Open Source SSAM Data Dictionary

New Global Systems
for Intelligent
Transportation
Management

Phone and Fax: (662) 324-4084
li.zhang@ngsim.com

75 Cavalier Blvd.
Florence, KY 41042

May 1, 2017

Table of Contents

Namespace Index 2
Hierarchical Index 3
Class Index 4
MotPredNameSpace 5
SSAMFuncs...6
Class Documentation...7
Conflict..7
SSAMFuncs::Dimensions 10
MotPredNameSpace::EvasiveAction 12
Event..14
InitEventParams 19
SSAMFuncs::InputFormat 20
MotPredNameSpace::NormalAdaption...21
MotPredNameSpace::NormAngle...23
SSAMPoint::point 24
MotPredNameSpace::PredMethod 25
MotPredNameSpace::PredObj...28
MotPredNameSpace::PredTraj 29
MotPredNameSpace::PredTrajConstant..31
MotPredNameSpace::PredTrajFactory...33
MotPredNameSpace::PredTrajRandom 34
SSAMFuncs::SSAM..36
SSAMException..44
Summary 45
SSAMFuncs::TimeStepData 47
MotPredNameSpace::TriangularDistri..48
SSAMFuncs::TrjRecord...49
Vehicle 51
ZoneGrid::Zone..56
ZoneGrid 57
Index..60

Document purpose

This document is a software data dictionary for the Surrogate Safety Assessment Model (SSAM) -3.0. SSAM is used to perform safety analysis. This document lists all classes defined in the open source software along with their member variables and functions, and it may be of use to simulation software developers, researchers, transportation engineers, and safety engineers.

Namespace Index

Namespace List

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

MotPredNameSpace 5
SSAMFuncs 6

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- Conflict 7
- SSAMFuncs::Dimensions 10
- Event 14
- InitEventParams 19
- SSAMFuncs::InputFormat 20
- MotPredNameSpace::NormAngle 23
- SSAMPoint::point 24
- MotPredNameSpace::PredMethod 25
  - MotPredNameSpace::EvasiveAction 12
  - MotPredNameSpace::NormalAdaption 21
- MotPredNameSpace::PredObj 28
- MotPredNameSpace::PredTraj 29
  - MotPredNameSpace::PredTrajConstant 31
  - MotPredNameSpace::PredTrajRandom 34
- MotPredNameSpace::PredTrajFactory 33
- runtime_error
  - SSAMException 44
- SSAMFuncs::SSAM 36
- Summary 45
- SSAMFuncs::TimeStepData 47
- MotPredNameSpace::TriangularDistri 48
- SSAMFuncs::TrjRecord 49
- Vehicle 51
- ZoneGrid::Zone 56
- ZoneGrid 57

Class Index

Class List

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

Conflict 7
SSAMFuncs::Dimensions 10
MotPredNameSpace::EvasiveAction 12
Event 14
InitEventParams 19
SSAMFuncs::InputFormat 20
MotPredNameSpace::NormalAdaption 21
MotPredNameSpace::NormAngle 23
SSAMPoint::point 24
MotPredNameSpace::PredMethod 25
MotPredNameSpace::PredObj 28
MotPredNameSpace::PredTraj 29
MotPredNameSpace::PredTrajConstant 31
MotPredNameSpace::PredTrajFactory 33
MotPredNameSpace::PredTrajRandom 34
SSAMFuncs::SSAM 36
SSAMException 44
Summary 45
SSAMFuncs::TimeStepData 47
MotPredNameSpace::TriangularDistri 48
SSAMFuncs::TrjRecord 49
Vehicle 51
ZoneGrid::Zone 56
ZoneGrid 57

Namespace Documentation

MotPredNameSpace Namespace Reference

Classes

- class **EvasiveAction**
- class **NormalAdaption**
- class **NormAngle**
- class **PredMethod**
- struct **PredObj**
- class **PredTraj**
- class **PredTrajConstant**
- class **PredTrajFactory**
- class **PredTrajRandom**
- class **TriangularDistri**

Typedefs

- typedef std::shared_ptr< **PredTraj** > **SP_PredTraj**
- typedef std::shared_ptr< **PredTrajRandom** > **SP_PredTrajRandom**
- typedef std::shared_ptr< **PredTrajConstant** > **SP_PredTrajConstant**
- typedef std::shared_ptr< **NormalAdaption** > **SP_NormalAdaption**
- typedef std::shared_ptr< **EvasiveAction** > **SP_EvasiveAction**

Detailed Description

MotPredNameSpace organizes classes, structs and functions for implementing motion prediction methods to calculate P(UEA), mTTC, mPET.

Typedef Documentation

typedef std::shared_ptr<EvasiveAction> MotPredNameSpace::SP_EvasiveAction

Smart pointer type to **EvasiveAction** class.

typedef std::shared_ptr<NormalAdaption> MotPredNameSpace::SP_NormalAdaption

Smart pointer type to **NormalAdaption** class.

typedef std::shared_ptr<PredTraj> MotPredNameSpace::SP_PredTraj

Smart pointer type to **PredTraj** class.

typedef std::shared_ptr<PredTrajConstant> MotPredNameSpace::SP_PredTrajConstant

Smart pointer type to **PredTrajConstant** class.

typedef std::shared_ptr<PredTrajRandom> MotPredNameSpace::SP_PredTrajRandom

Smart pointer type to **PredTrajRandom** class.

SSAMFuncs Namespace Reference

Classes

- class **Dimensions**
- struct **InputFormat**
- class **SSAM**
- class **TimeStepData**
- class **TrjRecord**

Typedefs

- typedef std::shared_ptr< **Dimensions** > **SP_Dimensions**
- typedef std::shared_ptr< **TrjRecord** > **SP_TrjRecord**
- typedef std::shared_ptr< **TimeStepData** > **SP_TimeStepData**
- typedef std::pair< std::string, std::list< **TrjRecord** > *> **TrjDataList**
- typedef std::pair< int, int > **VehiclePair**
- typedef std::shared_ptr< **SSAM** > **SP_SSAM**

Detailed Description

SSAMFuncs namespace encloses all **SSAM** DLL classes

Typedef Documentation

typedef std::shared_ptr<Dimensions> SSAMFuncs::SP_Dimensions

Smart pointer type to **Dimensions** class.

typedef std::shared_ptr<SSAM> SSAMFuncs::SP_SSAM

Smart pointer type to **SSAM** class.

typedef std::shared_ptr<TimeStepData> SSAMFuncs::SP_TimeStepData

Smart pointer type to **TimeStepData** class.

typedef std::shared_ptr<TrjRecord> SSAMFuncs::SP_TrjRecord

Smart pointer type to **TrjRecord** class.

typedef std::pair<std::string, std::list<TrjRecord>* > SSAMFuncs::TrjDataList

Define a pair of project name and a list of TRJ records as one TRJ Input source

Class Documentation

Conflict Class Reference

`#include <Conflict.h>`

Public Types

- enum **SSAM_MEASURE** { **trjFile_MEASURE**, **tMinTTC_MEASURE**, **xMinPET_MEASURE**, **yMinPET_MEASURE**, **zMinPET_MEASURE**, **TTC_MEASURE**, **PET_MEASURE**, **MaxS_MEASURE**, **DeltaS_MEASURE**, **DR_MEASURE**, **MaxD_MEASURE**, **MaxDeltaV_MEASURE**, **ConflictAngle_MEASURE**, **ClockAngle_MEASURE**, **ConflictType_MEASURE**, **PostCrashV_MEASURE**, **PostCrashHeading_MEASURE**, **FirstVID_MEASURE**, **FirstLink_MEASURE**, **FirstLane_MEASURE**, **FirstLength_MEASURE**, **FirstWidth_MEASURE**, **FirstHeading_MEASURE**, **FirstVMinTTC_MEASURE**, **FirstDeltaV_MEASURE**, **xFirstCSP_MEASURE**, **yFirstCSP_MEASURE**, **xFirstCEP_MEASURE**, **yFirstCEP_MEASURE**, **SecondVID_MEASURE**, **SecondLink_MEASURE**, **SecondLane_MEASURE**, **SecondLength_MEASURE**, **SecondWidth_MEASURE**, **SecondHeading_MEASURE**, **SecondVMinTTC_MEASURE**, **SecondDeltaV_MEASURE**, **xSecondCSP_MEASURE**, **ySecondCSP_MEASURE**, **xSecondCEP_MEASURE**, **ySecondCEP_MEASURE**, **PUEA_MEASURE**, **mTTC_MEASURE**, **mPET_MEASURE** }
- enum **CONFLICT_TYPE** { **UNCLASSIFIED**, **CROSSING**, **REAR_END**, **LANE_CHANGE** }

Public Member Functions

- **Conflict** (SP_Event e, const std::string &file)
- float **GetMeasure** (int i)
- std::string **GetStrippedTrjName** ()
- std::string **GetValueString** (int i)

Public Attributes

- std::string **trjFile**
- float **tMinTTC**
- float **xMinPET**
- float **yMinPET**
- float **zMinPET**
- float **TTC**
- float **PET**
- float **MaxS**
- float **DeltaS**
- float **DR**
- float **MaxD**
- float **MaxDeltaV**
- float **ConflictAngle**
- std::string **ClockAngle**
- int **ConflictType**
- float **PostCrashV**
- float **PostCrashHeading**
- int **FirstVID**
- int **FirstLink**
- int **FirstLane**
- float **FirstLength**
- float **FirstWidth**
- float **FirstHeading**

- float **FirstVMinTTC**
- float **FirstDeltaV**
- float **xFirstCSP**
- float **yFirstCSP**
- float **xFirstCEP**
- float **yFirstCEP**
- int **SecondVID**
- int **SecondLink**
- int **SecondLane**
- float **SecondLength**
- float **SecondWidth**
- float **SecondHeading**
- float **SecondVMinTTC**
- float **SecondDeltaV**
- float **xSecondCSP**
- float **ySecondCSP**
- float **xSecondCEP**
- float **ySecondCEP**
- float **PUEA**
- float **mTTC**
- float **mPET**

Static Public Attributes

- static const int **NUM_MEASURES** = 44
- static const std::string **MEASURE_LABEL** [**NUM_MEASURES**]
- static const int **NUM_CONFLICT_TYPES** = 4
- static const std::string **CONFLICT_TYPE_LABEL** [**NUM_CONFLICT_TYPES**]
- static const bool **SUM_MEASURES** [**NUM_MEASURES**]
- static const bool **KEY_MEASURES** [**NUM_MEASURES**]

Detailed Description

Conflict manages the safety measures for one confirmed conflict. Reference for measure meanings: Surrogate Safety Assessment Model and Validation: Final Report, Publication No. FHWA-HRT-08-051, JUNE 2008, CHAPTER 2. SSAM SOFTWARE, TERMS AND DEFINITIONS, Page 20-25.

Member Enumeration Documentation

enum Conflict::CONFLICT_TYPE

CONFLICT_TYPE enum defines integer representing each conflict type

enum Conflict::SSAM_MEASURE

SSAM_MEASURE enum defines the column number of safety measures

Constructor & Destructor Documentation

Conflict::Conflict (SP_Event *e*, const std::string & *file*) `[inline]`

A constructor populates safety measures from a conflict event.

Parameters:

pEvent	A pointer to a conflict event.

trjfile	Name of the trj file.

Member Function Documentation

float Conflict::GetMeasure (int *i*) `[inline]`

Get a numeric safety measure using its column order.

Parameters:

i	column order of the safety measure, starting from 0.

std::string Conflict::GetStrippedTrjName () `[inline]`

Get the trj file name without path.

std::string Conflict::GetValueString (int *i*) `[inline]`

Get a safety measure value as a string using its column order.

Parameters:

i	column order of the safety measure, starting from 0.

Member Data Documentation

const std::string Conflict::CONFLICT_TYPE_LABEL `[static]`

```
Initial value:=
{
   "unclassified",
   "crossing",
   "rear end",
   "lane change"
}
```

Strings represent names of conflict types

const bool Conflict::KEY_MEASURES `[static]`

An array of flags indicate whether a safety measure should be displayed in conflict list

const std::string Conflict::MEASURE_LABEL `[static]`

Strings represent names of safety measures

const int Conflict::NUM_CONFLICT_TYPES = 4 `[static]`

The number of conflict types

const int Conflict::NUM_MEASURES = 44 `[static]`

The number of SSAM measures to record

const bool Conflict::SUM_MEASURES `[static]`

An array of flags indicate whether a safety measure should be summarized

SSAMFuncs::Dimensions Class Reference

```
#include <SSAM.h>
```

Public Member Functions

- **Dimensions** (const **Dimensions** &rhs)
- **Dimensions** & **operator=** (const **Dimensions** &rhs)
- void **Validate** ()
- void **Print** (std::ostream &output)
- int **GetMinX** ()
- int **GetMaxX** ()
- int **GetMinY** ()
- int **GetMaxY** ()
- int **GetUnits** ()
- float **GetScale** ()
- void **SetMinX** (int i)
- void **SetMaxX** (int i)
- void **SetMinY** (int i)
- void **SetMaxY** (int i)
- void **SetUnits** (int i)
- void **SetScale** (float f)

Static Public Attributes

- static const int **ENGLISH_UNITS** = 0
- static const int **METRIC_UNITS** = 1

Private Member Functions

- void **CopyValues** (const **Dimensions** &rhs)

Private Attributes

- int **m_MinX**
- int **m_MaxX**
- int **m_MinY**
- int **m_MaxY**
- int **m_Units**
- float **m_Scale**

Detailed Description

Dimensions specifies the extent of the rectangular region of the vehicle observation area in terms of x-y coordinates.

Member Function Documentation

void Dimensions::Print (std::ostream & *output*)

Print current dimension parameters.

Parameters:

output	the output stream

void Dimensions::Validate ()

Validate current dimension parameters.

Member Data Documentation

const int SSAMFuncs::Dimensions::ENGLISH_UNITS = 0 `[static]`

feet, feet/sec, feet/sec*sec

int SSAMFuncs::Dimensions::m_MaxX `[private]`

Right edge of the observation area.

int SSAMFuncs::Dimensions::m_MaxY `[private]`

Top edge of the observation area.

int SSAMFuncs::Dimensions::m_MinX `[private]`

Left edge of the observation area.

int SSAMFuncs::Dimensions::m_MinY `[private]`

Bottom edge of the observation area.

float SSAMFuncs::Dimensions::m_Scale `[private]`

Distance per unit of X or Y

int SSAMFuncs::Dimensions::m_Units `[private]`

Engligh or Metric units

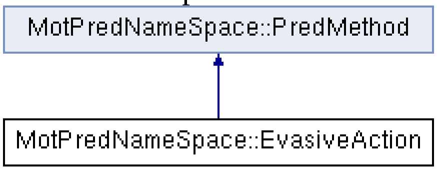
const int SSAMFuncs::Dimensions::METRIC_UNITS = 1 `[static]`

meters, meters/sec, meters/sec*sec

MotPredNameSpace::EvasiveAction Class Reference

`#include <MotionPrediction.h>`

Inheritance diagram for MotPredNameSpace::EvasiveAction:



Public Member Functions

- **EvasiveAction** (int nPredTrajs, double maxSpeed, double maxSteering, double maxAccRate, double minAccRate)
- virtual void **GenPredTrajs** (const **PredObj** &obj, std::vector< **SP_PredTraj** > &predTrajs)
- float **CalcPUEA** (const **PredObj** &obj1, const **PredObj** &obj2, double collisionThreshold, int nSteps)

Additional Inherited Members

Detailed Description

EvasiveAction defines the evasive action motion prediction method used for calculating P(UEA).

Constructor & Destructor Documentation

MotPredNameSpace::EvasiveAction::EvasiveAction (int *nPredTrajs*, double *maxSpeed*, double *maxSteering*, double *maxAccRate*, double *minAccRate*) `[inline]`

Constructor genarates an evasive action motion prediction method

Parameters:

nPredTrajs	number of trajectories to predict
maxSpeed	the maximum speed allowed in the roadway network
maxSteering	max of triangular distribution for steering angle: radians/s
maxAccRate	max of triangular distribution for acceleration rate: ft/s^2
minAccRate	min of triangular distribution for acceleration rate: ft/s^2

Member Function Documentation

float MotPredNameSpace::EvasiveAction::CalcPUEA (const PredObj & *obj1*, const PredObj & *obj2*, double *collisionThreshold*, int *nSteps*)

Calculate P(UEA)

Parameters:

obj1	initial position and velocity of first vehicle
obj2	initial position and velocity of second vehicle
collisionThreshold	a distance threshold to determine whether two vehicles collide
nSteps	number of steps to detect

Returns:

the calcualted P(UEA)

void MotPredNameSpace::EvasiveAction::GenPredTrajs (const PredObj & *obj*, std::vector< SP_PredTraj > & *predTrajs*) `[virtual]`

Generate a set of trajectories

Parameters:

	obj	initial vehicle position and velocity
out	*predTrajs*	the set of generated trajectories

Implements **MotPredNameSpace::PredMethod** (*p.26*).

Event Class Reference

```
#include <Event.h>
```

Public Member Functions

- **Event** (const **InitEventParams** &params)
- void **AddVehicleData** (SP_Vehicle v1, SP_Vehicle v2)
- bool **AnalyzeData** (float t)
- float **GetMinTTCTime** ()
- float **GetXMinPET** ()
- float **GetYMinPET** ()
- float **GetZMinPET** ()
- float **GetTTC** ()
- float **GetPET** ()
- float **GetMaxS** ()
- float **GetDeltaS** ()
- float **GetDR** ()
- float **GetMaxD** ()
- float **GetMaxDeltaV** ()
- float **GetConflictAngle** ()
- std::string **GetClockAngleString** ()
- int **GetConflictType** ()
- float **GetPostCrashV** ()
- float **GetPostCrashHeading** ()
- int **GetFirstVID** ()
- int **GetFirstLink** ()
- int **GetFirstLane** ()
- float **GetFirstLength** ()
- float **GetFirstWidth** ()
- float **GetFirstHeading** ()
- float **GetFirstVMinTTC** ()
- float **GetFirstDeltaV** ()
- float **GetXFirstCSP** ()
- float **GetYFirstCSP** ()
- float **GetXFirstCEP** ()
- float **GetYFirstCEP** ()
- int **GetSecondVID** ()
- int **GetSecondLink** ()
- int **GetSecondLane** ()
- float **GetSecondLength** ()
- float **GetSecondWidth** ()
- float **GetSecondHeading** ()
- float **GetSecondVMinTTC** ()
- float **GetSecondDeltaV** ()
- float **GetXSecondCSP** ()
- float **GetYSecondCSP** ()
- float **GetXSecondCEP** ()
- float **GetYSecondCEP** ()
- float **GetPUEA** ()
- float **GetMTTC** ()
- float **GetMPET** ()
- bool **IsConflict** ()

Static Public Attributes

- static const float **INVALID_SSM_VALUE** = 99.0

Private Member Functions

- void **CalcMeasures** ()

Private Attributes

- float **tMinTTC**
- float **xMinPET**
- float **yMinPET**
- float **zMinPET**
- float **TTC**
- float **PET**
- float **MaxS**
- float **DeltaS**
- float **DR**
- float **MaxD**
- float **MaxDeltaV**
- float **ConflictAngle**
- std::string **ClockAngleString**
- int **ConflictType**
- float **PostCrashV**
- float **PostCrashHeading**
- int **FirstVID**
- int **FirstLink**
- int **FirstLane**
- float **FirstLength**
- float **FirstWidth**
- float **FirstHeading**
- float **FirstVMinTTC**
- float **FirstDeltaV**
- float **xFirstCSP**
- float **yFirstCSP**
- float **xFirstCEP**
- float **yFirstCEP**
- int **SecondVID**
- int **SecondLink**
- int **SecondLane**
- float **SecondLength**
- float **SecondWidth**
- float **SecondHeading**
- float **SecondVMinTTC**
- float **SecondDeltaV**
- float **xSecondCSP**
- float **ySecondCSP**
- float **xSecondCEP**
- float **ySecondCEP**
- float **PUEA**
- float **mTTC**
- float **mPET**
- int **m_LowVID**
- int **m_HighVID**
- std::vector< SP_Vehicle > **m_LowVData**
- std::vector< SP_Vehicle > **m_HighVData**
- float **m_MaxTTC**
- float **m_MaxPET**

- int **m_RearEndAngle**
- int **m_CrossingAngle**
- float **m_StepSize**
- float **m_FirstTTC**
- float **m_LastTTC**
- float **m_PreTimeStep**
- float **m_FirstPET**
- float **m_LastPET**
- int **m_LastTTCIdx**
- int **m_LastPETIdx**
- bool **m_IsActive**
- bool **m_IsConflict**
- bool **m_IsPETComplete**
- bool **m_IsCalculatePUEA**
- **MotPredNameSpace::SP_NormalAdaption m_pNormalAdaption**
- **MotPredNameSpace::SP_EvasiveAction m_pEvasiveAction**
- int **m_NSteps**
- int **m_TotalSteps**
- double **m_CollisionThreshold**

Detailed Description

Event maintains the continueous vehicle data for the pair of vehicles involved in one conflict event, and calculates safety measures when the conflict is confirmed. Safety measure variables are defined same as the safety measures defined in document: Surrogate Safety Assessment Model and Validation: Final Report, Publication No. FHWA-HRT-08-051, JUNE 2008, CHAPTER 2. SSAM SOFTWARE, TERMS AND DEFINITIONS, Page 20-25.

Constructor & Destructor Documentation

Event::Event (const InitEventParams & *params*)

Create a new conflict event using the initial parameters.

Parameters:

params	Initial parameters.

Member Function Documentation

void Event::AddVehicleData (SP_Vehicle *v1*, SP_Vehicle *v2*)

Add new vehicle data.

Parameters:

v1	First vehicle.
v2	Second vehicle.

bool Event::AnalyzeData (float *t*)

Analyze event data up this time step.

Parameters:

t	This time step for analysis.

void Event::CalcMeasures () `[private]`

Calculate safety measures.

Member Data Documentation

double Event::m_CollisionThreshold `[private]`

a distance threshold to determine whether two vehicles collide

int Event::m_CrossingAngle `[private]`

Crossing Angle Threshold

float Event::m_FirstPET `[private]`

First time step of PET interval

float Event::m_FirstTTC `[private]`

First time step when conflict of vehicles is detected

std::vector<SP_Vehicle> Event::m_HighVData `[private]`

Data of vehicle with higher ID

int Event::m_HighVID `[private]`

Higher ID of the pair of vehicles

bool Event::m_IsActive `[private]`

Flag of whether TTC values are still less than the threshold

bool Event::m_IsCalculatePUEA `[private]`

Flag to calculate P(UEA), mTTC and mPET

bool Event::m_IsConflict `[private]`

Flag of whether current event is a conflict

bool Event::m_IsPETComplete `[private]`

Flag of whether PET calculations are complete

float Event::m_LastPET `[private]`

Last time step of PET interval

int Event::m_LastPETIdx `[private]`

Index in the **Vehicle** data array of the last location of the first vehicle for which PET is available

float Event::m_LastTTC `[private]`

Last time step when vehicles in collision

int Event::m_LastTTCIdx `[private]`

Index in the **Vehicle** data array of the last TTC

std::vector<SP_Vehicle> Event::m_LowVData `[private]`

Data of vehicle with lower ID

int Event::m_LowVID `[private]`

Lower ID of the pair of vehicles

float Event::m_MaxPET `[private]`

Max PET threshold

float Event::m_MaxTTC `[private]`

Max TTC threshold

int Event::m_NSteps `[private]`

Number of steps per second

float Event::m_PreTimeStep `[private]`

Previous time step when vehicles in collision

int Event::m_RearEndAngle `[private]`

Rear-end Angle Threshold

float Event::m_StepSize `[private]`

Step size when decrementing maxTTC to find the exact TTC

int Event::m_TotalSteps `[private]`

Total number of steps to detect collision or crossing zone

InitEventParams Struct Reference

```
#include <Event.h>
```

Public Attributes

- SP_Vehicle **m_V1**
- SP_Vehicle **m_V2**
- float **m_MaxTTC**
- float **m_MaxPET**
- int **m_RearEndAngleThreshold**
- int **m_CrossingAngleThreshold**
- bool **m_IsCalcPUEA**
- int **m_NSteps**
- double **m_CollisionThreshold**
- **MotPredNameSpace::SP_NormalAdaption m_pNormalAdaption**
- **MotPredNameSpace::SP_EvasiveAction m_pEvasiveAction**

Detailed Description

InitParams organizes parameters for creating a conflict event.

Member Data Documentation

double InitEventParams::m_CollisionThreshold

a distance threshold to determine whether two vehicles collide

int InitEventParams::m_CrossingAngleThreshold

Crossing Angle Threshold

bool InitEventParams::m_IsCalcPUEA

Flag to indicate whether to calculate P(UEA), mTTC, mPET

float InitEventParams::m_MaxPET

Max PET threshold

float InitEventParams::m_MaxTTC

Max TTC threshold

int InitEventParams::m_RearEndAngleThreshold

Rear-end Angle Threshold

SP_Vehicle InitEventParams::m_V1

First **Vehicle** in the conflict event

SP_Vehicle InitEventParams::m_V2

Second **Vehicle** in the conflict event

SSAMFuncs::InputFormat Struct Reference

```
#include <SSAM.h>
```

Public Member Functions

- void **Print** (std::ostream &output)

Public Attributes

- char **m_Endian**
- float **m_Version**
- int **m_ZOption**

Detailed Description

InputFormat specifies the byte format and file format of the trj source

---

Member Function Documentation

void InputFormat::Print (std::ostream & *output*)

Print current format parameters.

Parameters:

output	the output stream

Member Data Documentation

char SSAMFuncs::InputFormat::m_Endian

Byte endianness: 'L' - little endian; 'B' - big endian

float SSAMFuncs::InputFormat::m_Version

TRJ file format version

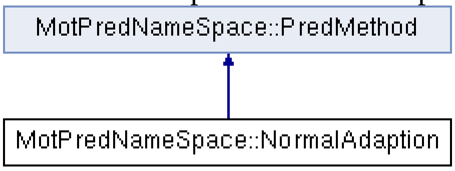
int SSAMFuncs::InputFormat::m_ZOption

Flag to indicate whether to use z-value

MotPredNameSpace::NormalAdaption Class Reference

```
#include <MotionPrediction.h>
```

Inheritance diagram for MotPredNameSpace::NormalAdaption:


MotPredNameSpace::PredMethod
MotPredNameSpace::NormalAdaption

Public Member Functions

- **NormalAdaption** (int nPredTrajs, double maxSpeed, double maxSteering, double maxAccRate)
- virtual void **GenPredTrajs** (const **PredObj** &obj, std::vector< **SP_PredTraj** > &predTrajs)
- void **CalcMTTCMPET** (const **PredObj** &obj1, const **PredObj** &obj2, double collisionThreshold, int nSteps, float &mTTC, float &mPET)

Additional Inherited Members

Detailed Description

NormalAdaption defines the normal adaption motion prediction method used for calculating mTTC and mPET.

Constructor & Destructor Documentation

MotPredNameSpace::NormalAdaption::NormalAdaption (int *nPredTrajs*, double *maxSpeed*, double *maxSteering*, double *maxAccRate*) `[inline]`

Constructor genarates a normal adaption motion prediction method

Parameters:

nPredTrajs	number of trajectories to predict
maxSpeed	the maximum speed allowed in the roadway network
maxSteering	max of triangular distribution for steering angle: radians/s
maxAccRate	max of triangular distribution for acceleration rate: ft/s^2

Member Function Documentation

void MotPredNameSpace::NormalAdaption::CalcMTTCMPET (const PredObj & *obj1*, const PredObj & *obj2*, double *collisionThreshold*, int *nSteps*, float & *mTTC*, float & *mPET*)

Calculate mTTC and mPET

Parameters:

	obj1	initial position and velocity of first vehicle
	obj2	initial position and velocity of second vehicle
	collisionThreshold	a distance threshold to determine whether two vehicles collide
	nSteps	number of steps to detect
out	*mTTC*	the calculated mTTC
out	*mPET*	the calculated mPET

void MotPredNameSpace::NormalAdaption::GenPredTrajs (const PredObj & *obj*, std::vector< SP_PredTraj > & *predTrajs*)`[virtual]`

Generate a set of trajectories

Parameters:

	obj	initial vehicle position and velocity
out	*predTrajs*	the set of generated trajectories

Implements **MotPredNameSpace::PredMethod** (*p.26*).

MotPredNameSpace::NormAngle Class Reference

```
#include <MotionPrediction.h>
```

Public Member Functions

- **NormAngle** (double norm, double angle)
- **NormAngle** (const **SSAMPoint::point** &p)
- **NormAngle operator+** (const **NormAngle** &rhs)
- **SSAMPoint::point getPoint** ()

Static Public Member Functions

- static **NormAngle fromPoint** (**SSAMPoint::point** p)

Public Attributes

- double **m_Norm**
- double **m_Angle**

Detailed Description

NormAngle represents a point in the form of norm and angle.

SSAMPoint::point Class Reference

```
#include <Utility.h>
```

Public Member Functions

- **point** (float ix=0.0, float iy=0.0, float iz=0.0)
- **point operator+** (const **point** &rhs) const
- **point operator+** (float addition) const
- **point** & **operator+=** (const **point** &rhs)
- **point operator-** (const **point** &rhs) const
- **point** & **operator-=** (const **point** &rhs)
- **point operator*** (float scale) const
- **point operator/** (float scale) const
- bool **operator==** (const **point** &rhs)
- bool **operator!=** (const **point** &rhs)
- float **norm** (void) const

Public Attributes

- float **x**
- float **y**
- float **z**

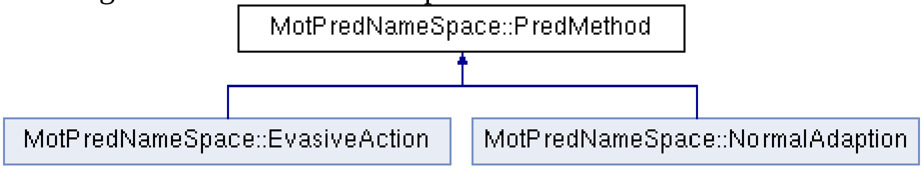
Detailed Description

Point class defines operators

MotPredNameSpace::PredMethod Class Reference

```
#include <MotionPrediction.h>
```

Inheritance diagram for MotPredNameSpace::PredMethod:


MotPredNameSpace::PredMethod
MotPredNameSpace::EvasiveAction
MotPredNameSpace::NormalAdaption

Public Types

- enum **METHOD_TYPE** { **NORMALADAPTION**, **EVASIVEACTION** }

Public Member Functions

- **PredMethod** (METHOD_TYPE mType, const std::string &name, int nPredTrajs, double maxSpeed, double maxSteering, double maxAccRate, double minAccRate)

Protected Member Functions

- virtual void **GenPredTrajs** (const **PredObj** &obj, std::vector< **SP_PredTraj** > &predTrajs)=0
- bool **DetectCollision** (**SP_PredTraj** pTrj1, **SP_PredTraj** pTrj2, float collisionThreshold, int nSteps, int &t, **SSAMPoint::point** &p1, **SSAMPoint::point** &p2)
- bool **DetectCrossingZone** (**SP_PredTraj** pTrj1, **SP_PredTraj** pTrj2, float collisionThreshold, int nSteps, double &pet)

Protected Attributes

- int **m_nPredTrajs**
- double **m_MaxSpeed**
- **TriangularDistri m_AccelDistri**
- **TriangularDistri m_SteerDistri**

Private Attributes

- METHOD_TYPE **m_Type**
- std::string **m_Name**

Detailed Description

PredMethod defines a motion prediction method.

Constructor & Destructor Documentation

MotPredNameSpace::PredMethod::PredMethod (METHOD_TYPE *mType*, const std::string & *name*, int *nPredTrajs*, double *maxSpeed*, double *maxSteering*, double *maxAccRate*, double *minAccRate*) `[inline]`

Constructor genarates a motion prediction method

Parameters:

mType	motion prediction method type: normal adaption or evasive action
name	method name
nPredTrajs	number of trajectories to predict
maxSpeed	the maximum speed allowed in the roadway network
maxSteering	max of triangular distribution for steering angle: radians/s

maxAccRate	max of triangular distribution for acceleration rate: ft/s^2
minAccRate	min of triangular distribution for acceleration rate: ft/s^2

Member Function Documentation

bool MotPredNameSpace::PredMethod::DetectCollision (SP_PredTraj *pTrj1*, SP_PredTraj *pTrj2*, float *collisionThreshold*, int *nSteps*, int & *t*, SSAMPoint::point & *p1*, SSAMPoint::point & *p2*) `[protected]`

Detect collision between two vehicles

Parameters:

	pTrj1	smart pointer to the trajectory of first vehicle
	pTrj2	smart pointer to the trajectory of second vehicle
	collisionThreshold	a distance threshold to determine whether two vehicles collide
	nSteps	number of steps to detect
out	*t*	the step when the collision is detected
out	*p1*	position of the first vehicle when the collision is detected
out	*p2*	position of the second vehicle when the collision is detected

Returns:

a flag to indicate whether a collision is detected

bool MotPredNameSpace::PredMethod::DetectCrossingZone (SP_PredTraj *pTrj1*, SP_PredTraj *pTrj2*, float *collisionThreshold*, int *nSteps*, double & *pet*) `[protected]`

Detect the crossing zone between two vehicles

Parameters:

	pTrj1	smart pointer to the trajectory of first vehicle
	pTrj2	smart pointer to the trajectory of second vehicle
	collisionThreshold	a distance threshold to determine whether two vehicles collide
	nSteps	number of steps to detect
out	*pet*	PET if crossing zone is detected

Returns:

a flag to indicate whether a crossing zone is detected

virtual void MotPredNameSpace::PredMethod::GenPredTrajs (const PredObj & *obj*, std::vector< SP_PredTraj > & *predTrajs*) `[protected]`, `[pure virtual]`

Generate a set of trajectories

Parameters:

	obj	initial vehicle position and velocity
out	*predTrajs*	the set of generated trajectories

Implemented in **MotPredNameSpace::EvasiveAction** (*p.13*), and **MotPredNameSpace::NormalAdaption** (*p.22*).

Member Data Documentation

TriangularDistri MotPredNameSpace::PredMethod::m_AccelDistri `[protected]`

a triangular distribution for generating acceleration rate

double MotPredNameSpace::PredMethod::m_MaxSpeed `[protected]`

maximum speed allowed in the roadway network

std::string MotPredNameSpace::PredMethod::m_Name `[private]`

name of the motion prediction method

int MotPredNameSpace::PredMethod::m_nPredTrajs `[protected]`

number of trajectories to predict

TriangularDistri MotPredNameSpace::PredMethod::m_SteerDistri `[protected]`

a triangular distribution for generating steering angle

METHOD_TYPE MotPredNameSpace::PredMethod::m_Type `[private]`

motion prediction method type: normal adaption or evasive action

MotPredNameSpace::PredObj Struct Reference

```
#include <MotionPrediction.h>
```

Public Attributes

- **SSAMPoint::point pos**
- **SSAMPoint::point vel**

Detailed Description

PredObj organizes vehicle position and velocity at one step.

Member Data Documentation

SSAMPoint::point MotPredNameSpace::PredObj::pos

Vehicle position

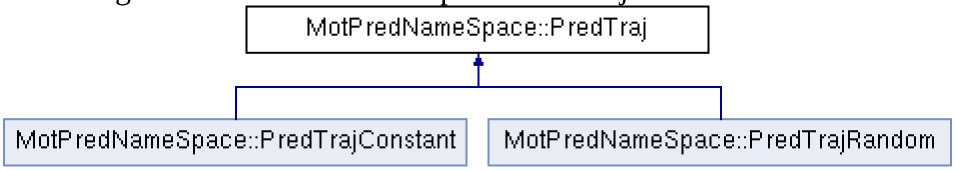
SSAMPoint::point MotPredNameSpace::PredObj::vel

Vehicle velocity

MotPredNameSpace::PredTraj Class Reference

`#include <MotionPrediction.h>`

Inheritance diagram for MotPredNameSpace::PredTraj:


MotPredNameSpace::PredTraj
MotPredNameSpace::PredTrajConstant
MotPredNameSpace::PredTrajRandom

Public Member Functions

- **PredTraj** (const **PredObj** &initObj, double maxSpeed=100)
- **SSAMPoint::point GetPos** (int nSteps)
- virtual **NormAngle GetControl** ()=0

Protected Member Functions

- void **GetNextPos** (const **SSAMPoint::point** &pos, **NormAngle** &spdOrien, **SSAMPoint::point** &nextPos, **NormAngle** &nextSpdOrien)

Protected Attributes

- std::vector< **SSAMPoint::point** > **m_PredPoses**
- std::vector< **NormAngle** > **m_PredSpdOriens**

Private Attributes

- double **m_MaxSpd**

Detailed Description

PredTraj represents one predicted vehicle trajectory

Constructor & Destructor Documentation

MotPredNameSpace::PredTraj::PredTraj (const PredObj & *initObj*, double *maxSpeed* = `100`) `[inline]`

Constructor generates a vehicle trajectory

Parameters:

initObj	initial position and speed orientation of vehicle
maxSpeed	the maximum speed allowed in the roadway network

Member Function Documentation

virtual NormAngle MotPredNameSpace::PredTraj::GetControl () `[pure virtual]`

Get a set of acceleration rate and steering angle

Implemented in **MotPredNameSpace::PredTrajConstant** (*p.31*), and **MotPredNameSpace::PredTrajRandom** (*p.34*).

void MotPredNameSpace::PredTraj::GetNextPos (const SSAMPoint::point & *pos*, NormAngle & *spdOrien*, SSAMPoint::point & *nextPos*, NormAngle & *nextSpdOrien*) `[protected]`

Calculate the next position

Parameters:

	pos	the current position.
	spdOrien	the current speed and orientation
out	*nextPos*	the next position
out	*nextSpdOrien*	the speed and orientation at next position

point MotPredNameSpace::PredTraj::GetPos (int *nSteps*)

Calculate position at a target step

Parameters:

nSteps	the target step

Returns:

the position at the target step

Member Data Documentation

double MotPredNameSpace::PredTraj::m_MaxSpd `[private]`

the maximum speed allowed in the roadway network

std::vector<SSAMPoint::point> MotPredNameSpace::PredTraj::m_PredPoses `[protected]`

the array of all predicted positions

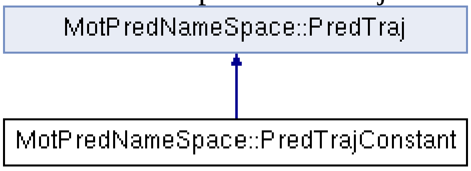
std::vector<NormAngle> MotPredNameSpace::PredTraj::m_PredSpdOriens `[protected]`

the array of all predicted speeds and orientations

MotPredNameSpace::PredTrajConstant Class Reference

`#include <MotionPrediction.h>`

Inheritance diagram for MotPredNameSpace::PredTrajConstant:


MotPredNameSpace::PredTraj
MotPredNameSpace::PredTrajConstant

Public Member Functions

- **PredTrajConstant** (const **PredObj** &initObj, double maxSpeed)
- void **SetControl** (const **NormAngle** &icontrol)
- virtual **NormAngle GetControl** ()

Private Attributes

- **NormAngle m_Control**

Additional Inherited Members

Detailed Description

PredTrajConstant represents one vehicle trajectory predicted with constant control at each step.

Constructor & Destructor Documentation

MotPredNameSpace::PredTrajConstant::PredTrajConstant (const PredObj & *initObj*, double *maxSpeed*) `[inline]`

Constructor genarates a constant vehicle trajectory

Parameters:

initObj	initial vehicle position and velocity
maxSpeed	the maximum speed allowed in the roadway network

Member Function Documentation

virtual NormAngle MotPredNameSpace::PredTrajConstant::GetControl () `[inline]`, `[virtual]`

Get the set of acceleration rate and steering angle

Implements **MotPredNameSpace::PredTraj** (*p.29*).

void MotPredNameSpace::PredTrajConstant::SetControl (const NormAngle & *icontrol*) `[inline]`

Set the constant vehicle trajectory

Parameters:

icontrol	the constant acceleration rate and steering angle

Member Data Documentation

NormAngle MotPredNameSpace::PredTrajConstant::m_Control `[private]`

the acceleration rate and steering angle

MotPredNameSpace::PredTrajFactory Class Reference

```
#include <MotionPrediction.h>
```

Public Member Functions

- **SP_PredTraj CreatePredTraj** (PredMethod::METHOD_TYPE mType, const **PredObj** &initObj, double maxSpeed)

Detailed Description

PredTrajFactory creates a smart pointer to **PredTrajRandom** or **PredTrajConstant** accroding to motion prediction method type.

Member Function Documentation

SP_PredTraj MotPredNameSpace::PredTrajFactory::CreatePredTraj (PredMethod::METHOD_TYPE *mType*, const PredObj & *initObj*, double *maxSpeed*)

Create smart pointer to **PredTraj**. The pointed object could be **PredTrajRandom** or **PredTrajConstant** accroding to motion prediction method type.

Parameters:

mType	motion prediction method type: normal adaption or evasive action
initObj	initial vehicle position and velocity
maxSpeed	the maximum speed allowed in the roadway network

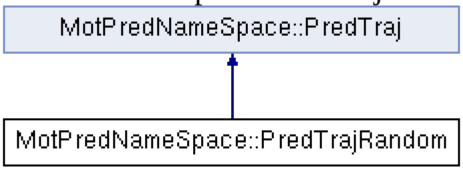
Returns:

a smart pointer to **PredTrajRandom** or **PredTrajConstant** accroding to motion prediction method type.

MotPredNameSpace::PredTrajRandom Class Reference

`#include <MotionPrediction.h>`

Inheritance diagram for MotPredNameSpace::PredTrajRandom:



Public Member Functions

- **PredTrajRandom** (const **PredObj** &initObj, double maxSpeed)
- void **SetDistributions** (const **TriangularDistri** &accelDistri, const **TriangularDistri** &steerDistri)
- virtual **NormAngle GetControl** ()

Private Attributes

- **TriangularDistri m_AccelDistri**
- **TriangularDistri m_SteerDistri**

Additional Inherited Members

Detailed Description

PredTrajRandom represents one vehicle trajectory predicted with random control at each step.

Constructor & Destructor Documentation

MotPredNameSpace::PredTrajRandom::PredTrajRandom (const PredObj & *initObj*, double *maxSpeed*) `[inline]`

Constructor genarates a random vehicle trajectory

Parameters:

initObj	initial vehicle position and velocity
maxSpeed	the maximum speed allowed in the roadway network

Member Function Documentation

virtual NormAngle MotPredNameSpace::PredTrajRandom::GetControl () `[inline]`, `[virtual]`

Get a set of acceleration rate and steering angle

Implements **MotPredNameSpace::PredTraj** (*p.29*).

void MotPredNameSpace::PredTrajRandom::SetDistributions (const TriangularDistri & *accelDistri*, const TriangularDistri & *steerDistri*) `[inline]`

Set triangular distributions for generating acceleration rate and steering angle at each step

Parameters:

accelDistri	a triangular distribution for generating acceleration rate
steerDistri	a triangular distribution for generating steering angle

Member Data Documentation

TriangularDistri MotPredNameSpace::PredTrajRandom::m_AccelDistri `[private]`

a triangular distribution for generating acceleration rate

TriangularDistri MotPredNameSpace::PredTrajRandom::m_SteerDistri `[private]`

a triangular distribution for generating steering angle

SSAMFuncs::SSAM Class Reference

`#include <SSAM.h>`

Public Member Functions

- SSAMFUNCSDLL_API void **Initialize** ()
- SSAMFUNCSDLL_API void **Terminate** ()
- SSAMFUNCSDLL_API void **CloseRun** ()
- SSAMFUNCSDLL_API void **Analyze** ()
- SSAMFUNCSDLL_API void **ExportResults** ()
- SSAMFUNCSDLL_API void **SetTrjSrcName** (const std::string &s)
- SSAMFUNCSDLL_API void **SetFormat** (const **InputFormat** &f)
- SSAMFUNCSDLL_API void **SetDimensions** (**SP_Dimensions** pDims)
- SSAMFUNCSDLL_API void **SetTimeStep** (float t)
- SSAMFUNCSDLL_API void **SetVehicle** (SP_Vehicle pVeh)
- SSAMFUNCSDLL_API void **CalcSummaries** ()
- void **SetMaxTTC** (float maxTTC)
- void **SetMaxPET** (float max)
- void **SetRearEndAngle** (int rna)
- void **SetCrossingAngle** (int ca)
- void **SetCSVFile** (const std::string &s)
- void **SetNThreads** (int n)
- void **SetIsCalcPUEA** (bool isCalcPUEA)
- void **SetPrintProgess** (bool b)
- void **SetWriteDat** (bool b)
- void **AddTrjFile** (const std::string &s)
- void **AddTrjDataList** (const std::string &s, std::list< **TrjRecord** > *trjDataList)
- float **GetMaxTTC** ()
- float **GetMaxPET** ()
- bool **GetIsCalcPUEA** ()
- int **GetRearEndAngle** ()
- int **GetCrossingAngle** ()
- int **GetUnits** ()
- std::string & **GetCSVFile** ()
- std::list< SP_Conflict > & **GetConflictList** ()
- std::list< SP_Summary > & **GetSummaries** ()
- SP_Summary **GetSummary** () const
- int **GetAnalysisTime** () const
- const std::list< std::string > & **GetTrjFileNames** () const

Public Attributes

- int **m_Boundary** [4]

Static Public Attributes

- static const float **DEFAULT_TTC** = 1.5
- static const float **DEFAULT_PET** = 5.0
- static const int **DEFAULT_REARENDANGLE** =30
- static const int **DEFAULT_CROSSINGANGLE** =80

Protected Attributes

- float **m_MaxTTC**
- float **m_MaxPET**
- int **m_RearEndAngleThreshold**

- int **m_CrossingAngleThreshold**
- int **m_AnalysisTime**
- int **m_StartTime**
- int **m_EndTime**
- std::list< std::string > **m_TrjFileNames**
- std::map< VehiclePair, SP_Event > **m_EventList**
- std::list< SP_Conflict > **m_ConflictList**
- std::map< std::string, std::list< SP_Conflict > > **m_FileToConflictsMap**
- SP_Summary **m_pSummary**
- std::list< SP_Summary > **m_Summaries**
- bool **m_IsCalcPUEA**
- std::string **m_CsvFileName**

Private Member Functions

- void **Analyze** (const std::list< std::string > &trjFileNames)
- void **Analyze** (const std::list< **TrjDataList** > &trjDataLists)
- void **ReadTrjInputSize** ()
- void **ReadInputFormat** ()
- void **ReadDimensions** ()
- void **ApplyDimensions** ()
- void **ReadVehicle** (SP_Vehicle pVeh)
- void **ValidateInputFormat** ()
- bool **ValidateVehicle** (SP_Vehicle pVeh)
- void **PrintTimeStep** (float t)
- void **ThrowFileReadingException** (**SSAMException** &e, std::string &errMsg)
- void **AnalyzeOneStep** ()
- void **DetectConflicts** (SP_ZoneGrid pZoneGrid, std::map< VehiclePair, SP_Event > &eventList)
- void **AnalyzEvents** (const std::string &trjSrcName, std::map< VehiclePair, SP_Event > &eventList)
- void **CreateConflict** (SP_Event e, const std::string &trjSrcName)
- char **ReadByte** (std::ifstream &in)
- float **ReadFloat** (std::ifstream &in)
- int **ReadInt** (std::ifstream &in)
- void **ConvertEndianness** (char *buffer, int size)

Private Attributes

- std::string **m_TrjSrcName**
- std::ifstream **m_TrjFile**
- std::list< **TrjDataList** > **m_TrjDataLists**
- std::list< **TrjRecord** > **m_TrjDataList**
- bool **m_IsWriteDat**
- std::ofstream **m_DatFile**
- int **m_NThreads**
- SP_ZoneGrid **m_pZoneGrid**
- float **m_Units**
- float **m_ZoneSize**
- float **m_ReadTimeStep**
- float **m_AnalysisTimeStep**
- **InputFormat m_Format**
- **SP_Dimensions m_pDimensions**
- std::list< **SP_TimeStepData** > **m_StepDataList**
- **SP_TimeStepData m_pCurStep**
- bool **m_IsFirstTimeStep**
- int **m_NSteps**
- **InitEventParams m_InitEventParams**
- bool **m_IsPrintProgress**

- char **m_BufferInt** [INT_SIZE]
- char **m_BufferFloat** [FLOAT_SIZE]

Static Private Attributes

- static const int **INT_SIZE** = sizeof(int)
- static const int **FLOAT_SIZE** = sizeof(float)

Detailed Description

SSAM reads TRJ input, runs **SSAM** simulation, and maintains conflict results

Member Function Documentation

void SSAM::Analyze ()

Run **SSAM** analysis.

void SSAM::Analyze (const std::list< std::string > & *trjFileNames*) `[private]`

Run **SSAM** analysis on a list of TRJ files.

void SSAM::Analyze (const std::list< TrjDataList > & *trjDataLists*) `[private]`

Run **SSAM** analysis on a list of TRJ data lists.

void SSAM::AnalyzeOneStep () `[private]`

Run one step of **SSAM** analysis.

void SSAM::AnalyzEvents (const std::string & *trjSrcName*, std::map< VehiclePair, SP_Event > & *eventList*) `[private]`

Analyze conflicts detected in the current step.

Parameters:

trjSrcName	name of TRJ source
eventList	a map container stores smart pointers to conflict events using vehicle pairs as keys

void SSAM::ApplyDimensions () `[private]`

Use dimensions to set grid zone, boundary coordinates and motion prediction parameters.

void SSAM::CalcSummaries ()

Calculate summary on safety measures of all conflicts

void SSAM::CloseRun ()

Finish current **SSAM** run.

void SSAMFuncs::SSAM::ConvertEndianness (char * *buffer*, int *size*) `[inline]`, `[private]`

Convert the endianness from Big Endian to Little Endian.

Parameters:

buffer	a char array stores the value
size	size of the buffer

void SSAMFuncs::SSAM::CreateConflict (SP_Event *e*, const std::string & *trjSrcName*) `[inline]`, `[private]`

Create a conflict record.

Parameters:

e	smart pointer to the conflict event for creating conflict record
trjSrcName	name of TRJ source

void SSAM::DetectConflicts (SP_ZoneGrid *pZoneGrid*, std::map< VehiclePair, SP_Event > & *eventList*) `[private]`

Detect conflicts in the current step of vehicles.

Parameters:

pZoneGrid	smart pointer to the zone grid
eventList	a map container stores conflict events using vehicle pairs as keys

void SSAM::ExportResults ()

Export conflict points and summary to the csv file.

void SSAM::Initialize ()

Initialize **SSAM** analysis parameters.

void SSAM::PrintTimeStep (float *t*) `[private]`

Print a read time step to csv file.

char SSAMFuncs::SSAM::ReadByte (std::ifstream & *in*) `[inline]`, `[private]`

Read a byte from binaray file.

Parameters:

in	the input stream

Returns:

byte value

void SSAM::ReadDimensions () `[private]`

Read dimensions of analysis area from TRJ file.

float SSAMFuncs::SSAM::ReadFloat (std::ifstream & *in*) `[inline]`, `[private]`

Read a float number from binaray file.

Parameters:

in	the input stream

Returns:

float value

void SSAM::ReadInputFormat () `[private]`

Read TRJ format from TRJ file.

int SSAMFuncs::SSAM::ReadInt (std::ifstream & *in*) `[inline]`, `[private]`

Read an integer number from binaray file.

Parameters:

in	the input stream

Returns:

int value

void SSAM::ReadTrjInputSize () `[private]`

Read the size of TRJ source: bytes for TRJ file, and records for TRJ data list.

void SSAM::ReadVehicle (SP_Vehicle *pVeh*) `[private]`

Read a vehicle from TRJ file.

Parameters:

pVeh	smart pointer to a vehicle as output value

void SSAM::SetDimensions (SP_Dimensions *pDims*)

Set the dimensions of analysis area.

Parameters:

pDims	smart pointer to the dimensions of analysis area

void SSAM::SetFormat (const InputFormat & *f*)

Set the TRJ format.

Parameters:

f	TRJ format

void SSAM::SetTimeStep (float *t*)

Set time step value and run one step of **SSAM** analysis.

Parameters:

t	time step

void SSAM::SetTrjSrcName (const std::string & *s*)

Set the name of TRJ data source.

Parameters:

s	name of TRJ data source

void SSAM::SetVehicle (SP_Vehicle *pVeh*)

Add a vehicle to the current time step.

Parameters:

pVeh	smart pointer to a vehicle as input value

void SSAM::Terminate ()

Wrap up **SSAM** analysis: calculate summaries and analysis time.

void SSAM::ThrowFileReadingException (SSAMException & *e*, std::string & *errMsg*) `[private]`

Throw a file reading exception.

Parameters:

e	a **SSAM** execption
errMsg	basic error messages

void SSAM::ValidateInputFormat () `[private]`

Validate current TRJ format.

bool SSAM::ValidateVehicle (SP_Vehicle *pVeh*) `[private]`

Validate a vehicle.

Parameters:

pVeh	smart pointer to a vehicle to validate

Returns:

true if intput is valid vehicle data.

Member Data Documentation

const int SSAMFuncs::SSAM::DEFAULT_CROSSINGANGLE =80 `[static]`

default crossing angle threshold

const float SSAM::DEFAULT_PET = 5.0 `[static]`

default maximum PET

const int SSAMFuncs::SSAM::DEFAULT_REARENDANGLE =30 `[static]`

default rear end angle threshold

const float SSAM::DEFAULT_TTC = 1.5 `[static]`

default maximum TTC

int SSAMFuncs::SSAM::m_AnalysisTime `[protected]`

Total time for analysis

float SSAMFuncs::SSAM::m_AnalysisTimeStep `[private]`

Current time step to run **SSAM** analysis

int SSAMFuncs::SSAM::m_Boundary[4]

Boundary coordinates of the observation area: 0: minX; 1: minY; 2: maxX; 3: maxY

std::list<SP_Conflict> SSAMFuncs::SSAM::m_ConflictList `[protected]`

List of smart pointers to conflict points A map container stores conflict smart pointers using TRJ source names as keys

int SSAMFuncs::SSAM::m_CrossingAngleThreshold `[protected]`

Crossing Angle Threshold

std::string SSAMFuncs::SSAM::m_CsvFileName `[protected]`

A csv file to output analysis results

std::ofstream SSAMFuncs::SSAM::m_DatFile `[private]`

A csv file to write input TRJ records

int SSAMFuncs::SSAM::m_EndTime `[protected]`

End time for analysis

std::map<VehiclePair, SP_Event> SSAMFuncs::SSAM::m_EventList `[protected]`

List of detected conflict events

InputFormat SSAMFuncs::SSAM::m_Format `[private]`

TRJ format

InitEventParams SSAMFuncs::SSAM::m_InitEventParams `[private]`

Parameters to initialize a conflict event

bool SSAMFuncs::SSAM::m_IsCalcPUEA `[protected]`

Flag to indicate whether to calculate P(UEA), mTTC, mPET

bool SSAMFuncs::SSAM::m_IsFirstTimeStep `[private]`

Flag to indicate whether the current read time step is the first time step

bool SSAMFuncs::SSAM::m_IsPrintProgress `[private]`

Flag to indicate whether to print the **SSAM** analysis progress in time step

bool SSAMFuncs::SSAM::m_IsWriteDat `[private]`

Flag indicates whether to write input TRJ records to a csv file

float SSAMFuncs::SSAM::m_MaxPET `[protected]`

Max PET threshold

float SSAMFuncs::SSAM::m_MaxTTC `[protected]`

Max TTC threshold

int SSAMFuncs::SSAM::m_NSteps `[private]`

Number of steps per second for motion prediction analysis

int SSAMFuncs::SSAM::m_NThreads `[private]`

Number of threads to use

SP_TimeStepData SSAMFuncs::SSAM::m_pCurStep `[private]`

Current time step data to run **SSAM** analysis

SP_Dimensions SSAMFuncs::SSAM::m_pDimensions `[private]`

Smart pointer to the dimensions of analysis area

SP_Summary SSAMFuncs::SSAM::m_pSummary `[protected]`

Smart pointer to the summary over all TRJ inputs

SP_ZoneGrid SSAMFuncs::SSAM::m_pZoneGrid `[private]`

Smart pointer to the zone grid object

float SSAMFuncs::SSAM::m_ReadTimeStep `[private]`

Current time step read from TRJ source

int SSAMFuncs::SSAM::m_RearEndAngleThreshold `[protected]`

Rear-End Angle Threshold

int SSAMFuncs::SSAM::m_StartTime `[protected]`

Start time for analysis

std::list<SP_TimeStepData> SSAMFuncs::SSAM::m_StepDataList `[private]`

Current list of time step data to run **SSAM** analysis

std::list<SP_Summary> SSAMFuncs::SSAM::m_Summaries `[protected]`

A list of summary smart pointers, each for one TRJ source

std::list<TrjRecord> SSAMFuncs::SSAM::m_TrjDataList `[private]`

A TRJ data lists to analyze

std::list<TrjDataList> SSAMFuncs::SSAM::m_TrjDataLists `[private]`

A list of TRJ data lists to analyze

std::ifstream SSAMFuncs::SSAM::m_TrjFile `[private]`

A TRJ file to analyze

std::list<std::string> SSAMFuncs::SSAM::m_TrjFileNames `[protected]`

A list of TRJ files to analyze

std::string SSAMFuncs::SSAM::m_TrjSrcName `[private]`

Name of TRJ data source

float SSAMFuncs::SSAM::m_Units `[private]`

Engligh or Metric units

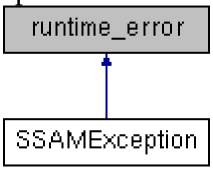
float SSAMFuncs::SSAM::m_ZoneSize `[private]`

Size of one zone

SSAMException Class Reference

```
#include <INCLUDE.h>
```

Inheritance diagram for SSAMException:



Public Member Functions

- **SSAMException** (const std::string &s)

Detailed Description

SSAMException is thrown when any error occurs in SSAM analysis

---

Constructor & Destructor Documentation

SSAMException::SSAMException (const std::string & *s*)`[inline]`, `[explicit]`

Create a **SSAMException**

Parameters:

s	Error message from std::runtime_error.

Summary Class Reference

```
#include <Summary.h>
```

Public Member Functions

- **Summary** (const std::string &trjFile, const std::list< SP_Conflict > &conflictList)
- const std::string & **GetTrjFile** () const
- const std::vector< float > & **GetMinVals** () const
- const std::vector< float > & **GetMaxVals** () const
- const std::vector< float > & **GetMeanVals** () const
- const std::vector< float > & **GetVarVals** () const
- const std::vector< int > & **GetConflictTypeCounts** () const

Static Public Attributes

- static const int **NUM_SUMMARY_LABELS** = 6
- static const std::string **SUMMARY_LABEL** [**NUM_SUMMARY_LABELS**]

Private Attributes

- std::string **m_TrjFile**
- std::vector< float > **m_MinVals**
- std::vector< float > **m_MaxVals**
- std::vector< float > **m_MeanVals**
- std::vector< float > **m_VarVals**
- std::vector< int > **m_ConflictCounts**

Detailed Description

Summary manages the summaries of safety measures for confirmed conflict in one TRJ input source.

Member Data Documentation

std::vector<int> Summary::m_ConflictCounts `[private]`

The numbers of conflicts of all conflict types. The last element is the total number of conflicts

std::vector<float> Summary::m_MaxVals `[private]`

The maximum values of summarized safety measures

std::vector<float> Summary::m_MeanVals `[private]`

The mean values of summarized safety measures

std::vector<float> Summary::m_MinVals `[private]`

The minimum values of summarized safety measures

std::string Summary::m_TrjFile `[private]`

The name of the trajectory file where the conflicts are identified

std::vector<float> Summary::m_VarVals `[private]`

The variance values of summarized safety measures

const int Summary::NUM_SUMMARY_LABELS = 6 `[static]`

The number of summary labels

const std::string Summary::SUMMARY_LABEL `[static]`

```
Initial value:=
{
   "Summary Group",
   "SSAM_Measure",
   "Min",
   "Max",
   "Mean",
   "Variance"
}
```

Strings represent summary labels

SSAMFuncs::TimeStepData Class Reference

```
#include <SSAM.h>
```

Public Member Functions

- void **SetTimestep** (float t)
- float **GetTimestep** () const
- std::map< int, SP_Vehicle > * **GetVehicleMap** ()
- std::vector< SP_Vehicle > * **GetVehicleVec** ()
- void **AddVehicle** (int vID, SP_Vehicle v)

Private Attributes

- float **m_TimeStep**
- std::map< int, SP_Vehicle > **m_VehicleMap**
- std::vector< SP_Vehicle > **m_VehicleVec**
- std::pair< std::map< int, SP_Vehicle >::iterator, bool > **m_MapReturn**

Detailed Description

TimeStepData manages vehicle data in one time step

Member Function Documentation

void SSAMFuncs::TimeStepData::AddVehicle (int *vID*, SP_Vehicle *v*)`[inline]`

Add one vehicle data of current time step

Parameters:

vID	ID of the vehicle to add
v	smart pointer to the vehicle data

Member Data Documentation

std::pair<std::map<int, SP_Vehicle>::iterator, bool> SSAMFuncs::TimeStepData::m_MapReturn `[private]`

A pair stores the return value of inserting new vehicle into map

float SSAMFuncs::TimeStepData::m_TimeStep `[private]`

Seconds since the start of the simulation

std::map<int, SP_Vehicle> SSAMFuncs::TimeStepData::m_VehicleMap `[private]`

A map container stores vehicle smart pointers using vehicle IDs as keys

std::vector<SP_Vehicle> SSAMFuncs::TimeStepData::m_VehicleVec `[private]`

A vector container stores vehicle smart pointers

MotPredNameSpace::TriangularDistri Class Reference

```
#include <MotionPrediction.h>
```

Public Member Functions

- **TriangularDistri** (double low, double high, double mode)
- double **operator()** ()

Private Attributes

- double **m_Low**
- double **m_High**
- double **m_Mode**

Detailed Description

TriangularDistri generates trigngular distribution values

SSAMFuncs::TrjRecord Class Reference

```
#include <SSAM.h>
```

Public Types

- enum **RECORD_TYPE** { **INVALID** = -1, **FORMAT**, **DIMENSIONS**, **TIMESTEP**, **VEHICLE** }

Public Member Functions

- **TrjRecord** (const **TrjRecord** &rhs)
- **TrjRecord** & **operator=** (const **TrjRecord** &rhs)
- void **SetRecordType** (**RECORD_TYPE** typeTag)
- void **SetFormat** (const **InputFormat** &f)
- void **SetDimensions** (**SP_Dimensions** d)
- void **SetTimestep** (float t)
- void **SetVehicle** (SP_Vehicle v)
- int **GetRecordType** () const
- const **InputFormat** & **GetFormat** () const
- **SP_Dimensions** **GetDimensions** () const
- float **GetTimestep** () const
- SP_Vehicle **GetVehicle** () const

Private Member Functions

- void **CopyValues** (const **TrjRecord** &rhs)

Private Attributes

- **RECORD_TYPE** **m_RecordType**
- **InputFormat** **m_Format**
- **SP_Dimensions** **m_pDimensions**
- float **m_TimeStep**
- SP_Vehicle **m_pVehicle**

Detailed Description

TrjRecord contains TRJ records from TRJ file or TRJ data set

Member Enumeration Documentation

enum SSAMFuncs::TrjRecord::RECORD_TYPE

RECORD_TYPE enumerates the record types in TRJ input

Enumerator:

INVALID	Invalid record type in TRJ input
FORMAT	Format record in TRJ input
DIMENSIONS	Dimenstions record in TRJ input
TIMESTEP	Time step record in TRJ input

VEHICLE	**Vehicle** record in TRJ input

Constructor & Destructor Documentation

SSAMFuncs::TrjRecord::TrjRecord (const TrjRecord & *rhs*) `[inline]`

Copy contructor from another **TrjRecord** object

Parameters:

rhs	Const reference to a **TrjRecord** object from which the values are copied

Member Function Documentation

void SSAMFuncs::TrjRecord::CopyValues (const TrjRecord & *rhs*) `[inline]`, `[private]`

Copy values from another **TrjRecord** object

Parameters:

rhs	Const reference to a **TrjRecord** object from which the values are copied

TrjRecord& SSAMFuncs::TrjRecord::operator= (const TrjRecord & *rhs*) `[inline]`

Overloaded assignment operator

Parameters:

rhs	Const reference to a **TrjRecord** object from which the values are copied

Member Data Documentation

InputFormat SSAMFuncs::TrjRecord::m_Format `[private]`

Byte format and file format

SP_Dimensions SSAMFuncs::TrjRecord::m_pDimensions `[private]`

Smart pointer to dimensions of analysis region

SP_Vehicle SSAMFuncs::TrjRecord::m_pVehicle `[private]`

Smart pointer to data of a vehicle in one time step

RECORD_TYPE SSAMFuncs::TrjRecord::m_RecordType `[private]`

TRJ record type

float SSAMFuncs::TrjRecord::m_TimeStep `[private]`

Seconds since the start of the simulation

Vehicle Class Reference

```
#include <Vehicle.h>
```

Public Types

- enum **VEHICLE_CORNER** { **FRONT_LEFT**, **FRONT_RIGHT**, **REAR_RIGHT**, **REAR_LEFT** }

Public Member Functions

- **Vehicle** (const **Vehicle** &rhs)
- **Vehicle** & **operator=** (const **Vehicle** &rhs)
- SSAMFUNCSDLL_API void **SetPosition** (float frontX, float frontY, float rearX, float rearY)
- SP_Vehicle **CalcProjection** (float maxTTC, float maxPET)
- bool **IsCollided** (SP_Vehicle v)
- void **Print** (std::ostream &output, float version)
- void **SetTimeStep** (float t)
- void **SetVehicleID** (int i)
- void **SetLinkID** (int i)
- void **SetLaneID** (char b)
- void **SetFrontX** (float i)
- void **SetFrontY** (float i)
- void **SetFrontZ** (float i)
- void **SetRearX** (float i)
- void **SetRearY** (float i)
- void **SetRearZ** (float i)
- void **setScale** (float f)
- void **SetSpeed** (float f)
- void **SetAcceleration** (float f)
- void **SetNext** (SP_Vehicle v)
- void **SetLength** (float f)
- void **SetWidth** (float f)
- float **GetTimeStep** () const
- int **GetVehicleID** () const
- int **GetLinkID** () const
- int **GetLaneID** () const
- float **GetFrontX** () const
- float **GetFrontY** () const
- float **GetFrontZ** () const
- float **GetRearX** () const
- float **GetRearY** () const
- float **GetRearZ** () const
- float **GetLength** () const
- float **GetWidth** () const
- float **GetSpeed** () const
- float **GetAcceleration** () const
- SP_Vehicle **GetNext** () const
- float **GetCenterX** ()
- float **GetCenterY** ()
- float **GetCenterZ** ()
- const float * **GetCornerXs** () const
- const float * **GetCornerYs** () const
- float **GetMaxX** () const
- float **GetMaxY** () const
- float **GetMinX** () const

- float **GetMinY** () const

Private Member Functions

- void **CopyValues** (const **Vehicle** &rhs)
- float **GetV2VDistance** (**Vehicle** &v)
- void **CalcPerpOffset** (float x1, float y1, float x2, float y2, float dist, float &dx, float &dy)

Private Attributes

- float **m_TimeStep**
- int **m_VehicleID**
- int **m_LinkID**
- char **m_LaneID**
- float **m_Length**
- float **m_Width**
- float **m_Speed**
- float **m_Acceleration**
- float **m_FrontX**
- float **m_FrontY**
- float **m_FrontZ**
- float **m_RearX**
- float **m_RearY**
- float **m_RearZ**
- float **m_Scale**
- float **m_ScaledLength**
- float **m_ScaledWidth**
- float **m_CornerX** [4]
- float **m_CornerY** [4]
- float **m_MinX**
- float **m_MinY**
- float **m_MaxX**
- float **m_MaxY**
- SP_Vehicle **m_pNext**

Detailed Description

Vehicle manages data of a vehicle in one time step

Member Enumeration Documentation

enum Vehicle::VEHICLE_CORNER

VEHICLE_CORNER enumerates the four corners of vehicle shape

Constructor & Destructor Documentation

Vehicle::Vehicle (const Vehicle & *rhs*)`[inline]`

Copy contructor from another **Vehicle** object

Parameters:

rhs	Const reference to a **Vehicle** object from which the values are copied

Member Function Documentation

void Vehicle::CalcPerpOffset (float *x1*, float *y1*, float *x2*, float *y2*, float *dist*, float & *dx*, float & *dy*)[private]

Calculate the offset for a point perpendicular to the line between point 1 and and point 2 with a specified distance from the line, returned point is on the right side of direction P2->P1

Parameters:

x1	x value of starting point p1
y1	y value of starting point p1
x2	x value of ending point p2
y2	y value of ending point p2
dist	Distance of the perpendicular offset
dx	output x offset
dy	output y offset

SP_Vehicle Vehicle::CalcProjection (float *maxTTC*, float *maxPET*)

Calculate vehicle projection position in maxTTC.

Parameters:

maxTTC	maxTTC threshold
maxPET	maxPET threshold

void Vehicle::CopyValues (const Vehicle & *rhs*)[inline], [private]

Copy values from another **Vehicle** object

Parameters:

rhs	Const reference to a **Vehicle** object from which the values are copied

float Vehicle::GetV2VDistance (Vehicle & *v*)[inline], [private]

Get the distance between the input vehicle and this vehicle

Parameters:

v	A vehicle for calculating the distance

bool Vehicle::IsCollided (SP_Vehicle *v*)

Check whether the input vehicle intersects with this vehicle.

Parameters:

v	A vehicle for collision check

Vehicle& Vehicle::operator= (const Vehicle & *rhs*)[inline]

Overloaded assignment operator

Parameters:

rhs	Const reference to a **Vehicle** object from which the values are copied

void Vehicle::Print (std::ostream & *output*, float *version*)

Print current vehicle info.

Parameters:

output	the output stream
version	the version of TRJ format

void Vehicle::SetPosition (float *frontX*, float *frontY*, float *rearX*, float *rearY*)

Set vehicle position from coordinates of front and rear bumpers

Parameters:

frontX	X coordinate of the middle front bumper of the vehicle
frontY	Y coordinate of the middle front bumper of the vehicle
rearX	X coordinate of the middle rear bumper of the vehicle
rearY	Y coordinate of the middle rear bumper of the vehicle

Member Data Documentation

float Vehicle::m_Acceleration `[private]`

Instantaneous forward acceleration (Units/sec2)

float Vehicle::m_CornerX[4] `[private]`

X coordinates of the corners of the vehicle

float Vehicle::m_CornerY[4] `[private]`

Y coordinates of the corners of the vehicle

float Vehicle::m_FrontX `[private]`

X coordinate of the middle front bumper of the vehicle

float Vehicle::m_FrontY `[private]`

Y coordinate of the middle front bumper of the vehicle

float Vehicle::m_FrontZ `[private]`

Z coordinate of the middle front bumper of the vehicle

char Vehicle::m_LaneID `[private]`

Unique identifier number of the lane

float Vehicle::m_Length `[private]`

Vehicle length (front to back) in Units (feet or meters)

int Vehicle::m_LinkID `[private]`

Unique identifier number of the link

float Vehicle::m_MaxX `[private]`

Right edge of the vehicle occupying area.

float Vehicle::m_MaxY `[private]`

Top edge of the vehicle occupying area.

float Vehicle::m_MinX `[private]`

Left edge of the vehicle occupying area.

float Vehicle::m_MinY `[private]`

Bottom edge of the vehicle occupying area.

SP_Vehicle Vehicle::m_pNext `[private]`

Pointer to vehicle data of next time step

float Vehicle::m_RearX `[private]`

X coordinate of the middle rear bumper of the vehicle

float Vehicle::m_RearY `[private]`

Y coordinate of the middle rear bumper of the vehicle

float Vehicle::m_RearZ `[private]`

Z coordinate of the middle rear bumper of the vehicle

float Vehicle::m_Scale `[private]`

Distance per unit of X or Y

float Vehicle::m_ScaledLength `[private]`

Vehicle length (front to back) in X, Y Units

float Vehicle::m_ScaledWidth `[private]`

Vehicle width (left to right) in X, Y Units

float Vehicle::m_Speed `[private]`

Instantaneous forward speed (Units/sec)

float Vehicle::m_TimeStep `[private]`

Seconds since the start of the simulation

int Vehicle::m_VehicleID `[private]`

Unique identifier number of the vehicle

float Vehicle::m_Width `[private]`

Vehicle width (left to right) in Units (feet or meters)

ZoneGrid::Zone Class Reference

Public Member Functions

- void **AddVehicle** (SP_Vehicle vNew, std::map< int, SP_Vehicle > &allCrashes)
- void **Clear** ()

Private Attributes

- std::list< SP_Vehicle > **m_Occupants**

Detailed Description

Zone maintains a list of occupying vehicles.

Member Function Documentation

void ZoneGrid::Zone::AddVehicle (SP_Vehicle *vNew*, std::map< int, SP_Vehicle > & *allCrashes*)

Add a new occupying vehicle to the zone and checks whether it crashes with other vehicles in the zone.

Parameters:

vNew	A pointer to a new vehicle.
allCrashes	A list of vehicles crashing with new vehicle.

void ZoneGrid::Zone::Clear () `[inline]`

Remove all vehicles from this zone.

Member Data Documentation

std::list<SP_Vehicle> ZoneGrid::Zone::m_Occupants `[private]`

The list of vehicles occupying this zone

ZoneGrid Class Reference

```
#include <ZoneGrid.h>
```

Classes

- class **Zone**

Public Member Functions

- **ZoneGrid** (int xMin, int yMin, int xMax, int yMax, int size)
- void **ResetGrid** (int xMin, int yMin, int xMax, int yMax, int size)
- void **ClearGrid** ()
- void **AddVehicle** (SP_Vehicle vNew, std::map< int, SP_Vehicle > &allCrashes)

Private Types

- typedef std::shared_ptr< **Zone** > **SP_Zone**
- typedef std::pair< int, int > **UsedZone**

Private Attributes

- int **m_OrigMinX**
- int **m_OrigMinY**
- int **m_OrigMaxX**
- int **m_OrigMaxY**
- int **m_MinX**
- int **m_MinY**
- int **m_MaxX**
- int **m_MaxY**
- int **m_ZoneSize**
- int **m_NXZones**
- int **m_NYZones**
- int **m_NZones**
- int **m_NZonesUsed**
- std::vector< std::vector< **SP_Zone** > > **m_Zones**
- std::vector< UsedZone > **m_UsedZones**

Detailed Description

ZoneGrid is constructed to conver the entire rectangular analysis area using the width and height from TRJ file.

Member Typedef Documentation

typedef std::shared_ptr<Zone> ZoneGrid::SP_Zone `[private]`

Smart pointer type to **Zone** class.

Constructor & Destructor Documentation

ZoneGrid::ZoneGrid (int *xMin*, int *yMin*, int *xMax*, int *yMax*, int *size*)

Constructor creates the **ZoneGrid** from parameters

Parameters:

xMin	Left edge of grid
yMin	Bottom edge of grid
xMax	Right edge of grid
yMax	Top edge of grid
size	Dimension of a single zone in pixels

Member Function Documentation

void ZoneGrid::AddVehicle (SP_Vehicle *vNew*, std::map< int, SP_Vehicle > & *allCrashes*)

Add the new vehicle to the zone grid and check whether there is any other vehicle it crashes with.

Parameters:

vNew	New vehicle to add
allCrashes	A set of all other vehicles the new vehicle crashes with

void ZoneGrid::ClearGrid () `[inline]`

Remove occupants in used zones.

void ZoneGrid::ResetGrid (int *xMin*, int *yMin*, int *xMax*, int *yMax*, int *size*)

Reset the **ZoneGrid** with parameters

Parameters:

xMin	Left edge of grid
yMin	Bottom edge of grid
xMax	Right edge of grid
yMax	Top edge of grid
size	Dimension of a single zone in pixels

Member Data Documentation

int ZoneGrid::m_MaxX `[private]`

Bottom edge of calculated grid

int ZoneGrid::m_MaxY `[private]`

Bottom edge of calculated grid

int ZoneGrid::m_MinX `[private]`

Left edge of calculated grid

int ZoneGrid::m_MinY `[private]`

Bottom edge of calculated grid

int ZoneGrid::m_NXZones `[private]`

Number of zones along x-axis

int ZoneGrid::m_NYZones `[private]`

Number of zones along y-axis

int ZoneGrid::m_NZones `[private]`

Total number of zones

int ZoneGrid::m_NZonesUsed `[private]`

Total number of used zones

int ZoneGrid::m_OrigMaxX `[private]`

Right edge of original grid

int ZoneGrid::m_OrigMaxY `[private]`

Top edge of original grid

int ZoneGrid::m_OrigMinX `[private]`

Left edge of original grid

int ZoneGrid::m_OrigMinY `[private]`

Bottom edge of original grid

std::vector<UsedZone> ZoneGrid::m_UsedZones `[private]`

A vector stores all used zones.

std::vector<std::vector<SP_Zone> > ZoneGrid::m_Zones `[private]`

A matrix of zones

int ZoneGrid::m_ZoneSize `[private]`

Dimension of a single zone

Index

AddVehicle
SSAMFuncs::TimeStepData, 47
ZoneGrid, 58
ZoneGrid::Zone, 56
AddVehicleData
Event, 16
Analyze
SSAMFuncs::SSAM, 38
AnalyzeData
Event, 16
AnalyzeOneStep
SSAMFuncs::SSAM, 38
AnalyzEvents
SSAMFuncs::SSAM, 38
ApplyDimensions
SSAMFuncs::SSAM, 38
CalcMeasures
Event, 16
CalcMTTCMPET
MotPredNameSpace::NormalAdaption, 21
CalcPerpOffset
Vehicle, 53
CalcProjection
Vehicle, 53
CalcPUEA
MotPredNameSpace::EvasiveAction, 12
CalcSummaries
SSAMFuncs::SSAM, 38
Clear
ZoneGrid::Zone, 56
ClearGrid
ZoneGrid, 58
CloseRun
SSAMFuncs::SSAM, 38
Conflict, 7
Conflict, 8
CONFLICT_TYPE, 8
CONFLICT_TYPE_LABEL, 9
GetMeasure, 9
GetStrippedTrjName, 9
GetValueString, 9
KEY_MEASURES, 9
MEASURE_LABEL, 9
NUM_CONFLICT_TYPES, 9
NUM_MEASURES, 9
SSAM_MEASURE, 8
SUM_MEASURES, 9
CONFLICT_TYPE
Conflict, 8
CONFLICT_TYPE_LABEL
Conflict, 9
ConvertEndianness
SSAMFuncs::SSAM, 38
CopyValues
SSAMFuncs::TrjRecord, 50
Vehicle, 53
CreateConflict
SSAMFuncs::SSAM, 39
CreatePredTraj
MotPredNameSpace::PredTrajFactory, 33
DEFAULT_CROSSINGANGLE
SSAMFuncs::SSAM, 41
DEFAULT_PET
SSAMFuncs::SSAM, 41
DEFAULT_REARENDANGLE
SSAMFuncs::SSAM, 41
DEFAULT_TTC
SSAMFuncs::SSAM, 41
DetectCollision
MotPredNameSpace::PredMethod, 26
DetectConflicts
SSAMFuncs::SSAM, 39
DetectCrossingZone
MotPredNameSpace::PredMethod, 26
DIMENSIONS
SSAMFuncs::TrjRecord, 49
ENGLISH_UNITS
SSAMFuncs::Dimensions, 11
EvasiveAction
MotPredNameSpace::EvasiveAction, 12
Event, 14
AddVehicleData, 16
AnalyzeData, 16
CalcMeasures, 16
Event, 16
m_CollisionThreshold, 17
m_CrossingAngle, 17
m_FirstPET, 17
m_FirstTTC, 17
m_HighVData, 17
m_HighVID, 17
m_IsActive, 17
m_IsCalculatePUEA, 17
m_IsConflict, 17
m_IsPETComplete, 17
m_LastPET, 17
m_LastPETIdx, 17
m_LastTTC, 17
m_LastTTCIdx, 17
m_LowVData, 17
m_LowVID, 17
m_MaxPET, 18
m_MaxTTC, 18
m_NSteps, 18
m_PreTimeStep, 18
m_RearEndAngle, 18
m_StepSize, 18
m_TotalSteps, 18
ExportResults
SSAMFuncs::SSAM, 39
FORMAT
SSAMFuncs::TrjRecord, 49

GenPredTrajs
MotPredNameSpace::EvasiveAction, 13
MotPredNameSpace::NormalAdaption, 22
MotPredNameSpace::PredMethod, 26
GetControl
MotPredNameSpace::PredTraj, 29
MotPredNameSpace::PredTrajConstant, 31
MotPredNameSpace::PredTrajRandom, 34
GetMeasure
Conflict, 9
GetNextPos
MotPredNameSpace::PredTraj, 30
GetPos
MotPredNameSpace::PredTraj, 30
GetStrippedTrjName
Conflict, 9
GetV2VDistance
Vehicle, 53
GetValueString
Conflict, 9
InitEventParams, 19
m_CollisionThreshold, 19
m_CrossingAngleThreshold, 19
m_IsCalcPUEA, 19
m_MaxPET, 19
m_MaxTTC, 19
m_RearEndAngleThreshold, 19
m_V1, 19
m_V2, 19
Initialize
SSAMFuncs::SSAM, 39
INVALID
SSAMFuncs::TrjRecord, 49
IsCollided
Vehicle, 53
KEY_MEASURES
Conflict, 9
m_AccelDistri
MotPredNameSpace::PredMethod, 26
MotPredNameSpace::PredTrajRandom, 35
m_Acceleration
Vehicle, 54
m_AnalysisTime
SSAMFuncs::SSAM, 41
m_AnalysisTimeStep
SSAMFuncs::SSAM, 41
m_Boundary
SSAMFuncs::SSAM, 41
m_CollisionThreshold
Event, 17
InitEventParams, 19
m_ConflictCounts
Summary, 45
m_ConflictList
SSAMFuncs::SSAM, 41
m_Control
MotPredNameSpace::PredTrajConstant, 32
m_CornerX
Vehicle, 54
m_CornerY
Vehicle, 54
m_CrossingAngle
Event, 17
m_CrossingAngleThreshold
InitEventParams, 19
SSAMFuncs::SSAM, 41
m_CsvFileName
SSAMFuncs::SSAM, 41
m_DatFile
SSAMFuncs::SSAM, 41
m_Endian
SSAMFuncs::InputFormat, 20
m_EndTime
SSAMFuncs::SSAM, 41
m_EventList
SSAMFuncs::SSAM, 41
m_FirstPET
Event, 17
m_FirstTTC
Event, 17
m_Format
SSAMFuncs::SSAM, 41
SSAMFuncs::TrjRecord, 50
m_FrontX
Vehicle, 54
m_FrontY
Vehicle, 54
m_FrontZ
Vehicle, 54
m_HighVData
Event, 17
m_HighVID
Event, 17
m_InitEventParams
SSAMFuncs::SSAM, 41
m_IsActive
Event, 17
m_IsCalcPUEA
InitEventParams, 19
SSAMFuncs::SSAM, 42
m_IsCalculatePUEA
Event, 17
m_IsConflict
Event, 17
m_IsFirstTimeStep
SSAMFuncs::SSAM, 42
m_IsPETComplete
Event, 17
m_IsPrintProgress
SSAMFuncs::SSAM, 42
m_IsWriteDat
SSAMFuncs::SSAM, 42
m_LaneID
Vehicle, 54
m_LastPET
Event, 17
m_LastPETIdx
Event, 17
m_LastTTC
Event, 17

m_LastTTCIdx
Event, 17
m_Length
Vehicle, 54
m_LinkID
Vehicle, 54
m_LowVData
Event, 17
m_LowVID
Event, 17
m_MapReturn
SSAMFuncs::TimeStepData, 47
m_MaxPET
Event, 18
InitEventParams, 19
SSAMFuncs::SSAM, 42
m_MaxSpd
MotPredNameSpace::PredTraj, 30
m_MaxSpeed
MotPredNameSpace::PredMethod, 26
m_MaxTTC
Event, 18
InitEventParams, 19
SSAMFuncs::SSAM, 42
m_MaxVals
Summary, 45
m_MaxX
SSAMFuncs::Dimensions, 11
Vehicle, 54
ZoneGrid, 58
m_MaxY
SSAMFuncs::Dimensions, 11
Vehicle, 54
ZoneGrid, 58
m_MeanVals
Summary, 45
m_MinVals
Summary, 45
m_MinX
SSAMFuncs::Dimensions, 11
Vehicle, 54
ZoneGrid, 58
m_MinY
SSAMFuncs::Dimensions, 11
Vehicle, 54
ZoneGrid, 58
m_Name
MotPredNameSpace::PredMethod, 27
m_nPredTrajs
MotPredNameSpace::PredMethod, 27
m_NSteps
Event, 18
SSAMFuncs::SSAM, 42
m_NThreads
SSAMFuncs::SSAM, 42
m_NXZones
ZoneGrid, 58
m_NYZones
ZoneGrid, 58
m_NZones
ZoneGrid, 58
m_NZonesUsed
ZoneGrid, 59
m_Occupants
ZoneGrid::Zone, 56
m_OrigMaxX
ZoneGrid, 59
m_OrigMaxY
ZoneGrid, 59
m_OrigMinX
ZoneGrid, 59
m_OrigMinY
ZoneGrid, 59
m_pCurStep
SSAMFuncs::SSAM, 42
m_pDimensions
SSAMFuncs::SSAM, 42
SSAMFuncs::TrjRecord, 50
m_pNext
Vehicle, 54
m_PredPoses
MotPredNameSpace::PredTraj, 30
m_PredSpdOriens
MotPredNameSpace::PredTraj, 30
m_PreTimeStep
Event, 18
m_pSummary
SSAMFuncs::SSAM, 42
m_pVehicle
SSAMFuncs::TrjRecord, 50
m_pZoneGrid
SSAMFuncs::SSAM, 42
m_ReadTimeStep
SSAMFuncs::SSAM, 42
m_RearEndAngle
Event, 18
m_RearEndAngleThreshold
InitEventParams, 19
SSAMFuncs::SSAM, 42
m_RearX
Vehicle, 55
m_RearY
Vehicle, 55
m_RearZ
Vehicle, 55
m_RecordType
SSAMFuncs::TrjRecord, 50
m_Scale
SSAMFuncs::Dimensions, 11
Vehicle, 55
m_ScaledLength
Vehicle, 55
m_ScaledWidth
Vehicle, 55
m_Speed
Vehicle, 55
m_StartTime
SSAMFuncs::SSAM, 42
m_SteerDistri
MotPredNameSpace::PredMethod, 27

MotPredNameSpace::PredTrajRandom, 35
m_StepDataList
SSAMFuncs::SSAM, 42
m_StepSize
Event, 18
m_Summaries
SSAMFuncs::SSAM, 42
m_TimeStep
SSAMFuncs::TimeStepData, 47
SSAMFuncs::TrjRecord, 50
Vehicle, 55
m_TotalSteps
Event, 18
m_TrjDataList
SSAMFuncs::SSAM, 43
m_TrjDataLists
SSAMFuncs::SSAM, 43
m_TrjFile
SSAMFuncs::SSAM, 43
Summary, 45
m_TrjFileNames
SSAMFuncs::SSAM, 43
m_TrjSrcName
SSAMFuncs::SSAM, 43
m_Type
MotPredNameSpace::PredMethod, 27
m_Units
SSAMFuncs::Dimensions, 11
SSAMFuncs::SSAM, 43
m_UsedZones
ZoneGrid, 59
m_V1
InitEventParams, 19
m_V2
InitEventParams, 19
m_VarVals
Summary, 45
m_VehicleID
Vehicle, 55
m_VehicleMap
SSAMFuncs::TimeStepData, 47
m_VehicleVec
SSAMFuncs::TimeStepData, 47
m_Version
SSAMFuncs::InputFormat, 20
m_Width
Vehicle, 55
m_Zones
ZoneGrid, 59
m_ZoneSize
SSAMFuncs::SSAM, 43
ZoneGrid, 59
m_ZOption
SSAMFuncs::InputFormat, 20
MEASURE_LABEL
Conflict, 9
METRIC_UNITS
SSAMFuncs::Dimensions, 11
MotPredNameSpace, 5
SP_EvasiveAction, 5
SP_NormalAdaption, 5
SP_PredTraj, 5
SP_PredTrajConstant, 5
SP_PredTrajRandom, 5
MotPredNameSpace::EvasiveAction, 12
CalcPUEA, 12
EvasiveAction, 12
GenPredTrajs, 13
MotPredNameSpace::NormalAdaption, 21
CalcMTTCMPET, 21
GenPredTrajs, 22
NormalAdaption, 21
MotPredNameSpace::NormAngle, 23
MotPredNameSpace::PredMethod, 25
DetectCollision, 26
DetectCrossingZone, 26
GenPredTrajs, 26
m_AccelDistri, 26
m_MaxSpeed, 26
m_Name, 27
m_nPredTrajs, 27
m_SteerDistri, 27
m_Type, 27
PredMethod, 25
MotPredNameSpace::PredObj, 28
pos, 28
vel, 28
MotPredNameSpace::PredTraj, 29
GetControl, 29
GetNextPos, 30
GetPos, 30
m_MaxSpd, 30
m_PredPoses, 30
m_PredSpdOriens, 30
PredTraj, 29
MotPredNameSpace::PredTrajConstant, 31
GetControl, 31
m_Control, 32
PredTrajConstant, 31
SetControl, 31
MotPredNameSpace::PredTrajFactory, 33
CreatePredTraj, 33
MotPredNameSpace::PredTrajRandom, 34
GetControl, 34
m_AccelDistri, 35
m_SteerDistri, 35
PredTrajRandom, 34
SetDistributions, 34
MotPredNameSpace::TriangularDistri, 48
NormalAdaption
MotPredNameSpace::NormalAdaption, 21
NUM_CONFLICT_TYPES
Conflict, 9
NUM_MEASURES
Conflict, 9
NUM_SUMMARY_LABELS
Summary, 46
operator=
SSAMFuncs::TrjRecord, 50
Vehicle, 53

pos
MotPredNameSpace::PredObj, 28
PredMethod
MotPredNameSpace::PredMethod, 25
PredTraj
MotPredNameSpace::PredTraj, 29
PredTrajConstant
MotPredNameSpace::PredTrajConstant, 31
PredTrajRandom
MotPredNameSpace::PredTrajRandom, 34
Print
SSAMFuncs::Dimensions, 10
SSAMFuncs::InputFormat, 20
Vehicle, 53
PrintTimeStep
SSAMFuncs::SSAM, 39
ReadByte
SSAMFuncs::SSAM, 39
ReadDimensions
SSAMFuncs::SSAM, 39
ReadFloat
SSAMFuncs::SSAM, 39
ReadInputFormat
SSAMFuncs::SSAM, 39
ReadInt
SSAMFuncs::SSAM, 39
ReadTrjInputSize
SSAMFuncs::SSAM, 40
ReadVehicle
SSAMFuncs::SSAM, 40
RECORD_TYPE
SSAMFuncs::TrjRecord, 49
ResetGrid
ZoneGrid, 58
SetControl
MotPredNameSpace::PredTrajConstant, 31
SetDimensions
SSAMFuncs::SSAM, 40
SetDistributions
MotPredNameSpace::PredTrajRandom, 34
SetFormat
SSAMFuncs::SSAM, 40
SetPosition
Vehicle, 53
SetTimeStep
SSAMFuncs::SSAM, 40
SetTrjSrcName
SSAMFuncs::SSAM, 40
SetVehicle
SSAMFuncs::SSAM, 40
SP_Dimensions
SSAMFuncs, 6
SP_EvasiveAction
MotPredNameSpace, 5
SP_NormalAdaption
MotPredNameSpace, 5
SP_PredTraj
MotPredNameSpace, 5
SP_PredTrajConstant
MotPredNameSpace, 5
SP_PredTrajRandom
MotPredNameSpace, 5
SP_SSAM
SSAMFuncs, 6
SP_TimeStepData
SSAMFuncs, 6
SP_TrjRecord
SSAMFuncs, 6
SP_Zone
ZoneGrid, 57
SSAM_MEASURE
Conflict, 8
SSAMException, 44
SSAMException, 44
SSAMFuncs, 6
SP_Dimensions, 6
SP_SSAM, 6
SP_TimeStepData, 6
SP_TrjRecord, 6
TrjDataList, 6
SSAMFuncs::Dimensions, 10
ENGLISH_UNITS, 11
m_MaxX, 11
m_MaxY, 11
m_MinX, 11
m_MinY, 11
m_Scale, 11
m_Units, 11
METRIC_UNITS, 11
Print, 10
Validate, 11
SSAMFuncs::InputFormat, 20
m_Endian, 20
m_Version, 20
m_ZOption, 20
Print, 20
SSAMFuncs::SSAM, 36
Analyze, 38
AnalyzeOneStep, 38
AnalyzEvents, 38
ApplyDimensions, 38
CalcSummaries, 38
CloseRun, 38
ConvertEndianness, 38
CreateConflict, 39
DEFAULT_CROSSINGANGLE, 41
DEFAULT_PET, 41
DEFAULT_REARENDANGLE, 41
DEFAULT_TTC, 41
DetectConflicts, 39
ExportResults, 39
Initialize, 39
m_AnalysisTime, 41
m_AnalysisTimeStep, 41
m_Boundary, 41
m_ConflictList, 41
m_CrossingAngleThreshold, 41
m_CsvFileName, 41
m_DatFile, 41
m_EndTime, 41

m_EventList, 41
m_Format, 41
m_InitEventParams, 41
m_IsCalcPUEA, 42
m_IsFirstTimeStep, 42
m_IsPrintProgress, 42
m_IsWriteDat, 42
m_MaxPET, 42
m_MaxTTC, 42
m_NSteps, 42
m_NThreads, 42
m_pCurStep, 42
m_pDimensions, 42
m_pSummary, 42
m_pZoneGrid, 42
m_ReadTimeStep, 42
m_RearEndAngleThreshold, 42
m_StartTime, 42
m_StepDataList, 42
m_Summaries, 42
m_TrjDataList, 43
m_TrjDataLists, 43
m_TrjFile, 43
m_TrjFileNames, 43
m_TrjSrcName, 43
m_Units, 43
m_ZoneSize, 43
PrintTimeStep, 39
ReadByte, 39
ReadDimensions, 39
ReadFloat, 39
ReadInputFormat, 39
ReadInt, 39
ReadTrjInputSize, 40
ReadVehicle, 40
SetDimensions, 40
SetFormat, 40
SetTimeStep, 40
SetTrjSrcName, 40
SetVehicle, 40
Terminate, 40
ThrowFileReadingException, 40
ValidateInputFormat, 40
ValidateVehicle, 40
SSAMFuncs::TimeStepData, 47
AddVehicle, 47
m_MapReturn, 47
m_TimeStep, 47
m_VehicleMap, 47
m_VehicleVec, 47
SSAMFuncs::TrjRecord, 49
CopyValues, 50
DIMENSIONS, 49
FORMAT, 49
INVALID, 49
m_Format, 50
m_pDimensions, 50
m_pVehicle, 50
m_RecordType, 50
m_TimeStep, 50
operator=, 50
RECORD_TYPE, 49
TIMESTEP, 49
TrjRecord, 50
VEHICLE, 50
SSAMPoint::point, 24
SUM_MEASURES
Conflict, 9
Summary, 45
m_ConflictCounts, 45
m_MaxVals, 45
m_MeanVals, 45
m_MinVals, 45
m_TrjFile, 45
m_VarVals, 45
NUM_SUMMARY_LABELS, 46
SUMMARY_LABEL, 46
SUMMARY_LABEL
Summary, 46
Terminate
SSAMFuncs::SSAM, 40
ThrowFileReadingException
SSAMFuncs::SSAM, 40
TIMESTEP
SSAMFuncs::TrjRecord, 49
TrjDataList
SSAMFuncs, 6
TrjRecord
SSAMFuncs::TrjRecord, 50
Validate
SSAMFuncs::Dimensions, 11
ValidateInputFormat
SSAMFuncs::SSAM, 40
ValidateVehicle
SSAMFuncs::SSAM, 40
Vehicle, 51
CalcPerpOffset, 53
CalcProjection, 53
CopyValues, 53
GetV2VDistance, 53
IsCollided, 53
m_Acceleration, 54
m_CornerX, 54
m_CornerY, 54
m_FrontX, 54
m_FrontY, 54
m_FrontZ, 54
m_LaneID, 54
m_Length, 54
m_LinkID, 54
m_MaxX, 54
m_MaxY, 54
m_MinX, 54
m_MinY, 54
m_pNext, 54
m_RearX, 55
m_RearY, 55
m_RearZ, 55
m_Scale, 55
m_ScaledLength, 55

m_ScaledWidth, 55
m_Speed, 55
m_TimeStep, 55
m_VehicleID, 55
m_Width, 55
operator=, 53
Print, 53
SetPosition, 53
Vehicle, 52
VEHICLE_CORNER, 52
VEHICLE
SSAMFuncs::TrjRecord, 50
VEHICLE_CORNER
Vehicle, 52
vel
MotPredNameSpace::PredObj, 28
ZoneGrid, 57
AddVehicle, 58
ClearGrid, 58
m_MaxX, 58
m_MaxY, 58
m_MinX, 58
m_MinY, 58
m_NXZones, 58
m_NYZones, 58
m_NZones, 58
m_NZonesUsed, 59
m_OrigMaxX, 59
m_OrigMaxY, 59
m_OrigMinX, 59
m_OrigMinY, 59
m_UsedZones, 59
m_Zones, 59
m_ZoneSize, 59
ResetGrid, 58
SP_Zone, 57
ZoneGrid, 57
ZoneGrid::Zone, 56
AddVehicle, 56
Clear, 56
m_Occupants, 56