sequences in the task of producing a model.

Return Value: `Nothing.`

Parameters:

	`nLevels`	The number of levels the multi-scale AAM should be in.Default 3.
	`inDir`	Input directory where annotations (.asf) resides.
	`acf`	Filename of an AAM configuration file. If omitted defaults are used.
	`modelReduction`	Model reduction multiplier. Default off == 1. This sets the size of the lowest level in the model pyramid, i.e. level 0.
	`excludeShape`	Excludes one shape number 'excludeShape'from the input directory. Default -1, i.e.no shapes are removed.Used to perform leave-one-out testing.

Author: Mikkel B. Stegmann
Version: 4-14-2000

17.6

void **BuildAllLevels** (const int nLevels, const std::vector<CString> &asfFiles, const CString &acf, const int modelReduction, const int excludeShape)

Diver method for model generation.

This method automates the model generation as much as possible by using the various class methods for all the sequences in the task of producing a model.

Return Value:	`Nothing.`	
Parameters:	`nLevels`	The number of levels the multi-scale AAM should be in.Default 3.
	`asfFiles`	Vector of asf filenames.
	`acf`	Filename of an AAM configuration file. If omitted defaults are used.
	`modelReduction`	Model reduction multiplier. Default off == 1. This sets the size of the lowest level in the model pyramid, i.e. level 0.
	`excludeShape`	Excludes one shape number 'excludeShape'from the input directory. Default -1, i.e.no shapes are removed.Used to perform leave-one-out testing.
Author:	Mikkel B. Stegmann	
Version:	12-6-2002	

17.7

bool **WriteModel** (const CString &filename, const bool txt_only) const

Writes the complete multi-scale AAMModel to disk as a set of .txt and an .amf files.

Writes the complete multi-scale AAMModel to disk as a set of .txt and an .amf files.

Return Value:	`true`	on success, false on file errors.
Parameters:	`filename`	Output filename without any extension. Multi-scale prefixes will be added.
	`txt_only`	If true binary model data is not written.
See Also:	ReadModel	
Author:	Mikkel B. Stegmann	
Version:	12-6-2002	

17.8

bool **ReadModel** (const CString &basename)

Reads the complete AAMModel from disk.

Reads the complete AAMModel from disk.

Return Value: `true` on success, false on file errors.

Parameters:

`filename` Input filename without any extension and scale prefixes.E.g. if the files on disk are'model_msl<level>.txt' & 'model_msl<level>.amf' -> filename = 'model'

See Also: WriteModel

Author: Mikkel B. Stegmann

Version: 12-6-2002

17.9

CAAMModel::CAAMOptRes **OptimizeModel** (const CDMultiBand<TAAMPixel> &image, CAAMShape &s, CDVector &c, const int maxIterations, std::vector<CAAMOptState>* pOptStates, bool disableDamping) const

Performs AAM optimization of a shape (using a model pyramid).

Performs AAM optimization of a shape containing initial pose using a model pyramid.

Return Value: `The` results of the optimization in the form of a 'CAAMOptRes' instance.

Parameters:

`image`	The image to search in (in size corresponding to level zero).
`s`	The initial shape (also containing the inital pose, thus;not a normalized shape). Actually only the pose of 's' is used (to align the reference shape as the initial shape).NOTE: The optimal shape is returned in 's' after execution.NOTE: 's' is defined at level 0, i.e. shapes should have thesame size as when calling CAAMModel::OptimizeModel()directly.
`c`	The optimal model parameters at level zero. UnlikeCAAMModel::OptimizeModel() the content of this vector is notused, since the optimisation is started the smallest (i.e. top)level of the pyramid.NOTE: The optimal model parameters are returned in 'c' after execution.
`maxIterations`	The maximum iterations allowed at each level.
`pOptStates`	Optional parameter convergence info (at level zero)can be returned in. See CAAMOptState.
`disableDamping`	Disables the damping steps (default false).

Author: Mikkel B. Stegmann

Version: 12-6-2002

17.10

void **BuildPyr** (const CDMultiBand<TAAMPixel> &image) const

Builds an image pyramid (if its not cached beforehand).

Builds an image pyramid (if its not cached beforehand). The pyramid is stored in m_VImagePyr.

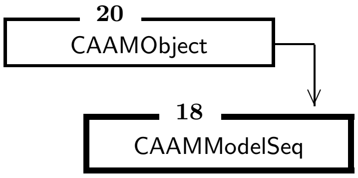
Return Value: `Nothing.`
Parameters: `image` Input image that should be convertet to a pyramid.This image is assumed to live to the *complete*usage of the pyramid, since level 0 is a pointerto this image.
Author: Mikkel B. Stegmann
Version: 12-6-2002

18

class **CAAMModelSeq** : public CAAMObject

AAM sequence object.

Inheritance



Public Members

AAM sequence object. This is a generalisation of a multi-scale model.

See Also: CAAMModel, CAAMModelMS
Author: Mikkel B. Stegmann

Version: 2-20-2003

18.1

inline const CAAMModel& **Model** (const int i) const

Get the model at level i

Get the model at level i

18.2

inline const int **NModels** () const

Get the number of models

Get the number of models

18.3

inline const CAAMModel& **FinalModel** () const

Get the final model

Get the final model

18.4

int **ModelScale** (int i) const

Returns the model reduction at that level

Returns the model reduction at that level

18.5

CAAMModelSeq ()

Default multi-scale constructor.

Default multi-scale constructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 2-20-2003

18.6

~CAAMModelSeq ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 2-20-2003

18.7

void **BuildFromSACF** (const CString &SACF, const CString &inDir, const int excludeShape)

Diver method for model generation.

This method automates the model generation as much as possible by using the various class methods for all the sequences in the task of producing a model.

Return Value: `Nothing.`

Parameters:

`SACF`	Filename of a sequence ACF, which is a file containing a list of acfs, one per line. E.g.<BOF>scale4_convex_hull.acfscale4_whiskers.acfscale4.acfscale2.acfscale1.acf<E
`inDir`	Input directory where annotations (.asf) resides.
`excludeShape`	Excludes one shape number 'excludeShape'from the input directory. Default -1, i.e.no shapes are removed.Used to perform leave-one-out testing.

Author: Mikkel B. Stegmann
Version: 2-20-2003

18.8

void **BuildFromSACF** (const CString &SACF, const std::vector<CString> &asfFiles, const int excludeShape)

Diver method for model generation.

This method automates the model generation as much as possible by using the various class methods for all the sequences in the task of producing a model.

Return Value: `Nothing.`

Parameters:

`SACF`	Filename of a sequence ACF, which is a file containing a list of acfs, one per line. E.g.<BOF>scale4_convex_hull.acfscale4_whiskers.acfscale4.acfscale2.acfscale1.acf<E
`asfFiles`	Vector of asf filenames.
`excludeShape`	Excludes one shape number 'excludeShape'from the input directory. Default -1, i.e.no shapes are removed.Used to perform leave-one-out testing.

Author: Mikkel B. Stegmann
Version: 2-20-2003

18.9

bool **WriteModels** (const CString &filename, const bool txt_only) const

Writes the sequence AAM to disk as a set of .txt and .amf files.

Writes the sequence AAM to disk as a set of .txt and .amf files. Filenames are determined from the ACF file names.

Return Value: `true` on success, false on file errors.

Parameters:

`filename`	Filename of the output .samf-file.
`txt_only`	If true, binary model data is not written.

See Also: ReadModel
Author: Mikkel B. Stegmann
Version: 2-20-2003

18.10

bool **ReadModels** (const CString &samf)

Reads the complete AAMModel from disk.

Reads the complete AAMModel from disk.

Return Value: `true` on success, false on file errors.
Parameters: `filename` Input filename (.samf).
See Also: WriteModel
Author: Mikkel B. Stegmann
Version: 2-20-2003

18.11

void **ScaleShape2Model** (const int model, CAAMShape &shape) const

Scale a shape defined in 'FinalModel' coordinates to 'model' coordinates.

Scale a shape defined in 'FinalModel' coordinates to 'model' coordinates .

Return Value:
Parameters: `model` The model the input shape are scaled to.
`shape` Input shape, which going to be scaled.
See Also: ScaleShape2Final
Author: Mikkel B. Stegmann
Version: 2-20-2003

18.12

void **ScaleShape2Final** (const int model, CAAMShape &shape) const

Scale a shape defined in 'model' coordinates to 'FinalModel' coordinates.

Scale a shape defined in 'model' coordinates to 'FinalModel' coordinates.

Return Value: `Nothing.`
Parameters: `model` The model the input shape are scaled from.
`shape` Input shape, which going to be scaled.
See Also: ScaleShape2Model
Author: Mikkel B. Stegmann
Version: 2-20-2003

19 class **CAAMMovieAVI**

Implements a simple AVI writer.

Public Members

Implements a simple AVI writer. Based on the old CVisAviStreamHandler code, but partly rewritten and extented. Credit to the VisionSDK group.

Author: Mikkel B. Stegmann
Version: 9-20-2000

19.1 **CAAMMovieAVI** (int framesPerSecond)

Return Value:
Parameters:

See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 9-20-2000

19.2 void **WriteStill** (CVisImage<CVisRGBABytePixel> image, const int duration)

Writes one image as a still in a movie.

Writes one image as a still in a movie given a certain duration.

Return Value: `Nothing.`
Parameters:

`image`	An RGBA image.
`duration`	Duration in ms.

Author: Mikkel B. Stegmann
Version: 7-22-2000

19.3

void **WriteStill** (CDMultiBand<TAAMPixel> image, const int duration)

Writes one image as a still in a movie.

Writes one image as a still in a movie given a certain duration.

Return Value: `Nothing.`
Parameters:

`image`	A grayscale image.
`duration`	Duration in ms.

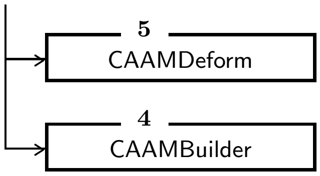
Author: Mikkel B. Stegmann
Version: 7-22-2000

20 class **CAAMObject**

Abstract base object for the AAM-API.

Inheritance

- 20 CAAMObject
  - 34 CAAMWarp
  - 31 CAAMUtil
  - 27 CAAMTransferFunction
  - 26 CAAMTest
  - 24 CAAMShapeCollection
  - 23 CAAMShape
  - 32 CAAMPropsReader
  - 21 CAAMOptimize
  - 18 CAAMModelSeq
  - 16 CAAMModel
  - 15 CAAMMesh
  - 11 CAAMLog
  - 10 CAAMLinearReg
  - 8 CAAMInitialize
  - 7 CAAMDelaunay



Public Members

Abstract base object for the AAM-API. Doesn't do much aside from devising a common way to save and dump objects.

Author: Mikkel B. Stegmann
Version: 10-19-2000

20.1

CAAMObject ()

Constructor.

Constructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 10-19-2000

20.2

˜CAAMObject ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 10-19-2000

20.3

void **ToFile** (const char* szFilename) const

Write the object to a binary file.

Write the object to a binary file. This is a very old-school load/save function. Maybe we should look into serialization some day.

Return Value: `Nothing.@throws` CVisError
Parameters: `szFilename` Destination filename.
See Also: FromFile
Author: Mikkel B. Stegmann
Version: 10-18-2000

20.4

void **ToFile** (FILE* fh) const

Write the object to a binary file.

Write the object to a binary file. This is a very old-school load/save function. Maybe we should look into serialization some day.

Return Value: `Nothing.@throws` CVisError
Parameters: `fh` Handle to an open file.
See Also: FromFile
Author: Mikkel B. Stegmann
Version: 10-18-2000

20.5

void **FromFile** (const char* szFilename)

Reads an object from a binary file.

Reads the object to a binary file. This is a very old-school load/save function. Maybe we should look into serialization some day.

Return Value: `Nothing.@throws` CVisError
Parameters: `szFilename` Destination filename.
See Also: ToFile
Author: Mikkel B. Stegmann
Version: 10-18-2000

20.6

void **FromFile** (FILE* fh)

Reads an object from a binary file.

Reads the object to a binary file. This is a very old-school load/save function. Maybe we should look into serialization some day.

Return Value: `Nothing.@throws` CVisError
Parameters: `fh` Handle to an open file.
See Also: ToFile
Author: Mikkel B. Stegmann
Version: 10-18-2000

20.7

void **Dump** (const char* szPath) const

Writes object specific data to text file(s).

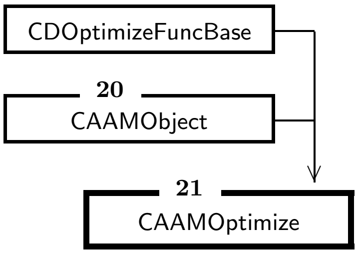
Debug method that provides a human readable file dump of the object data.

Return Value: `Nothing.@throws` CVisError
Parameters: `szPath` Path including terminating backslash where the data is dumped.
Author: Mikkel B. Stegmann
Version: 10-18-2000

21 class **CAAMOptimize** : public CDOptimizeFuncBase, CAAMObject

General purpose optimization of the AAM.

Inheritance



Public Members

21.9	inline double	**EvalFunction** (CDVector& vX)	*Fuction to be optimized.*	128
21.10	void	**OptResults** (CDVector &c, CAAMShape &s, double &fit)	*Returns the optimisation results as c, shape and error.*	128

Private Members

21.1	const CAAMModel*	**m_pModel**	*The AAM*	128
21.2	const CAAMShape*	**m_pShape**	*The initial shape pose*	129
21.3	const CDMultiBand<TAAMPixel> *	**m_pImage**	*The image where the optimization shuold be done*	129
21.4	int	**m_iSimilarityMeasure**	*The similarity measure to be used*	129
21.5	CDVector	**m_vDiff**	*The difference vector*	129
21.6	double	**m_dMinFit**	*The minimum fit*	129
21.7	CAAMShape	**m_sMinShape**	*The minimum fit shape*	130
21.8	CDVector	**m_vMinC**	*The minimum fit model parameters*	130

Class for optimizing the AAM using traditional optimization methods such as simluated annealingm, conjugated gradient, steepest descent, BGFS and pattern search.

Author: Mikkel B. Stegmann
Version: 7-8-2000

21.9

inline double **EvalFunction** (CDVector& vX)

Fuction to be optimized.

Function providing a scalar interpretation of the AAM fit based on a set of paramters.

Return Value: `The` scalar fit of the AAM.
Parameters: `vX` The independent parameters that will be optimized.The first n-4 elements constitues the normal modelparameters of the AAM. The last four are the pose parameters.
See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 7-7-2000

21.10

void **OptResults** (CDVector &c, CAAMShape &s, double &fit)

Returns the optimisation results as c, shape and error.

Returns the optimisation results as c, shape and error. This is to avoid and extra conversion after ended optimisation, and worse, an extra image sampling to get the error.

Return Value: `Nothing.`
Parameters:
`c` The optimal model parameters.
`s` The optimal shape.
`s` The optimal error.
Author: Mikkel B. Stegmann
Version: 7-7-2000

21.1

const CAAMModel* **m_pModel**

The AAM

The AAM

21.2

const CAAMShape* **m_pShape**

The initial shape pose

The initial shape pose

21.3

const CDMultiBand<TAAMPixel> * **m_pImage**

The image where the optimization shuold be done

The image where the optimization shuold be done

21.4

int **m_iSimilarityMeasure**

The similarity measure to be used

The similarity measure to be used

21.5

CDVector **m_vDiff**

The difference vector

The difference vector

21.6

double **m_dMinFit**

The minimum fit

The minimum fit

21.7

CAAMShape m_sMinShape

The minimum fit shape

The minimum fit shape

21.8

CDVector m_vMinC

The minimum fit model parameters

The minimum fit model parameters

22

class **CAAMReferenceFrame**

The geometrical reference frame (or shape-free frame).

Public Members

This class defined the geomtrical reference frame of an AAM. Hence, it is the spatial layout where all texture sampling takes place.

The main objective of this class is to provide fast conversions from a shape-free image to a texture vector and vice versa.

Author: Mikkel B. Stegmann
Version: 6-7-2002

22.1

bool **UseConvexHull** () const

Returns true of the convex hull determines the shape extent

Returns true of the convex hull determines the shape extent

22.2

const CAAMShape& **RefShape** () const

Returns the reference shape

Returns the reference shape

22.3

int **NTextureSamples** () const

Returns the number of texture samples in the texture model

Returns the number of texture samples in the texture model

22.4

const CAAMMesh& **RefMesh** () const

Returns the reference mesh

Returns the reference mesh

22.5

const CDMultiBand<TAAMPixel> & **MaskImage** () const

Returns the mask image

Returns the mask image

22.6

CAAMReferenceFrame ()

Constructor.

Constructor. Sets up an empty reference frame. Use Setup() to make such an object any useful.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.7

CAAMReferenceFrame& **operator=** (const CAAMReferenceFrame &rf)

Assignment operator.

Assignment operator.

Return Value: `Nothing.`
Parameters: `rf` Object to copy from.
Author: Mikkel B. Stegmann
Version: 6-13-2002

22.8

CAAMReferenceFrame (const CAAMReferenceFrame &rf)

Copy constructor.

Copy constructor.

Return Value: `Nothing.`
Parameters: `rf` Reference frame object to be copied by construction.
Author: Mikkel B. Stegmann
Version: 6-13-2002

22.9

void **Setup** (const CAAMShape &referenceShape, const bool useConvexHull)

Sets up the class with the reference shape.

Sets up the class with the reference shape and information about how to determine the extent of the shape.

Return Value: `Nothing.`
Parameters: `referenceShape` The reference shape.
`useConvexHull` If true the convex hull of the referenceshape is used to determine its extent.
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.10

void **ToFile** (FILE* fh) const

Writes the class data to disk.

This method writes the class data to disk. Actually it is only the reference shape and the convex hull option that is written. The rest is reconstructed during read.

Return Value: `Nothing.`
Parameters: `fh` Open file handle to a binary file.
Author: Mikkel B. Stegmann
Version: 6-13-2002

22.11

void **FromFile** (FILE* fh)

Reads the class data form disk.

This method reads the class data from disk. Actually it is only the reference shape and the convex hull option that is read. The rest is reconstructed.

Return Value: `Nothing.`
Parameters: `fh` Open file handle to a binary file.
Author: Mikkel B. Stegmann
Version: 6-13-2002

22.12

void **DebugDump** ()

Writes some of the class data to disk.

This method writes the reference shape, the mask image and a test of the Image2Vector() and Vector2image() to the current directory.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.13

~CAAMReferenceFrame ()

Destructor.

Destructor. Deletes any reference shape, mesh and mask image.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.14

void **CalcMaskImage** (const bool useConvexHull)

Calculates a reference mask image.

This method calculates a reference mask image, where the shape area is set to 255 and the background to zero. The resulting image is stored in 'm_pMaskImage'.

Return Value:	`Nothing.`	
Parameters:	`useConvexHull`	If true the convex hull is used to definethe extent of the reference shape.
Author:	Mikkel B. Stegmann	
Version:	6-7-2002	

22.15

void **CalcScanLines** ()

Calculates the scanlines of the mask image.

This method calculates the scanlines of the mask image. A scanline is a subpart of a horsontal image line that is occupied but a subpart of the reference shape.

The resulting scanlines are used for quick'n'easy conversion from spatial to vector layout (and back).

Scanlines are stored in 'm_ScanlineParts'.

Return Value:	`Nothing.`
Author:	Mikkel B. Stegmann
Version:	6-7-2002

22.16

double **RefShapeWidth** () const

Returns the width of the reference shape.

Returns the width of the reference shape.

Return Value:	`The` width.
Author:	Mikkel B. Stegmann
Version:	6-7-2002

22.17

double **RefShapeHeight** () const

Returns the height of the reference shape.

Returns the height of the reference shape.

Return Value: `The` height.
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.18

void **Image2Vector** (const CDMultiBand<TAAMPixel> &refImg, CDVector &v) const

Conversion from shape-free image to a texture vector.

This method performs a quick conversion from a shape-free image to a texture vector.

Return Value: `Nothing.`
Parameters:
`refImg` Destination reference image.
`v` Input texture vector.
See Also: Vector2Image
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.19

void **Vector2Image** (const CDVector &v, CDMultiBand<TAAMPixel> &outImg) const

Conversion from a texture vector to a shape-free image.

This method performs a quick conversion from a texture vector to a shape-free image.

Return Value: `Nothing.`
Parameters:
`refImg` Destination reference image.
`v` Input texture vector.

See Also: Vector2Matrix, Image2Vector
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.20

void **Vector2Matrix** (const CDVector &v, CDMatrix &m) const

Conversion from a texture vector to a shape-free matrix.

This method performs a quick conversion from a texture vector to a shape-free image on matrix form.

Will currently only work on single band AAMs.

Return Value: `Nothing.`
Parameters: `refImg` Destination reference image.
`v` Input texture vector.
See Also: Vector2Matrix, Image2Vector
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.21

int **RefImageWidth** () const

Returns the width of the reference image.

Returns the width of the reference image.

Return Value: `The` width.
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.22

int **RefImageHeight** () const

Returns the height of the reference image.

Returns the height of the reference image.

Return Value: `The` height.
Author: Mikkel B. Stegmann
Version: 6-7-2002

22.23

bool FindTriangle (const CAAMPoint &p, unsigned short &triangle, double &alpha, double &beta, double &gamma) const

Find the corresponding triangle in the source mesh for a point.

Find the corresponding triangle in the source mesh for a point.

Return Value: `True` if the point is inside the source mesh.
Parameters:

`p`	Input point
`triangle`	Triangle index.
`alpha`	Relative position on triangle (barycentric coordinate).
`beta`	Relative position on triangle (barycentric coordinate).
`gamma`	Relative position on triangle (barycentric coordinate).

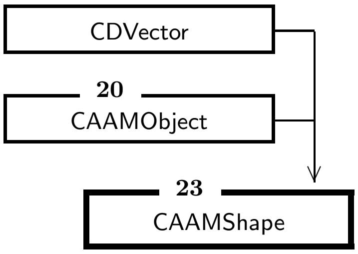
Author: Mikkel B. Stegmann
Version: 6-7-2002

23

class **CAAMShape** : public CDVector, CAAMObject

Shape container.

Inheritance



Public Members

Private Members

This class act as a container for a shape. Essentially it's just a set of 2D points stored in a vector in the format xxxyyy.

See Also: CAAMShapeCollection
Author: Mikkel B. Stegmann
Version: 02-08-2000

23.1 class **CAAMPointInfo**

Auxiliary point data.

Public Members

Auxiliary point data. Needed as a part of the enhanced shape representation extension [Which explains why it's layered upon the CAAMShape class and not merged into it :-(].

See Also: CAAMShape
Author: Mikkel B. Stegmann
Version: 4-26-2000

23.1.1

CAAMPointInfo ()

Constructor

Constructor

23.1.2

inline bool **IsOuterEdge** () const

Point info method

Point info method

23.1.3

inline bool **IsOriginal** () const

Point info method

Point info method

23.1.4

inline bool **IsClosed** () const

Point info method

Point info method

23.1.5

inline bool **IsHole** () const

Point info method

Point info method

23.1.6

inline void **SetOuterEdge** (bool enable = true)

Sets whether the point belongs to an outer edge or not

Sets whether the point belongs to an outer edge or not

23.1.7

inline void **SetOriginal** (bool enable = true)

Sets whether the point is an original model point or an artificial generated point

Sets whether the point is an original model point or an artificial generated point

23.1.8

inline void **SetClosed** (bool enable = true)

Sets whether the point belongs to a closed or opened path

Sets whether the point belongs to a closed or opened path

23.1.9

inline void **SetHole** (bool enable = true)

Sets whether the point belongs to an hole in the model

Sets whether the point belongs to an hole in the model

23.10

inline const CString& **HostImage** () const

Host image (if any)

Host image (if any)

23.11

void **SetHostImage** (const CString &hostImageFilename)

Returns the host image (if any).

Returns the host image (if any).

23.12

double **Width** () const

Shape width

Shape width

23.13

double **Height** () const

Shape height

Shape height

23.14

inline const int **NPoints** () const

The number of shape points

The number of shape points

23.15

bool **IsAbs** () const

Returns true if the shape is in absolute coordinates

Returns true if the shape is in absolute coordinates

23.16

std::vector<CAAMPointInfo> & **PointAux** ()

Returns the complete point aux vector of the shape

Returns the complete point aux vector of the shape

23.17

const std::vector<CAAMPointInfo> & **PointAux** () const

Returns the complete point aux vector of the shape

Returns the complete point aux vector of the shape

23.18

CAAMShape (int nbPoints)

Constructs a shape with 'nbPoints' points.

Constructs a shape with 'nbPoints' points. Defaults to absolute point coordinates and single closed path connectivity.

Return Value: `Nothing.`
Parameters: `nbPoints` The number of points the shape should contain.
See Also: SetClosedPathConnectivity
Author: Mikkel B. Stegmann
Version: 5-8-2000

23.19

CAAMShape (const CDVector &v)

Constructs a shape from a vector. Assumes closed path connectivity.

Constructs a shape from a vector. Defaults to absolute point coordinates and single closed path connectivity.

Return Value: `Nothing.`
Parameters: `v` The input vector.
See Also: SetClosedPathConnectivity
Author: Mikkel B. Stegmann
Version: 5-8-2000

23.20

CAAMShape (const CAAMShape &s)

Copy contructor.

Copy contructor.

Return Value: `Nothing.`
Parameters: `s` The input shape.
Author: Mikkel B. Stegmann
Version: 5-8-2000

23.21

void **SetClosedPathConnectivity** ()

Sets all point to be connected in one closed path.

Manipulates the aux point info such that the shape points is interpreted as one closed outer path defined clock-wise with repect to it's normals.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-26-2000

23.22

CDVector& **operator=** (double value)

Assignment operator (double).

Sets all x and y compoments of a shape equal to a double. [Actually only used when calculating a mean shape – since we want to accumulate in an empty shape].

Return Value: `Nothing.`
Parameters: `value` The value to set all shape points to.
Author: Mikkel B. Stegmann
Version: 4-26-2000

23.23

CAAMShape& **operator=** (const CAAMShape &s)

Assignment operator (CAAMShape).

Set one shape equal another.

Return Value: `Nothing.`
Parameters: `s` The shape to copy.
Author: Mikkel B. Stegmann
Version: 5-9-2000

23.24

void **CopyData** (const CAAMShape &s)

Copies all data from a shape to this.

Copies all data from a shape to this. Called from the assignment operator.

Return Value: `Nothing,`
Parameters: `s` Shape to copy data from.
Author: Mikkel B. Stegmann
Version: 10-20-2000

23.25

CDVector& **operator=** (const CVisDVector &vIn)

Assignment operator (CAAMShape).

Set the shape to be equal an xxx-yyy formatted vector. NOTE: this method does not manipulate any connectivity (i.e. point aux) info. So, if the shape beforehand was empty one should call SetClosedPathConnectivity() afterwards this call to obtain sensible point aux info.

Return Value: `Nothing.`
Parameters: `vIn` xxx-yyy formatted vector.
Author: Mikkel B. Stegmann
Version: 4-26-2000

23.26

~CAAMShape ()

Shape destructor.

Author: Mikkel B. Stegmann
Version: 5-15-2000

23.27

void **Rotate** (const double theta, const bool aroundCOG)

Rotates the shape.

Rotates the shape 'theta' radians.

Return Value:	`Nothing.`	
Parameters:	`theta`	Rotation angle in radians.
	`aroundCOG`	If true the rotation is being done around the cog of the shape instead of around the globalcenter.
Author:	Mikkel B. Stegmann	
Version:	3-15-2000	

23.28

void **Translate** (const CAAMPoint &p)

Translates the shape.

Return Value:	`Nothing.`	
Parameters:	`p`	The offset to translate.
Author:	Mikkel B. Stegmann	
Version:	5-15-2000	

23.29

void **Translate** (const double x, const double y)

Translates the shape.

Return Value:	`Nothing.`	
Parameters:	`x`	X-translation.
	`y`	Y-translation.
Author:	Mikkel B. Stegmann	
Version:	5-15-2000	

23.30

double **ShapeSize** () const

Returns the 2-norm of this shape centralized.

Return Value: The 2-norm.
Author: Mikkel B. Stegmann
Version: 5-15-2000

23.31

double **MinX** () const

Find the maximum x component of the shape.

Return Value: The x-maximum.
Author: Mikkel B. Stegmann
Version: 5-15-2000

23.32

double **MaxX** () const

Find the maximum x component of the shape.

Return Value: The x-maximum.
Author: Mikkel B. Stegmann
Version: 5-15-2000

23.33

double **MinY** () const

Find the minimum y component of the shape.

Return Value:	`The` y-minimum.
Author:	Mikkel B. Stegmann
Version:	5-15-2000

23.34

double **MaxY** () const

Find the maximum y component of the shape.

Return Value:	`The` y-maximum.
Author:	Mikkel B. Stegmann
Version:	5-15-2000

23.35

void **Scale** (const double s, const bool aroundCOG)

Scales the shape.

Scales the shape.

Return Value:	`Nothing.`	
Parameters:	`s`	Scale factor.
	`aroundCOG`	If true the scale is being done around the cog of the shape instead of around the globalcenter.
Author:	Mikkel B. Stegmann	
Version:	3-15-2000	

23.36

CAAMPoint **COG** () const

Calculates the center of gravity of the shape.

Calculates the center of gravity of the shape (actually it's the center of the centroid).

Return Value: `Nothing.`
Parameters: `p` cog output.
Author: Mikkel B. Stegmann
Version: 3-15-2000

23.37

void **COG** (double &x, double &y) const

Calculates the center of gravity of the shape.

Calculates the center of gravity of the shape (actually it's the center of the centroid).

Return Value: `Nothing.`
Parameters: `x` X cog output.
`y` X cog output.
Author: Mikkel B. Stegmann
Version: 3-15-2000

23.38

double **Normalize** ()

Normalize to unit scale and translates COG to origo.

Normalizes the shape by translating it's center of gravity to origo and scale by the reciprocal of the 2-norm.

Return Value: `The` 2-norm of the shape seen as a 2*nbPoint vector afterthe translation to origo.
Author: Mikkel B. Stegmann
Version: 02-08-2000

23.39

int **SetPoint** (int i, const double &x, const double &y)

Overwrites the i'th point.

Return Value: `Zero.`
Author: Mikkel B. Stegmann
Version: 02-10-2000

23.40

int **SetPoint** (const int i, const CAAMPoint &p)

Overwrites the i'th point. If the point doesn't exists, non-zero is returned.

Return Value: `Zero` on success, non-zero if the point doesn't exists.
Author: Mikkel B. Stegmann
Version: 02-10-2000

23.41

int **GetPoint** (int i, double &x, double &y) const

Returns the i'th point.

Returns the i'th point.

Return Value: `Zero`.
Author: Mikkel B. Stegmann
Version: 02-10-2000

23.42

CAAMPoint **GetPoint** (int i) const

Returns the i'th point.

Returns the i'th point.

Return Value: `The` i'th point.
Author: Mikkel B. Stegmann
Version: 02-10-2000

23.43

void **Resize** (int length, double* storage)

Change number of shape points in the shape.

Change number of shape points in the shape. Note: Destroys *all* current point data and point info data.

Return Value: `Nothing.`
Parameters: `length`
`storage`
See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 4-26-2000

23.44

void **AlignTransformation** (const CAAMShape &ref, double &scale, double &theta, CAAMPoint &t) const

Returns the transformation that aligns this to 'ref' with respect to pose.

Returns the transformation that aligns this to 'ref' with respect to pose.

Return Value: `Nothing.`
Parameters: `ref` The reference shape.
Author: Mikkel B. Stegmann
Version: 03-06-2000

23.45

double **AlignTo** (const CAAMShape &ref, double* pTheta)

Aligns this to 'ref' with respect to pose.

Aligns this to 'ref' with respect to pose.

Return Value: `The` 2-norm of the this shape seen as a 2*nbPoint vector afterthe translation to origo but before the scale done to fit'ref'.
Parameters: `ref` The reference shape.
`pTheta` Optional pointer to return the rotationcarried out on this.

Author: Mikkel B. Stegmann
Version: 03-06-2000

23.46

double **GetRotation** (const CAAMShape &ref) const

Returns the rotation between ref and this (in radians).

Get the rotation between two shapes by minimizing the sum of squared point distances, as described by Goodall (and Bookstein) using Singular Value Decomposition (SVD).

Note that both shapes must be normalized with respect to scale and position beforehand. This could be done by using CAAMSAhape::Normalize().

Return Value: `The` estimated angle, theta, between the two shapes.
Author: Mikkel B. Stegmann
Version: 02-09-2000

23.47

void **FromFile** (FILE* fh)

Reads a shape from a file.

Return Value: `Nothing.`
Parameters: `fh` An open file handle.
Author: Mikkel B. Stegmann
Version: 3-15-2000

23.48

void **ToFile** (FILE* fh) const

Write the shape to a binary file.

Write the shape to a binary file.

Return Value: `Nothing.`
Parameters: `sFilename` Destination filename.

See Also: FromFile
Author: Mikkel B. Stegmann
Version: 5-4-2000

23.49

void **Param2PoseVec** (const double scale, const double theta, const double tx, const double ty, CDVector &poseVec)

Converts pose parameters: scale, theta, tx, ty to a pose vector.

Converts pose parameters: scale, theta, tx, ty to a pose vector. The pose vector will be in the format:

[s, theta, tx, ty]

where s = scale-1

Return Value: `Nothing.`

Parameters:

`scale`	Scale input.
`theta`	Rotational input.
`tx`	X translation input.
`ty`	Y translation input.
`poseVec`	The output pose vector.

See Also: PoseVec2Param, Displace
Author: Mikkel B. Stegmann
Version: 3-15-2000

23.50

void **PoseVec2Param** (const CDVector &poseVec, double &scale, double &theta, double &tx, double &ty)

Converts a pose vector to pose parameters: scale, theta, tx, ty.

Converts a pose vector to pose parameters: scale, theta, tx, ty. The pose vector are expected to be in the format:

[s, theta, tx, ty]

where s = scale-1

Return Value:

Parameters:

`poseVec`	The input pose vector.
`scale`	Scale output.
`theta`	Rotational output.
`tx`	X translation output.
`ty`	Y translation output.

See Also: Param2PoseVec, Displace
Author: Mikkel B. Stegmann
Version: 3-27-2000

23.51

void **Displace** (const CDVector &poseVec)

Displaces the shape around it's center of gravity.

Displaces the shape around it's center of gravity using a displacement vector.

Return Value: `Nothing.`
Parameters: `poseVec` The input pose vector.
See Also: PoseToParam
Author: Mikkel B. Stegmann
Version: 3-15-2000

23.52

bool **IsInsidePath** (const CAAMPoint &p, const int path_start) const

Tests if the point 'p' is inside the path starting at position 'path_start'. Rarely used.

Tests if the point 'p' is inside the path starting at position 'path_start'. Rarely used. Primary a helper function to IsInside.

Return Value: `True` if 'p' is inside
Parameters:
`p` The point to test for.
`path_start` The point index where a path starts
See Also: IsInside
Author: Mikkel B. Stegmann
Version: 5-4-2000

23.53

bool **IsInside** (const CAAMPoint &p, bool bBoundTest) const

Tests if the point 'p' belongs to the shape.

Tests if the point 'p' belongs to the shape. In the current version all shape points are assumes to be ordered points in a n-point polygon with no holes. For example used to clean up the meshes (removing unwanted triangles from the triangulisation).

Return Value: `True` if 'p' is inside.
Parameters:

`'p'`	The test point.
`bBoundTest`	Flags wheather a bounding box test should beperformed prior to the exact test.

Author: Mikkel B. Stegmann
Version: 4-4-2000

23.54

void **Expand** (int nPixels)

Expands the shape (contraction can be done by using a negative nPixels).

Expands the shape by moving each model point 'nPixels' perperdicular to the shape contour (that is: along the model point normal).

This function will expand each outer (closed) path of the shape.

No tests for crossing contours are being made as of now.

Return Value: `Nothing.`
Parameters: `nPixels` The number of pixel to expand the shape with.
Author: Mikkel B. Stegmann
Version: 4-4-2000

23.55

void **Normal** (const int i, CAAMPoint& p1, CAAMPoint& p2, const double dist) const

Finds the normal to the i'th point on the shape.

Finds the normal to the i'th point on the shape. If the point should be a single point the normal points is defined to be equal to the point it self.

Return Value: `Nothing.`
Parameters:

`i`	Index of point.
`p1`	Reference to the outside normal point.
`p2`	Reference to the inside normal point.
`dist`	The desired distance from p[1—2] to the i-th point.

Author: Mikkel B. Stegmann
Version: 4-4-2000

23.56

void **NormalDisplacement** (const int i, const double dist)

Displaces the i-th point along the normal.

Displaces the i-th point along the normal.

Return Value: `Nothing.`
Parameters:

`i`	Index of point.
`dist`	The distance to move the point (>0 move point outwards, <0 inwards).

See Also: Normal
Author: Mikkel B. Stegmann
Version: 12-6-2000

23.57

bool **WriteASF** (const CString &filename, const int image_width, const int image_height)

Writes the shape to a ASF file.

Writes the shape to a ASF file. Remember asf's are always in relative coordinates. Se format description else where.

Return Value: `true` on success, false on errors
Parameters:

`filename`	Output filename.
`image_width`	The image the coord. is relative to.
`image_height`	The image the coord. is relative to.

See Also: ReadASF
Author: Mikkel B. Stegmann
Version: 4-25-2000

23.58

bool **WriteASF0_90** (const CString &filename, const int image_width, const int image_height)

ASF writer version 0.90.

Writes the shape to a ver. 0.90 ASF file. Remember asf's are always in relative coordinates. Se format description else where.

Return Value: `true` on success, false on errors

Parameters:

`filename`	Output filename.
`image_width`	The image the coord. is relative to.
`image_height`	The image the coord. is relative to.

See Also: ReadASF

Author: Mikkel B. Stegmann

Version: 7-2-2001

23.59

bool **ReadASF** (const CString &filename)

Reads an ver. 0.90 .asf into relative coordinates.

Reads an ver. 0.90 .asf into relative coordinates. Se format description else where.

Return Value: `true` on success, false on errors

Parameters: `filename` Input filename.

See Also: WriteASF

Author: Mikkel B. Stegmann

Version: 4-25-2000

23.60

bool **ReadASF0_90** (const CString &filename)

Reads an ver. 0.90 .asf into relative coordinates.

Reads an ver. 0.90 .asf into relative coordinates. Se format description else where.

Return Value: `true` on success, false on errors

Parameters: `filename` Input filename.

See Also: WriteASF
Author: Mikkel B. Stegmann
Version: 4-25-2000

23.61

void **Abs2Rel** (const CString hostImagePath)

Converts shape coordinates from absolute to relative by using the hostimage.

Converts shape coordinates from absolute to relative by using the hostimage. **Author:** Mikkel B. Stegmann
Version: 11-15-2000

23.62

void **Rel2Abs** (const CString hostImagePath)

Converts shape coordinates from relative to absolute by using the hostimage.

Converts shape coordinates from relative to absolute by using the hostimage.

Author: Mikkel B. Stegmann
Version: 11-15-2000

23.63

void **Abs2Rel** (const int image_width, const int image_height)

Converts shape coordinates from absolute to relative.

Converts shape coordinates from absolute to relative. Relative coordinates are specified as:

x_relative = x_abs/image_width

y_relative = y_abs/image_height

Return Value: `Nothing.`
Parameters:

`image_width`	The image the coord. should be relative to.
`image_height`	The image the coord. should be relative to.

See Also: Rel2Abs
Author: Mikkel B. Stegmann
Version: 4-25-2000

23.64

void **Rel2Abs** (const int image_width, const int image_height)

Converts shape coordinates from relative to absolute.

Converts shape coordinates from relative to absolute. Relative coordinates are specified as:

x_relative = x_abs/image_width

y_relative = y_abs/image_height

Return Value: `Nothing.`

Parameters:

	`image_width`	The image the coord. is relative to.
	`image_height`	The image the coord. is relative to.

See Also: Abs2Rel

Author: Mikkel B. Stegmann

Version: 4-25-2000

23.65

void **AddPath** (const CAAMShape &shape, const CAAMPointInfo &pointType)

Adds a path to the shape.

Adds a path to the shape. All added point inherits the given pointtype.

Return Value: `Nothing.`

Parameters:

	`shape`	A shape containing one path.
	`pointType`	The pointtype of the added points.

Author: Mikkel B. Stegmann

Version: 4-26-2000

23.66

void **ReversePointOrder** ()

Reverses the point point ordering.

Reverses the point point ordering. Used when a clock-wise outer path shall converted to a counter-clock wise hole.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 5-2-2000

23.67

std::vector<int> **GetPaths** () const

Extracts the starting positions of each path in the shape.

Extracts the starting positions of each path in the shape by a simple linear search. The starting position is identified by a change in path id (saved in the m_vPointAux member).

Return Value: `A` vector of path starting positions.
Author: Mikkel B. Stegmann
Version: 5-3-2000

23.68

int **PathLen** (const int startPosition) const

Returns the length of a path.

Returns the length of a path.

Return Value: `The` path length.
Parameters: `startPosition` The starting position of the path.
See Also: GetPaths
Author: Mikkel B. Stegmann
Version: 5-5-2000

23.69

CAAMShape **ExtractPath** (const int startPosition) const

Extracts one path from a shape.

Extracts one path from a shape into a new shape.

Return Value: `The` path as a new shape.
Parameters: `startPosition` The starting position of the path.

See Also: PathLen, GetPaths
Author: Mikkel B. Stegmann
Version: 5-5-2000

23.70

void **AddShapeExtends** (int nPixels)

Adds an extra border on each outer path.

Adds an extra outer path on each outer path in the distance of 'nPixels' along the point normal.

This method is primary used in conjunction with the "do not use the convex hull" feature. In such a case one often still wants a certain neighborhood of the shape to be included in the model.

Return Value: `Nothing.`
Parameters: `nPixels` The size of the extents.
Author: Mikkel B. Stegmann
Version: 5-5-2000

23.71

void **SetPointInfoFlagsInPath** (const int startPosition, const int flags)

Sets all flags in one path to the same value.

Sets all flags in one path to the same value.

Return Value: `Nothing.`
Parameters: `startPosition` Starting position of the path.
`flags` The flags.
Author: Mikkel B. Stegmann
Version: 5-9-2000

23.72

void **AddInterior** (const int interations)

Adds artificial interior points to the shape.

Add articifial interior points to the shape by making a Delaunay triangulation and adding the centroid of each triangle. This is done iteratively. Default is one iteration

Return Value: `Nothing.`
Parameters: `interations` Controls the number of artificial points. One iteration equals one triangulation.
Author: Mikkel B. Stegmann
Version: 12-12-2000

23.73

bool **ConsistencyCheck** ()

Tests if any interior points has gone outside outer path of the shape.

Tests if any interior points has gone outside outer path of the shape.

Return Value: `True` if the shape looks ok, false if not.
Author: Mikkel B. Stegmann
Version: 12-12-2000

23.74

void **MakeBorderShape** (int size)

Converts a one path shape into a border shape.

Converts a one path shape into a border shape by adding two symmetric borders: an inside (a hole) and an outside border.

No checks for folding paths are done as of now.

Return Value: `Nothing.`
Parameters: `size` The size of the border in pixels.
Author: Mikkel B. Stegmann
Version: 7-27-2000

23.75

CAAMShape operator+ (const CVisDVector &v) const

Plus operator.

Plus operator.

Return Value:	`The` addition of this and 'v'.
Parameters:	`v` Vector to add.
Author:	Mikkel B. Stegmann
Version:	10-20-2000

23.76

CAAMShape operator- (const CVisDVector &v) const

Minus operator.

Minus operator.

Return Value:	`The` subtraction of this and 'v'.
Parameters:	`v` Vector to add.
Author:	Mikkel B. Stegmann
Version:	10-20-2000

23.77

void GetHostImage (CDMultiBand<TAAMPixel> &dest, const CString &path, const int rfactor) const

Retrives the image connected to the shape.

Retrives the image connected to the shape. As of now it's loaded from disk using the HostImage() member in the shape.

Return Value:	`Nothing.@throws`	CVisFileIOError
Parameters:	`dest`	The destination image.
	`path`	The path to the .asf file.
	`rfactor`	Optional reduction factor. Performs a scaling of thethe shape by 1/rfactor. Default 1 i.e. no scaling.

Author: Mikkel B. Stegmann
Version: 10-24-2000

23.78

double **Area** (bool use_covex_hull) const

Returns the total area of the shape (with holes excluded).

Returns the total area of the shape (with holes excluded).

Return Value: `The` area.
Parameters: `use_covex_hull` Use the convex hull of the shape forarea calculation (default false).
Author: Mikkel B. Stegmann
Version: 11-15-2000

23.79

CAAMShape **CalcConvexHull** () const

Calculates the convex hull of each path in the shape.

Calculates the convex hull of each path in the shape. The calulation is built upon the geometrical fact that the homogenous point matrix: [p1x p2x p3x ; p1y p2y p3y ; 1 1 1] is positive if p1, p2, p3 is a convex segment and negative if concave. Note: this holds for a clock-wise ordering of p1,p2,p3.

Any open paths are considered cyclic in the concavity calulation.

Remember that paths should be defined clock-wise in the asf format.

BUG COMMENT: This does not seem to work with multiple paths.

Return Value: `A` convex version of this shape w.r.t. each path.
See Also: IsConvex
Author: Mikkel B. Stegmann
Version: 5-2-2001

23.80

bool **IsConvex** () const

Tests if the shap is convex w.r.t. each path.

Tests if the shap is convex w.r.t. each path. This call is rather expensive, since it spaws a call to CalcConvexHull().

Return Value:	`True` on convex, false on concave.
See Also:	CalcConvexHull
Author:	Mikkel B. Stegmann
Version:	5-2-2001

23.81

void **RemovePoint** (const int i)

Removes the i-th point from the shape.

Removes the i-th point from the shape. Since this require massive reordering of the preceeding point connectivity this is actually a very expensive call [a fairly ugly in it's implementation] :-(

Return Value:	`Nothing.`
Parameters:	`i` The index of the point to remove.
Author:	Mikkel B. Stegmann
Version:	5-2-2001

23.82

std::vector<float> & **UserField** (const int field_nb)

Returns a reference to a user defined field vector.

Returns the reference of the user defined field vector number 'field_nb'.

Return Value:	`A` reference to the specified field vector.
Parameters:	`field_nb` Field number [1-3].
See Also:	(→, *page 239*)
Author:	Mikkel B. Stegmann
Version:	7-2-2001

23.83

const std::vector<float> & **UserField** (const int field_nb) const

Returns the value of a user defined field vector.

Returns the value of the user defined field vector number 'field_nb'.

Return Value: `A` float holding the value of the field.
Parameters: `field_nb` Field number [1-3].
See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 7-2-2001

23.84

void **AllocateUserFields** ()

Allocates room for the three user defined fields.

Allocates room for the three user defined fields.

Notice that the user defined fields is *not* allocated by default.

In order to preserved memory the user must to this explicitly by using AllocateUserFields();

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 7-2-2001

23.2

std::vector<CAAMPointInfo> **m_vPointAux**

Auxillary point data

Auxillary point data

23.3

std::vector<float> **m_vUser1**

User-defined field 1

User-defined field 1

23.4

std::vector<float> **m_vUser2**

User-defined field 2

User-defined field 2

23.5

std::vector<float> **m_vUser3**

User-defined field 3

User-defined field 3

23.6

CString **m_szASFVer**

Current ASF version number

Current ASF version number

23.7

int **m_iNbPoints**

The number of points

The number of points

23.8

bool **m_bAbsPointCoordinates**

Indicates if the point coordinates is in relative or absolute format

Indicates if the point coordinates is in relative or absolute format

23.9

CString **m_szHostImage**

Optional 'host image' filename including full path.

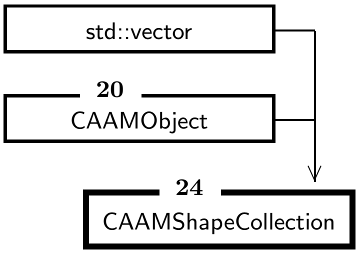
Optional 'host image' filename including full path.

24

class **CAAMShapeCollection** : public std::vector<CAAMShape>, CAAMObject

Shape collection container and shape-aligner.

Inheritance



Public Members

24.4	inline int	**NShapes** () const	*The number of shapes in the collection*	176
24.5	inline int	**NPoints** () const	*The number of points in each of shape*	177
24.6	double	**MeanSize** () const	*Returns the average shape size *before* the aligment process*	177
24.7	const CString&	**Path** () const	*Returns the path of all shapes*	177
24.8		**Assignment operator**		177
24.9		**CAAMShapeCollection** ()	*Constructor.*	177
24.10		**~CAAMShapeCollection** ()	*Destructor.*	178
24.11	void	**Insert** (const CAAMShape &s)	*Inserts a shape into the collection.*	178
24.12	int	**ToMatlab** (const CString& sFilename, const CString& sName, const CString& sComment, bool fAppend) const		178
24.13	int	**AlignShapes** (bool use_tangentspace)		179
24.14	void	**MeanShape** (CAAMShape &meanShape) const	*Calcs the mean shape of all shapes.*	179
24.15	void	**ReferenceShape** (CAAMShape &refShape) const	*Calcs the mean shape of all aligned shapes and size it to mean size.*	179
24.16	double	**MinX** () const	*Find the minimum x component of all shapes.*	180
24.17	double	**MaxX** () const	*Find the maximum x component of all shapes.*	180
24.18	double	**MinY** () const	*Find the minimum x component of all shapes.*	180

Private Members

This class act as a container for a set of shapes. Secondary it can align the set of shapes to a normalised reference frame with respect to position, scale and orientation.

See Also: CAAMShape
Author: Mikkel B. Stegmann
Version: 02-08-2000

24.4

inline int NShapes () const

The number of shapes in the collection

The number of shapes in the collection

24.5

inline int **NPoints** () const

The number of points in each of shape

The number of points in each of shape

24.6

double **MeanSize** () const

*Returns the average shape size *before* the aligment process*

Returns the average shape size *before* the aligment process

24.7

const CString& **Path** () const

Returns the path of all shapes

Returns the path of all shapes

24.8

Assignment operator

Assignment operator

24.9

CAAMShapeCollection ()

Constructor.

Constructor.

Author: Mikkel B. Stegmann
Version: 7-27-2000

24.10

~CAAMShapeCollection ()

Destructor.

Destructor.

Author: Mikkel B. Stegmann
Version: 7-27-2000

24.11

void **Insert** (const CAAMShape &s)

Inserts a shape into the collection.

Parameters: `s` The input shape.
Author: Mikkel B. Stegmann
Version: 7-27-2000

24.12

int **ToMatlab** (const CString& sFilename, const CString& sName, const CString& sComment, bool fAppend) const

Writes the shapes in a (NShapes x 2*NPoints) matrix in Matlab (*.m) format. The i-th row thus contains the i-th shape in xxxyy format.

Return Value: `Zero` on sucess, non-zero if no shapes are stored in the collection.

Parameters:
- `sFilename` Filename including path to be written.
- `sName` The matlab variable name of the matrix.
- `sComment` An optional comment.
- `fAppend` Addend to an existing file or overwrite.

Author: Mikkel B. Stegmann
Version: 02-09-2000

24.13

int AlignShapes (bool use_tangentspace)

Normalizes all shapes with respect to position, scale and orientation. Position normalization are done by a translation of the center of gravity to origo. Scale normalization are done by a scaling of 1/<norm2>. Rotation normalization are done by minimizing the sum of squared point distances, as described by Goodall using Singular Value Decomposition (SVD).

Return Value: `Zero` on succes.
Author: Mikkel B. Stegmann
Version: 02-09-2000

24.14

void MeanShape (CAAMShape &meanShape) const

Calcs the mean shape of all shapes.

Calcs the mean shape of all shapes.

Parameters: `meanShape` The output mean shape.
See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 7-27-2000

24.15

void ReferenceShape (CAAMShape &refShape) const

Calcs the mean shape of all aligned shapes and size it to mean size.

Calcs the mean shape of all aligned shapes and size it to mean size.

Parameters: `refShape` The output reference shape.
See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 7-27-2000

24.16

double **MinX** () const

Find the minimum x component of all shapes.

Return Value: The x-minimum.
Author: Mikkel B. Stegmann
Version: 5-15-2000

24.17

double **MaxX** () const

Find the maximum x component of all shapes.

Return Value: The x-maximum.
Author: Mikkel B. Stegmann
Version: 5-15-2000

24.18

double **MinY** () const

Find the minimum x component of all shapes.

Return Value: The x-minimum.
Author: Mikkel B. Stegmann
Version: 5-15-2000

24.19

double **MaxY** () const

Find the maximum y component of all shapes.

Return Value:	`The y-maximum.`	
Author:	Mikkel B. Stegmann	
Version:	5-15-2000	

24.20

void **Scale** (const double s, const bool aroundCOG)

Scale all shapes.

Scale the shapes.

Return Value:	`Nothing.`	
Parameters:	`s`	Scale factor.
	`aroundCOG`	If true the scale is being done around the cog of the shape instead of around the globalcenter.
Author:	Mikkel B. Stegmann	
Version:	3-15-2000	

24.21

void **Expand** (int nPixels)

Expands all shapes (contraction can be done by using a negative nPixels).

Expands all shapes by moving each model point 'nPixels' perperdicular to the shape contour (that is: along the model point normal).

This function will expand each outer (closed) path of the shape.

No tests for crossing contours are being made as of now.

Return Value:	`Nothing.`	
Parameters:	`nPixels`	The number of pixel to expand the shape with.
Author:	Mikkel B. Stegmann	
Version:	4-4-2000	

24.22

void **ToFile** (const CString& sFilename) const

Write the set of shapes to a binary file.

Write the set of shapes to a binary file.

Return Value: `Nothing.`
Parameters: `sFilename` Destination filename.
See Also: FromFile
Author: Mikkel B. Stegmann
Version: 5-4-2000

24.23

void **FromFile** (const CString& sFilename)

Reads a set of shapes from a file.

Return Value: `Nothing.`
Parameters: `sFilename` The filename.
Author: Mikkel B. Stegmann
Version: 3-15-2000

24.24

void **ToFile** (FILE* fh) const

Write the set of shapes to a binary file.

Write the set of shapes to a binary file.

Return Value: `Nothing.`
Parameters: `fh` Destination file handle.
See Also: FromFile
Author: Mikkel B. Stegmann
Version: 5-4-2000

24.25

void **FromFile** (FILE* fh)

Reads a set of shapes from a file.

Return Value: `Nothing.`
Parameters: `fh` An open file handle.
Author: Mikkel B. Stegmann
Version: 3-15-2000

24.26

bool **ReadShapes** (const std::vector<CString> &asfFiles, bool validate)

Reads a set of shapes.

Reads a set of shapes in the order given in the vector of strings.

Return Value: `True` on a valid training set - otherwise false.
Parameters: `asfFiles` Vector of asf filenames.
`validate` Validates that all shapes have thesame number of points.
Author: Mikkel B. Stegmann
Version: 10-24-2000

24.27

bool **ReadDir** (const CString &path, bool validate)

Scans a dir for annotations (.asf) and read shapes.

Scans a dir for annotations (.asf) and read shapes in alphabetical order.

Return Value: `True` on a valid training set - otherwise false.
Parameters: `path` Full path to annotations.
`validate` Validates that all shapes have thesame number of points.
Author: Mikkel B. Stegmann
Version: 10-24-2000

24.28

void **Rel2Abs** (int rfactor)

Converts all shapes with relative coordinates to absolute.

Converts all shapes with relative coordinates to absolute. Unfortunately this requires to read the headers of all host images. VisSDK does not provide any operation for this. Thus, all images are one by one read into memory and discarded again to obtain height and width. Very costly :-(

Return Value: `Nothing.`
Parameters: `rfactor` Optional reduction factor. Performs a scaling of thethe shape by 1/rfactor. Default 1 i.e. no scaling.
Author: Mikkel B. Stegmann
Version: 10-25-2000

24.1

CAAMShape **m_MeanShapeBeforeTS**

The meanshape prior to tangent space projection (calculated during alignment)

The meanshape prior to tangent space projection (calculated during alignment)

24.2

double **m_dAvgSize**

The average shape size (calculated during alignment)

The average shape size (calculated during alignment)

24.3

CString **m_szPath**

Path to the asf files (if any)

Path to the asf files (if any)

25 class CAAMEvaluationResults

Container to store evaluation results in.

Public Members

Container to store evaluation results in. Further, it can print out the results along with some simple statistics.

Author: Mikkel B. Stegmann
Version: 12-2-2002

25.1 void **AddResult** (const CAAMShape &model_shape, const CAAMShape &ground_truth, const double &time, const CAAMModel::CAAMOptRes &optRes)

Adds a new result to the back

Adds a new result to the back

25.2 void **AddResults** (const CAAMEvaluationResults &results)

Adds a new set of results to the back

Adds a new set of results to the back

25.3

void **PrintStatistics** (FILE* fh = stdout)

Prints all results and some statistics to either file or screen (default)

Prints all results and some statistics to either file or screen (default)

25.4

int **PrintStatistics** (const CString &filename)

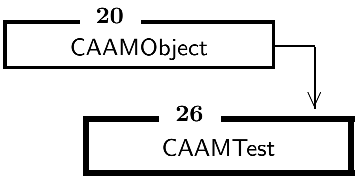
Prints all results to a file

Prints all results to a file

26 class **CAAMTest** : public CAAMObject

Container for all sorts of test functions.

Inheritance



Public Members

Container for all sorts of test functions. In this way a history of all mock-up test functions developed during debugging and testing are kept.

All functions are implemented as static functions. Thus no instantiation the CAAMTest are needed.

Author: Mikkel B. Stegmann
Version: 03-05-2001

26.1 CAAMTest ()

Constructor.

Constructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 3-5-2001

26.2 ˜CAAMTest ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 3-5-2001

26.3 void **GetRotationTest** (const CAAMShape &s1, const CAAMShape &s2)

Tests CAAMShape:GetRotation() for rotations in the range [0;360].

Tests CAAMShape:GetRotation() for rotations in the range [0;360].

Return Value: `Nothing.`
Parameters:
`s1` First shape.
`s2` Second shape.
Author: Mikkel B. Stegmann
Version: 3-5-2001

26.4

void **AnalyzeTest** (const CAAMShape &refShape, const CAAMShape &s, const CDMultiBand<TAAMPixel> &image, const bool useConvexHull)

Benchmarks the software warping method against the OpenGL ditto.

Benchmarks the software warping method against the OpenGL ditto.

Return Value:	`Nothing.`	
Parameters:	`refShape`	Reference shape, i.e. mean shape scale to mean size.
	`s`	Input shape in relative or absolute coordinates.
	`image`	Host image of 's'.
	`useConvexHull`	If true the convex hull is used.
Author:	Mikkel B. Stegmann	
Version:	6-13-2002	

26.5

void **TestPosePrediction** (const CAAMModel &model, const CString &path)

Tests the prediction matrices ability to predict pose displacements.

Output are returned in the form of eight matlab formatted files in the current directory.

Return Value:	`Nothing.`	
Parameters:	`model`	The AAM to test.
	`path`	The path where test images and annotations are placed(including terminating backslash).
Author:	Mikkel B. Stegmann	
Version:	3-27-2000	

26.6

CAAMEvaluationResults **EvaluateModel** (const CAAMModel* pModel, const CString &gt_path, const CString &result_file, const bool writeStills, const bool writeMovies, const bool autoinit, const bool dump2screen, CAAMLowerBounds* pLB)

Optimizes a set of images and compares the result to a ground truth annotation.

Optimizes a set of images and compares the result to a ground truth annotation. As initialization the ground truth pose is systematically displaced (default) or an automatic initialisation is performed.

Return Value: `Evaluation` results.

Parameters:

`model`	The model to evaluate.
`gt_path`	Path to ground truth images and annotations.
`result_file`	The file to write the results in.
`writeStills`	If true two model border images are written; one of the initialisation and one of the optimization.
`writeMovies`	It true a movie of the whole optimization is written, one frame per iteration.
`autoinit`	If true automatic initialization is performed insteadof the systematic displacement of the ground truthpose.
`dump2screen`	If true, results are written to the screen also (default true).
`pLB`	Optional pointer to a CAAMLowerBounds object.

Author: Mikkel B. Stegmann

Version: 4-5-2000

26.7

CAAMEvaluationResults **EvaluateModelSeq** (const CAAMModelSeq &modelSeq, const CString &gt_path, const CString &result_file, const bool writeStills, const bool writeMovies, const bool autoinit, const bool dump2screen, CAAMLowerBounds* pLB)

Optimizes a set of images and compares the result to a ground truth annotation using a sequence of AAMs.

Optimizes a set of images and compares the result to a ground truth annotation using a sequence of AAMs. As initialization the ground truth pose is systematically displaced (default) or an automatic initialisation is performed.

Return Value: `Evaluation` results.

Parameters:

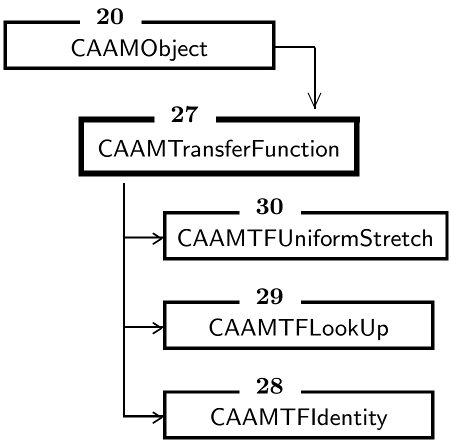
`modelSew`	A sequence of models.
`gt_path`	Path to ground truth images and annotations.
`result_file`	The file to write the results in.
`writeStills`	If true two model border images are written; one of the initialisation and one of the optimization.
`writeMovies`	It true a movie of the whole optimization is written, one frame per iteration.
`autoinit`	If true automatic initialization is performed insteadof the systematic displacement of the ground truthpose.
`dump2screen`	If true, results are written to the screen also (default true).
`pLBS`	Optional pointer to a CAAMLBShapeModel object.

Author: Mikkel B. Stegmann
Version: 4-5-2000

27 class **CAAMTransferFunction** : public CAAMObject

Abstract base class for all transfer functions.

Inheritance



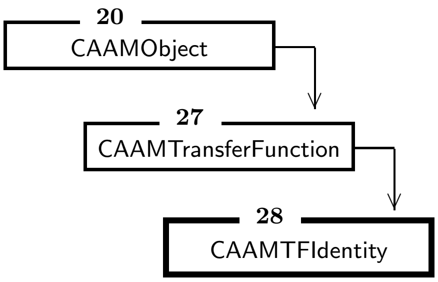
This class defines an interface for the so-called transfer functions, which are nothing but differnt mapping from one vector space to another.

Author:	Mikkel B. Stegmann
Version:	1-25-2002

28 class **CAAMTFIdentity** : public CAAMTransferFunction

Transfer function that implements the identity transformation.

Inheritance



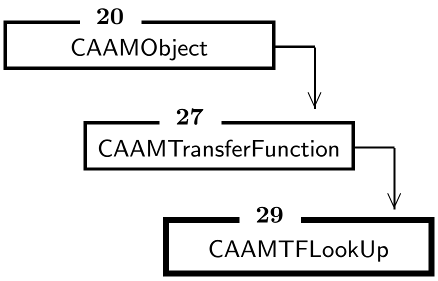
Transfer function that implements the identity transformation. Actually this function is one of the reasons for the design of the Map and DeMap methods. An identity transform should not induce any computational overhead whatsoever.

Author: Mikkel B. Stegmann
Version: 1-25-2002

29 class **CAAMTFLookUp** : public CAAMTransferFunction

Transfer function that implements a lookup table.

Inheritance



Public Members

29.1		**CAAMTFLookUp** ()	*Constructor*	195
29.2		~**CAAMTFLookUp** ()	*Destructor*	195
29.3	CAAMTransferFunction*	**Clone** () const	*Clone function (conveys type info)*	195
29.4	const char*	**TypeName** () const	*Returns the clear-text type name*	195
29.5	void	**LoadLUT** (const CDVector &lut)	*Loads the lookup table and generates the inverse.*	195
29.6	void	**Map** (CDVector &v) const	*Maps an input vector using the current lookup table.*	196
29.7	void	**DeMap** (CDVector &v) const	*The inverse of the lookup table transform.*	196
29.8	void	**FromFile** (FILE* fh)	*Reads the class info from file.*	196
29.9	void	**ToFile** (FILE* fh) const	*Writes the class info to file.*	197

Transfer function that implements a lookup table, which is useful for e.g. histogram equalisations.

Author: Mikkel B. Stegmann
Version: 1-28-2002

29.1 CAAMTFLookUp ()

Constructor

Constructor

29.2

~CAAMTFLookUp ()

Destructor

Destructor

29.3

CAAMTransferFunction* **Clone** () const

Clone function (conveys type info)

Clone function (conveys type info)

29.4

const char* **TypeName** () const

Returns the clear-text type name

Returns the clear-text type name

29.5

void **LoadLUT** (const CDVector &lut)

Loads the lookup table and generates the inverse.

Loads the lookup table and generates the inverse, i.e. assumes that the LUT monotonic.

Return Value: `Nothing.`
Parameters: `lut` A lookup table in vector form.
Author: Mikkel B. Stegmann
Version: 1-28-2002

29.6

void **Map** (CDVector &v) const

Maps an input vector using the current lookup table.

Maps an input vector using the current lookup table.

Return Value: `Nothing.` The result is returned in v.
Parameters: `v` Input vector.
Author: Mikkel B. Stegmann
Version: 1-28-2002

29.7

void **DeMap** (CDVector &v) const

The inverse of the lookup table transform.

The inverse of the lookup table transform. The input vector is transformed into [0;255] and the the inverse LUT is applied.

Return Value: `Nothing.` The result is returned in v.
Parameters: `v` Input vector to de-map.
See Also: Map
Author: Mikkel B. Stegmann
Version: 1-28-2002

29.8

void **FromFile** (FILE* fh)

Reads the class info from file.

Reads the class info from file.

Return Value: `Nothing.`
Parameters: `fh` Open binary file handle.
See Also: ToFile
Author: Mikkel B. Stegmann
Version: 1-28-2002

29.9

void **ToFile** (FILE* fh) const

Writes the class info to file.

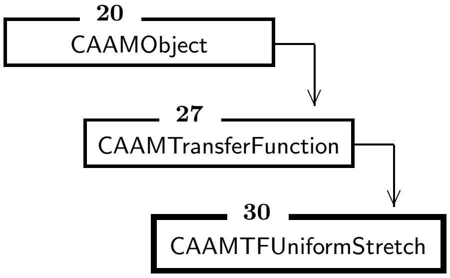
Writes the class info to file.

Return Value:	`Nothing.`
Parameters:	`fh` Open binary file handle.
See Also:	FromFile
Author:	Mikkel B. Stegmann
Version:	1-28-2002

30 class **CAAMTFUniformStretch** : public CAAMTransferFunction

Performs a uniform stretch into [0;255] on the demapping side.

Inheritance



Public Members

30.1		**CAAMTFUniformStretch** ()	*Constructor*	198
30.2		**~CAAMTFUniformStretch** ()	*Destructor*	199
30.3	CAAMTransferFunction*	**Clone** () const	*Clone function (conveys type info)*	199
30.4	const char*	**TypeName** () const	*Returns the clear-text type name*	199
30.5	void	**Map** (CDVector &v) const	*Does nothing.*	199
30.6	void	**DeMap** (CDVector &v) const	*Maps the vector into [0;255].*	200

Performs a uniform stretch into [0;255] on the demapping side (yes, a really degenerate class... I know...).

Author: Mikkel B. Stegmann
Version: 1-25-2002

30.1 CAAMTFUniformStretch ()

Constructor

Constructor

30.2

~CAAMTFUniformStretch ()

Destructor

Destructor

30.3

CAAMTransferFunction* **Clone** () const

Clone function (conveys type info)

Clone function (conveys type info)

30.4

const char* **TypeName** () const

Returns the clear-text type name

Returns the clear-text type name

30.5

void **Map** (CDVector &v) const

Does nothing.

Does nothing.

Return Value: `Nothing.`
Parameters: `v` Input vector.
Author: Mikkel B. Stegmann
Version: 1-28-2002

30.6

void **DeMap** (CDVector &v) const

Maps the vector into [0;255].

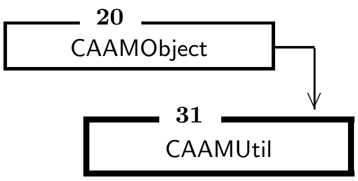
Maps the vector into [0;255].

Return Value:	`Nothing.` The result is returned in v.
Parameters:	`v` Input vector.
Author:	Mikkel B. Stegmann
Version:	1-28-2002

31 class **CAAMUtil** : public CAAMObject

Utility methods for the AAM project.

Inheritance



Public Members

31.1 static double **Deg2Rad** (double deg) *Converts from degrees to radians* 203

31.2 static double **Rad2Deg** (double rad) *Convert from radians to degrees* 204

31.3 void **PlotMeshIntoImage** (CDMultiBand<TAAMPixel> image, const CAAMMesh &mesh, TAAMPixel* point_color, TAAMPixel* line_color)
Plots a mesh into an image. 204

31.4 void **PlotShapeIntoImage** (CDMultiBand<TAAMPixel> image, const CAAMShape &shape, TAAMPixel* point_color, TAAMPixel* line_color, bool drawNormals, bool drawArea, bool drawPoints, bool drawLines)
Draws the shape into an image with a given color. 204

31.5 void **VecCat3** (CDVector &dest, const CDVector &v1, const CDVector &v2, const CDVector &v3)
Concatenates three vectors 205

31.6 std::vector<CString> **ScanSortDir** (const CString &path, const CString &extension)
Scans and sorts a directory for files. 205

31.7 void **ReadExample** (const CString &filename, CDMultiBand<TAAMPixel> &img, CAAMShape &shape, int rfactor)
Reads an image and a shape. 206

31.8 double **DistEuclidianPoints** (const CAAMShape &s1, const CAAMShape &s2)
Calculates the point to point error. 206

31.9 double **DistEuclidianAssBorder** (const CAAMShape &s1, const CAAMShape &s2, CDVector* pvDist)
Calculates the point to curve error. 207

31.10 double **DistEuclidianPoints** (const CDVector& vX1, const CDVector& vY1, const CDVector& vX2, const CDVector& vY2)
Calculates the point to point error. 207

31.11 double **DistEuclidianAssBorder** (const CDVector& vX1, const CDVector& vY1, const CDVector& vX2, const CDVector& vY2, CDVector* pvDist)

This class consists of methods that are self-containing and could not logically fit into any other AAM class.

Thus all methods are static, so remember that there is never need for an instantiation of this class.

Author: Mikkel B. Stegmann
Version: 4-6-2000

31.1

static double **Deg2Rad** (double deg)

Converts from degrees to radians

Converts from degrees to radians

31.2

static double **Rad2Deg** (double rad)

Convert from radians to degrees

Convert from radians to degrees

31.3

void **PlotMeshIntoImage** (CDMultiBand<TAAMPixel> image, const CAAMMesh &mesh, TAAMPixel* point_color, TAAMPixel* line_color)

Plots a mesh into an image.

Plots a mesh into an image.

Return Value: `Nothing.`

Parameters:

`image`	The image in which the shape should be plotted.
`shape`	The shape to be plotted.
`point_color`	The point color of the mesh.
`line_color`	The line color of the mesh.

See Also: PlotShapeIntoImage
Author: Mikkel B. Stegmann
Version: 5-3-2000

31.4

void **PlotShapeIntoImage** (CDMultiBand<TAAMPixel> image, const CAAMShape &shape, TAAMPixel* point_color, TAAMPixel* line_color, bool drawNormals, bool drawArea, bool drawPoints, bool drawLines)

Draws the shape into an image with a given color.

Draws the shape into an image with a given color.

Return Value:	`Nothing.`	
Parameters:	`image`	The image in which the shape should be plotted.
	`shape`	The shape to be plotted.
	`point_color`	The point color of the shape.
	`line_color`	The line color of the shape.
	`drawNormals`	If true the point normals are drawn.
	`drawArea`	If true the inside area of the shape is drawn.
	`drawPoints`	If true the points of the shape is drawn.
	`drawLines`	If true the lines of the shape is drawn.
See Also:	PlotMeshIntoImage	
Author:	Mikkel B. Stegmann	
Version:	3-27-2000	

31.5

void **VecCat3** (CDVector &dest, const CDVector &v1, const CDVector &v2, const CDVector &v3)

Concatenates three vectors

Concatenates three vectors.

Return Value:	`Nothing.`	
Parameters:	`dest`	Output vector.
	`v1`	First input vector.
	`v2`	Second input vector.
	`v3`	Third input vector.
Author:	Mikkel B. Stegmann	
Version:	3-14-2000	

31.6

std::vector<CString> **ScanSortDir** (const CString &path, const CString &extension)

Scans and sorts a directory for files.

Scans and sorts a directory for files with a specfied extension.

Return Value:	`The` filenames found without any path.	
Parameters:	`path`	Path to read from.
	`extension`	The file extension to search for. ex. "hips".
Author:	Mikkel B. Stegmann	
Version:	3-27-2000	

31.7

void **ReadExample** (const CString &filename, CDMultiBand<TAAMPixel> &img, CAAMShape &shape, int rfactor)

Reads an image and a shape.

Reads an image and a shape (in the corrosponding .m-file).

Return Value: `Nothing` (but throws CVisError)@throws CVisError
Parameters:
- `filename` The filename of an annotation. Ex. "horse.asf".
- `img` Output image.
- `shape` Output shape.
- `rfactor` The reduction factor. Performs a scaling of theinput of 1/rfactor.

See Also: (→, *page 239*)
Author: Mikkel B. Stegmann
Version: 3-10-2000

31.8

double **DistEuclidianPoints** (const CAAMShape &s1, const CAAMShape &s2)

Calculates the point to point error.

Calculates the point to point error between two shapes.

Return Value: `The` average point to point error.
Parameters:
- `s1` Shape 1.
- `s2` Shape 2.

See Also: DistEuclidianAssBorder
Author: Rune Fisker
Version: 4-6-2000

31.9

double **DistEuclidianAssBorder** (const CAAMShape &s1, const CAAMShape &s2, CDVector* pvDist)

Calculates the point to curve error.

Calculates the point to curve error between two shapes.

Return Value: `The` average point to curve error.

Parameters:

- `s1` Shape 1.
- `s2` Shape 2.
- `pvDist` Optional pointer to an output vector containing all point to curve distances (one for each landmark).

See Also: DistEuclidianPoints

Author: Rune Fisker

Version: 4-6-2000

31.10

double **DistEuclidianPoints** (const CDVector& vX1, const CDVector& vY1, const CDVector& vX2, const CDVector& vY2)

Calculates the point to point error.

Calculates the point to point error between two shapes.

Return Value: `The` average point to point error.

Parameters:

- `vX1` X-positions of shape 1.
- `vY1` Y-positions of shape 1.
- `vX2` X-positions of shape 2.
- `vY2` Y-positions of shape 2.

See Also: DistEuclidianAssBorder

Author: Rune Fisker

Version: 4-6-2000

31.11

double **DistEuclidianAssBorder** (const CDVector& vX1, const CDVector& vY1, const CDVector& vX2, const CDVector& vY2, CDVector* pvDist)

Calculates the point to curve error.

Calculates the point to curve error between two shapes.

Return Value:	`The` average point to curve error.	
Parameters:	`vX1`	X-positions of shape 1.
	`vY1`	Y-positions of shape 1.
	`vX2`	X-positions of shape 2.
	`vY2`	Y-positions of shape 2.
	`pvDist`	Optional pointer to an output vector containing all point to curve distances (one for each landmark).
See Also:	DistEuclidianPoints	
Author:	Rune Fisker	
Version:	4-6-2000	

31.12

void **PointOnAssBorder** (const CDVector& vX1, const CDVector& vY1, const CDVector& vX2, const CDVector& vY2, CDVector& vXBorder, CDVector& vYBorder, CDVector& vDist)

Calculates the point to curve error.

Calculates the point to curve error between two shapes.

Return Value:	`Nothing.`	
Parameters:	`vX1`	X-positions of shape 1.
	`vY1`	Y-positions of shape 1.
	`vX2`	X-positions of shape 2.
	`vY2`	Y-positions of shape 2.
	`vXBorder`	Output x border.
	`vYBorder`	Output y border.
	`vDist`	Point to curve distance for each landmark.
See Also:	DistEuclidianPoints	
Author:	Rune Fisker	
Version:	4-6-2000	

31.13

double **ProjPointOnLine** (const double dXL1, const double dYL1, const double dXL2, const double dYL2, const double dXP, const double dYP, double& dXProj, double& dYProj)

Find the projection dProj of the point dP on the line through dL1 and dL2.

Find the projection dProj of the point dP on the line through dL1 and dL2. Returns the distance between dProj and dP, if the dProj lays between dL1 and dL2 otherwise DBL_MAX.

dXL1,dYL1,dXL2,dYL2: points on line

dXP,dYP: points to project

dXProj,dYProj: projected point on the line

Return Value: `The` distance.
Author: Rune Fisker
Version: 11-22-1999

31.14

CString **RemoveExt** (const CString &s)

Removes the extension of a file name.

Removes the extension of a file name.

Return Value: `File` name without extension.
Parameters: `s` Input file name.
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.15

CString **GetExt** (const CString &s)

Returns the extension of a file name.

Returns the extension of a file name.

Return Value: `The` extension.
Parameters: `s` Input filename
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.16

bool **CreateTest** (const CString &file)

Tests if a file can be created for writing.

Tests if a file can be created for writing.

Return Value: `True,` if the file could be created (and deleted again).
Parameters: `file` Input file name.
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.17

CString **AddBackSlash** (const CString &path)

Ensures that a string is terminated with a backslash

Ensures that a string is terminated with a backslash. If the already has a terminating backslash, nothing is done.

Return Value: `Backslash-terminated` output string.
Parameters: `path` Input string.
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.18

CString **GetPath** (const CString &fullfilename)

Returns the path of a filename.

Returns the path of a filename.

Return Value: `The` path to the filename.
Parameters: `fullfilename` Filename including any path.
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.19

CString **GetFilename** (const CString &filename)

Returns the file name of a path+file name string.

Returns the file name of a path+file name string.

Return Value: `The` file name including any extension, but without any path.
Parameters: `filename` Full qualified filename including path.
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.20

CString **ForceExt** (const CString &filename, const CString &ext)

Forces an extension onto a file name.

The method tests the extension of a filename. If the wanted extension is presented nothing is done. If not, the extension is appended.

Return Value: `The` file name including the wanted extension.
Parameters: `filename` Input file name.
`ext` Wanted extension.
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.21

bool **FileExists** (const CString &filename)

Tests if a file exists.

Tests if a file exists.

Return Value: `True` if the file exists.
Parameters: `filename` File name to test.
Author: Mikkel B. Stegmann
Version: 1-17-2003

31.22

void **ExpandImg2DyadicSize** (const CDMultiBand<TAAMPixel> &img, CDMultiBand<TAAMPixel> &out)

Expands an image to have dyadic size.

The method expands an image by using zero-padding so that the resulting image has width and height that are powers of two.

Return Value: `Nothing.`
Parameters:
`img` Input image.
`out` Output dyadic image.
Author: Mikkel B. Stegmann
Version: 4-22-2002

31.23

void **MirrorEdge** (CDMatrix &m, const CDMultiBand<TAAMPixel> &mask, int edgeWidth)

Mirrors the edge of an arbitrary shape.

Mirrors the edge of an arbitrary shape mask in a matrix using poor-mans mirroring.

Return Value: `Nothing.`
Parameters:
`img` Input matrix. Overwritten by mirrored version.
`mask` Image mask defining the shape.
`edgeWidth` The width of the edge.
Author: Mikkel B. Stegmann
Version: 4-25-2002

31.24

void **MirrorEdge** (CDMultiBand<TAAMPixel> &img, const CDMultiBand<TAAMPixel> &mask, int edgeWidth)

Mirrors the edge of an arbitrary shape.

Mirrors the edge of an arbitrary shape mask in an image using poor-mans mirroring.

Return Value: `Nothing.`

Parameters:

`img`	Input image. Overwritten by mirrored version.
`mask`	Image mask defining the shape.
`edgeWidth`	The width of the edge.

Author: Mikkel B. Stegmann

Version: 4-25-2002

31.25

bool **ShapeInsideImage** (const CAAMShape &s, const CDMultiBand<TAAMPixel> &img)

Tests if a shape is fully inside an image.

Tests if a shape is fully inside an image.

Return Value: `True` if the shape is fully inside the image.

Parameters:

`s`	Input shape.
`img`	Input image.

Author: Mikkel B. Stegmann

Version: 5-7-2002

31.26

void **CalcShapeDistances** (const CAAMShape &optimized, const CAAMShape &groundTruth, double &ptpt, double &ptcrv, CDVector* pvDists)

Calculates optimization results.

Calculates optimization results.

Return Value: `Nothing.`

Parameters:

`optimized`	Model shape.
`groundTruth`	Ground truth shape.
`ptpt`	Average point to point landmark error.
`ptcrv`	Average point to curve landmark error.NOTICE: This is not a symmetric measure!

Author: Mikkel B. Stegmann

Version: 4-10-2000

31.27

CString **Secs2Mins** (double secs)

Converts seconds to a MM:SS string.

Converts seconds to a MM:SS string.

Return Value: `The` time in MM:SS.
Parameters: `secs` Time in seconds.
Author: Mikkel B. Stegmann
Version: 8-2-2002

31.28

CString **FindVacantFilename** (const CString &filename_suggestion)

Finds a file name that is not 'occupied'.

This method finds a file name that is not 'occupied' by adding at number to the base part of the suggested file name.

Return Value: `An` unused filename resembling the suggstion.
Parameters: `filename_suggestion` Suggestion including extension.
Author: Mikkel B. Stegmann
Version: 9-3-2002

31.29

void **SampleTextures** (const CAAMShapeCollection &unalignedShapes, std::vector< CDVector > &vTextures, CAAMReferenceFrame &outputRF, const int imageReduction, const bool removeMean, const bool useTSP, const bool useConvexHull)

Samples a set of texture vectors given a set of shape in absolute (i.e. image) coordinates.

Samples a set of texture vectors given a set of shape in absolute (i.e. image) coordinates.

Return Value:

Parameters:

`unalignedShapes`	Shapes in absolute coordinates.
`vTextures`	The set of textures.
`outputRF`	The output reference frame generated for sampling the textures.
`removeMean`	If true the mean from each texture vector (i.e. the DC) isremoved.
`useTSP`	Use tangent space projection toalign the shapes.
`useConvexHull`	If true the convex hull is usedto determine the extent of a shape.

Author: Mikkel B. Stegmann

Version: 10-29-2002

31.30

void **RegistrationMovie** (const CString &filename, const std::vector<CDVector> &vTexture, const CAAMReferenceFrame &rf)

Writes a movie file containing a set of textures warped to their mean shape.

Writes a movie file containing a set of textures warped to their mean shape.

Return Value: `Nothing.`

Parameters:

`filename`	Output movie filename.
`vTexture`	The set of textures.
`rf`	The reference frame used for samplingthe textures.

Author: Mikkel B. Stegmann

Version: 10-29-2002

31.31

void **RegistrationMovie** (const CString &filename, const CString asfPath, const bool useConvexHull, const bool writeRefShape)

Writes a movie file containing all shapes from a directory warped to their mean shape.

Writes a movie file containing all shapes from a directory warped to their mean shape.

Return Value: `Nothing.`

Parameters:	`filename`	Output movie filename.
	`asfPath`	Path to annotation files.
	`useConvexHull`	If true the convex hull is usedto determine the extent of a shape.
	`writeRefShape`	If true the reference shape correspondingto the movie file is written as "regshape.asf".

Author: Mikkel B. Stegmann
Version: 10-29-2002

31.32

void **ASF2PTS** (const CString &path)

Converts AAM-API shape files (.asf) to the ISBE .pts format.

Converts AAM-API shape files (.asf) to the ISBE .pts format. Output is written in the directory 'pts'.

Return Value: `Nothing.`
Parameters: `path` Path to .asf files.
Author: Mikkel B. Stegmann
Version: 12-3-2002

31.33

double **SymmetricPtCrv** (const CAAMShape &s1, const CAAMShape &s2)

A symmetric point-to-curve measure.

A symmetric poin- to-curve measure, i.e.

SymmetricPtCrv(a,b) == SymmetricPtCrv(b,a);

Return Value: `The` average symmetric point-to-curve error over all landmarks.
Parameters: `s1` First shape
`s2` Second shape
See Also: DistEuclidianAssBorder
Author: Mikkel B. Stegmann
Version: 2-9-2003

31.34

double **ShapeOverlap** (const CAAMShape &model, const CAAMShape &gt)

Calculates the overlap between two shapes.

Calculates the overlap between two shapes as specified in "Active Shape Model Segmentation With Optimal Features" Bram van Ginneken et al., IEEE TMI 21(8) Aug. 2002.

Notice that this only makes sense for one-path closed shapes.

Return Value: `The` shape overlap (1 = perfect match, 0 = no overlap).
Parameters:
`model` Model shape
`gt` Ground truth shape
Author: Mikkel B. Stegmann
Version: 2-9-2003

31.35

std::vector<CString> **ReadLines** (const CString &filename)

Reads a file into an array of lines.

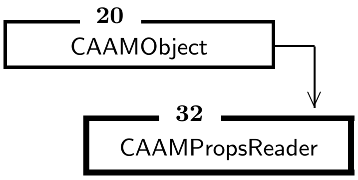
Reads a file into an array of lines.

Return Value: `A` vector of text lines.
Parameters: `filename` Name of input file.
Author: Mikkel B. Stegmann
Version: 2-22-2003

32 class **CAAMPropsReader** : public CAAMObject

Simple lo-fi property reader.

Inheritance



Public Members

32.1	void	**Sync** ()	*Skips whitespace and any commments preceeding the current file position*	219
32.2	bool	**IsValid** ()	*Returns true if the file is valid*	219
32.3	FILE*	**FH** ()	*Current open file*	219
32.4		**CAAMPropsReader** (const CString &filename)	*Constructor.*	219
32.5		**~CAAMPropsReader** ()	*Destructor.*	220
32.6	void	**SkipWhiteSpace** ()	*Increments the file pointer beyond any white space.*	220
32.7	void	**SkipRestOfLine** ()	*Increments the file pointer to the start of the next line.*	220
32.8	bool	**MoreNonWhiteSpaceOnLine** ()	*Returns true if more white space is present.*	220
32.9	void	**SkipComments** ()	*Increments the file pointer beyond any comments.*	221

Simple lo-fi property reader. Used as naive parser/scanner for the .asf, acf files et cetera.

See Also: CAAMBuilder::ReadACF(), CAAMShape::ReadASF()
Author: Mikkel B. Stegmann
Version: 4-17-2000

32.1 void **Sync** ()

Skips whitespace and any commments preceeding the current file position

Skips whitespace and any commments preceeding the current file position

32.2

bool **IsValid** ()

Returns true if the file is valid

Returns true if the file is valid

32.3

FILE* **FH** ()

Current open file

Current open file

32.4

CAAMPropsReader (const CString &filename)

Constructor.

Constructor. Opens the file.

Return Value: `Nothing.`
Parameters: `filename` The file to open.
Author: Mikkel B. Stegmann
Version: 4-17-2000

32.5

~CAAMPropsReader ()

Destructor.

Destructor. Closes the file.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-17-2000

32.6

void **SkipWhiteSpace** ()

Increments the file pointer beyond any white space.

Increments the file pointer beyond any white space.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-17-2000

32.7

void **SkipRestOfLine** ()

Increments the file pointer to the start of the next line.

Increments the file pointer to the start of the next line.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-17-2000

32.8

bool **MoreNonWhiteSpaceOnLine** ()

Returns true if more white space is present.

Returns true if more white space is present on the current line.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-17-2000

32.9

void **SkipComments** ()

Increments the file pointer beyond any comments.

Increments the file pointer beyond any comments.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-17-2000

33

class **CAAMVisualizer**

Class that visualizes different aspects of a model.

Public Members

33.1		**CAAMVisualizer** (const CAAMModel* pModel)	*Constructor.*	223
33.2		**~CAAMVisualizer** ()	*Destructor.*	223
33.3	void	**OptimizationMovie** (const std::vector<CAAMModel::CAAMOptState> &optStates, const CDMultiBand<TAAMPixel> &img, const char* filename, const int frameRate) const	*Visualizes the iterations during a model search.*	223
33.4	void	**ShapeStill** (const CDMultiBand<TAAMPixel> &img, const CAAMShape &shape, const char* filename)	*Plots a shape into an image and save it to disk.*	224
33.5	void	**WriteEigenImages** () const	*Writes the texture eigen modes as shape-free images.*	224
33.6	void	**WritePredictionImages** () const	*Writes the parameter update matrix as shape-free images.*	225
33.7	bool	**TextureMovie** (const CString &filename, const int nbModes, const double range, const int steps, const bool whiteBackground) const	*Generates movies files showing each mode of texture variation.*	225
33.8	bool	**ShapeMovie** (const CString &filename, const int nbModes, const double range, const int steps, const bool whiteBackground) const	*Generates movies files showing each mode of shape variation.*	226
33.9	bool	**CombinedMovie** (const CString &filename, const int nbModes, const double range, const int steps, const bool whiteBackground) const	*Generates movies files showing each mode of combined variation.*	226
33.10	bool	**RegistrationMovie** (const CString &filename, const std::vector<CDVector> &vTexture) const	*Generates a movie of the registered training set.*	227

Class that visualizes different aspects of a model.

For visualization of e.g. annotations without a model see CAAMUtil.

See Also: CAAMUtil
Author: Mikkel B. Stegmann
Version: 5-8-2002

33.1

CAAMVisualizer (const CAAMModel* pModel)

Constructor.

Constructor that sets the model pointer.

Return Value: `Nothing.`
Parameters: `pModel` Pointer to the model that you want visualised.
Author: Mikkel B. Stegmann
Version: 5-8-2002

33.2

~CAAMVisualizer ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 5-8-2002

33.3

void **OptimizationMovie** (const std::vector<CAAMModel::CAAMOptState> &optStates, const CDMultiBand<TAAMPixel> &img, const char* filename, const int frameRate) const

Visualizes the iterations during a model search.

This method visualizes the iterations during a model search. Output format is an AVI-file that you can put into e.g. Powerpoint to impress your supervisor with.

Return Value: `Nothing.`

Parameters:

`optStates`	An array of optimization states.
`img`	The image used for model searching.
`filename`	The output movie filename, e.g. 'search.avi'.
`frameRate`	The number of frame per second.

Author: Mikkel B. Stegmann

Version: 5-7-2002

33.4

void **ShapeStill** (const CDMultiBand<TAAMPixel> &img, const CAAMShape &shape, const char* filename)

Plots a shape into an image and save it to disk.

Plots a shape into an image and save it to disk.

Return Value: `Nothing.`

Parameters:

`img`	Image to plot shape onto.
`shape`	Shape in image coordinates.
`fileman`	Destination image file, e.g. 'result.bmp'.

Author: Mikkel B. Stegmann

Version: 5-7-2002

33.5

void **WriteEigenImages** () const

Writes the texture eigen modes as shape-free images.

This method converts each column of the texture eigen vectors into corresponding shape-free images and save them to disk in the BMP format.

Return Value: `Nothing.`

Author: Mikkel B. Stegmann

Version: 2-26-2002

33.6

void **WritePredictionImages** () const

Writes the parameter update matrix as shape-free images.

This method converts each column of the parameter update matrices into corresponding shape-free images and save them to disk in the BMP format.

Return Value:	`Nothing.`
Author:	Mikkel B. Stegmann
Version:	2-26-2002

33.7

bool **TextureMovie** (const CString &filename, const int nbModes, const double range, const int steps, const bool whiteBackground) const

Generates movies files showing each mode of texture variation.

Generates movies files showing each mode of texture variation by adjusting the each mode of variation to

+/- 'range' times <the mode standard deviation>

in 2*'step'+1 steps.

Return Value:	`true` on success, false on file errors.	
Parameters:	`filename`	The output filename excl. extension.
	`nbModes`	The number of modes to generate movies from.
	`range`	The rendered range in standard deviations.
	`steps`	The number of frames to generate to reach 'range'.Thus the total number of frames will be 2*step+1.(the +1 term is the mean texture).
	`whiteBackground`	If true the background is rendered in white.Default is black.
See Also:	Shape, Combined	
Author:	Mikkel B. Stegmann	
Version:	2-28-2000	

33.8

bool **ShapeMovie** (const CString &filename, const int nbModes, const double range, const int steps, const bool whiteBackground) const

Generates movies files showing each mode of shape variation.

Generates movies files showing each mode of shape variation by adjusting the each mode of variation to

+/- 'range' times <the mode standard deviation>

in 2*'step'+1 steps.

Return Value: `true` on success, false on file errors.

Parameters:

`filename`	The output filename excl. extension.
`nbModes`	The number of modes to generate movies from.
`range`	The rendered range in standard deviations.
`steps`	The number of frames to generate to reach 'range'.Thus the total number of frames will be 2*step+1.(the +1 term is the mean shape).
`whiteBackground`	If true the background is rendered in white.Default is black.

See Also: Texture, Combined

Author: Mikkel B. Stegmann

Version: 2-28-2000

33.9

bool **CombinedMovie** (const CString &filename, const int nbModes, const double range, const int steps, const bool whiteBackground) const

Generates movies files showing each mode of combined variation.

Generates movies files showing each mode of combined shape and texture variation by adjusting the each mode of variation to:

+/- 'range' times <the mode standard deviation>

in 2*'step'+1 steps.

Return Value: `true` on success, false on file errors.

Parameters:

`filename`	The output filename excl. extension.
`nbModes`	The number of modes to generate movies from.
`range`	The rendered range in standard deviations.
`steps`	The number of frames to generate to reach 'range'.Thus the total number of frames will be 2*step+1.(the +1 term is the mean shape).
`whiteBackground`	If true the background is rendered in white.Default is black.

See Also: Texture, Shape

Author: Mikkel B. Stegmann

Version: 2-25-2000

33.10

bool **RegistrationMovie** (const CString &filename, const std::vector<CDVector> &vTexture) const

Generates a movie of the registered training set.

Generates a movie of the registered training set by warping each texture to the mean shape and output this to an AVI-file.

Return Value: `true` on succes, false on file errors.

Parameters:
`filename` The output filename.
`m_vTexture` The vector of textures.

See Also: CombinedMovie, TextureMovie, ShapeMovie

Author: Mikkel B. Stegmann

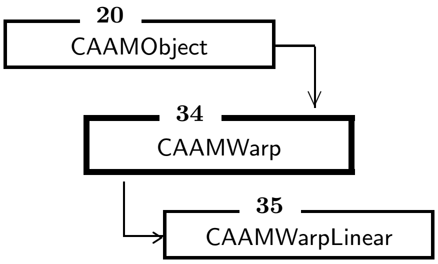
Version: 2-22-2000

34

class **CAAMWarp** : public CAAMObject

Base class for 2D warp classes.

Inheritance



Public Members

34.6		**CAAMWarp** ()	*Constructor*	229
34.7	virtual inline bool	**Warp** (const CAAMPoint &in, CAAMPoint &out) const	*Warps point 'in' to point 'out'.*	229
34.8	virtual void	**SetDestShape** (const CAAMShape &s)	*Sets the shape to warp to*	229
34.9	void	**UseConvexHull** (bool enable = true)	*Allows warping inside the convex hull (default=off)*	229
34.10		~**CAAMWarp** ()	*Destructor.*	230
34.11	void	**SetSrcShape** (const CAAMShape &s)	*Sets the shape to warp from.*	230

Protected Members

34.5	bool	**m_bUseConvexHull**	*Allows warping inside the convex hull*	230

Private Members

34.1	double	**m_dSrcShapeMinX**	*Source shape extents*	230
34.2	double	**m_dSrcShapeMaxX**	*Source shape extents*	231
34.3	double	**m_dSrcShapeMinY**	*Source shape extents*	231
34.4	double	**m_dSrcShapeMaxY**	*Source shape extents*	231

CAAMWarp defines a 2D warp function between two shapes with an equal amount of points.

See Also: CAAMWarpLinear
Author: Mikkel B. Stegmann

Version: 02-14-2000

34.6

CAAMWarp ()

Constructor

Constructor

34.7

virtual inline bool **Warp** (const CAAMPoint &in, CAAMPoint &out) const

Warps point 'in' to point 'out'.

Warps the point 'in' to the point 'out' using the two shapes as irregular point clouds defining a 2D warp function. 'in' defines a point contained in the source shape and 'out' is the corrosponding point in the destination shape.

Return Value: `True` if the warp can be done, false if not.
Parameters: `in` Input point.
`out` Output point.
Author: Mikkel B. Stegmann
Version: 02-14-2000

34.8

virtual void **SetDestShape** (const CAAMShape &s)

Sets the shape to warp to

Sets the shape to warp to

34.9

void **UseConvexHull** (bool enable = true)

Allows warping inside the convex hull (default=off)

Allows warping inside the convex hull (default=off)

34.10

˜CAAMWarp ()

Destructor.

Destructor.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-26-2000

34.11

void **SetSrcShape** (const CAAMShape &s)

Sets the shape to warp from.

Sets the shape to warp from.

Return Value: `Nothing.`
Author: Mikkel B. Stegmann
Version: 4-26-2000

34.5

bool **m_bUseConvexHull**

Allows warping inside the convex hull

Allows warping inside the convex hull

34.1

double **m_dSrcShapeMinX**

Source shape extents

Source shape extents

34.2

double **m_dSrcShapeMaxX**

Source shape extents

Source shape extents

34.3

double **m_dSrcShapeMinY**

Source shape extents

Source shape extents

34.4

double **m_dSrcShapeMaxY**

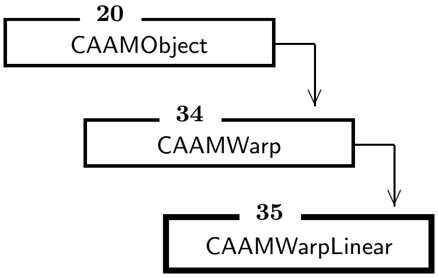
Source shape extents

Source shape extents

35 class **CAAMWarpLinear** : public CAAMWarp

Piece-wise affine warping between two shapes.

Inheritance



Public Members

35.3	bool	**HasSrcShape** () const	*Returns true if the source shape has been set*	233
35.4		**CAAMWarpLinear** (bool useSrcDelaunay)	*Constructor.*	233
35.5	void	**SetSrcShape** (const CAAMShape &s)	*Sets the shape to warp from.*	233
35.6	void	**SetDestShape** (const CAAMShape &s)	*Sets the shape to warp to.*	234
35.7	bool	**Warp** (const CAAMPoint &in, CAAMPoint &out) const	*Warps point 'in' to point 'out' (if possible).*	234

Private Members

35.1	CAAMMesh	**m_Mesh**	*The Delaunay Triangulisation of the source or dest shape*	234
35.2	std::vector<CAAMPoint>	**m_vDestPoints**	*The points of the destination shape*	234

This class implements a piece-wise affine warping between two shapes using a Delaunay Triangulisation of the source shape. This triangulisation is the used for both the source and destination shape. The corrospondance between points and triangles gives a continuous deformation field inside the triangles. Since the deformation is performed on a linear per-triangle basis, the field is not smooth across triangles.

See Also: CAAMWarp
Author: Mikkel B. Stegmann
Version: 02-14-2000

35.3

bool **HasSrcShape** () const

Returns true if the source shape has been set

Returns true if the source shape has been set

35.4

CAAMWarpLinear (bool useSrcDelaunay)

Constructor.

Constructor.

Return Value: `Nothing.`
Parameters: `useSrcDelaunay` If true the Delaunay of the source shape is used.
Author: Mikkel B. Stegmann
Version: 4-26-2000

35.5

void **SetSrcShape** (const CAAMShape &s)

Sets the shape to warp from.

Sets the shape to warp from.

Return Value: `Nothing.`
Parameters: `s` Input shape.
Author: Mikkel B. Stegmann
Version: 4-26-2000

35.6

void **SetDestShape** (const CAAMShape &s)

Sets the shape to warp to.

Sets the shape to warp to.

Return Value:	`Nothing.`
Parameters:	`s` Input shape.
Author:	Mikkel B. Stegmann
Version:	4-26-2000

35.7

bool **Warp** (const CAAMPoint &in, CAAMPoint &out) const

Warps point 'in' to point 'out' (if possible).

Warps point 'in' to point 'out' (if possible).

Return Value:	`True` if 'in' is inside the source mesh, false if not.
Parameters:	`in` Input point. `out` Output point.
Author:	Mikkel B. Stegmann
Version:	4-26-2000

35.1

CAAMMesh **m_Mesh**

The Delaunay Triangulisation of the source or dest shape

The Delaunay Triangulisation of the source or dest shape

35.2

std::vector<CAAMPoint> **m_vDestPoints**

The points of the destination shape

The points of the destination shape

36

CAAMAnalyzeSynthesize* **AAMLoadAnalyzerSynthesizer** (FILE* fh, const CAAMReferenceFrame &rf)

Analyzer/Synthesizer loader.

The function loads an Analyzer/Synthesizer from a stream, instantiates the correct concrete class and returns a base class pointer.

Return Value: `A` pointer to an Analyzer/Synthesize object created on the heap.

Parameters:

`fh`	Open binary stream.
`pModel`	Model pointer (currently not used).

Author: Mikkel B. Stegmann

Version: 6-13-2002

37

CAAMTransferFunction* **AAMLoadTransferFunction** (FILE* fh, CAAM-Model* pModel)

Transfer function loader.

The function loads a transfer function from a stream, instantiates the correct concrete class and returns a base class pointer.

Return Value: `A` pointer to a transfer function created on the heap.
Parameters:
`fh` Open binary stream.
`pModel` Model pointer (currently not used).
Author: Mikkel B. Stegmann
Version: 1-29-2002

38

template<class TPixel> void **DIVAEnlargeImage** (CVisImage<TPixel> &img, int nFactor)

Enlarge the image.

Enlarge the image.

Return Value:

Parameters:

`img`	Input image.
`nFactor`	The factor to enlarge the width and height of the image.

See Also: DIVAReduceImage

Author: Rune Fisker

Version: 2-22-1999

39

template<class TPixel> void **DIVAReduceImage** (CVisImage<TPixel> &img, int nFactor)

Reduce the image.

Reduce the image.

Return Value:

Parameters:	`img`	Input image.
	`nFactor`	The factor to reduce the width and height of the image.
See Also:	DIVAEnlargeImage	
Author:	Rune Fisker	
Version:	2-22-1999	

Class Graph