BT-Gen: A Scalable Big Trajectory Generator

USER’S Manual

Data science lab

2019

Table of Contents

[1 General Information 2](#_Toc25938919)

[1.1 System Overview 2](#_Toc25938920)

[1.2 Organization of the Manual 2](#_Toc25938921)

[2 Project Packages 2](#_Toc25938922)

[2.1 com.algorithm 2](#_Toc25938923)

[2.2 com.database 8](#_Toc25938924)

# General Information

## System Overview

BT-Gen is a trajectory generator that provides large scale generation. This application can read Digital Building Information (DBI) and use for data generation. BT-Gen also provide data visualization for validation purpose.

## Organization of the Manual

This user manual describes each packages and related class placed in projects. Each methods and attributes will explain clearly. This manual also includes some bugs that occurs during development phase.

# Project Packages

## com.algorithm

|  |  |
| --- | --- |
| Class name: | Algorithm |
| Class type: | Abstract Class |
| Description: | This class used to select positioning algorithm for RSSI data. |
| Class attributes: | inputPath: String  outputPath: String  props: Properties |
| Class methods: | public Algorithm(String proPropPath, String txtInputPath, String txtOutputPath)  public String getInputPath()  public void setInputPath(String inputPath)  public String getOutputPath()  public void setOutputPath(String outputPath)  public Properties getProps()  public void setProps(Properties props)  public abstract int calAlgorithmForAll(ExecutorService threadPool) |

|  |  |
| --- | --- |
| Class name: | AlgorithmForOne |
| Class type: | Abstract Class implements Runnable |
| Description: | This class used to embed positioning algorithm into each moving object. |
| Class attributes: | inputFileName: String  outputFileName: String  comments: String |
| Class methods: | public AlgorithmForOne()  public AlgorithmForOne(String inputFileName, String outputFileName)  public ArrayList<String> calAlgorithmForOneObj(String fileName)  public ArrayList<String> readRecord(String fileName)  public abstract ArrayList<String> calculateTrajectory(ArrayList<String> records);  protected void exportResult(ArrayList<String> result, String outputName)  public String getInputFileName()  public void setInputFileName(String inputFileName)  public String getOutputFileName()  public void setOutputFileName(String outputFileName)  public String getComments()  public void setComments(String comments) |

|  |  |
| --- | --- |
| Class name: | ClassificationTypeEnum |
| Class type: | enum |
| Description: | This class used to enumerate classification algorithm. |
| Class attributes: | WKNN(0, "algorithm\_weighted\_kNN"),  NB(1, "algorithm\_naive\_bayes"),  SVM(2, "algorithm\_svm")  classificationType: int  classificationName: String |
| Class methods: | ClassificationTypeEnum(int classificationType, String classificationName)  public int getClassificationType()  public void setClassificationType(int classificationType)  public String getClassificationName()  public void setClassificationName(String classificationName) |

|  |  |
| --- | --- |
| Class name: | DistanceMeasureTypeEnum |
| Class type: | enum |
| Description: | This class used to enumerate distance measurement algorithm. |
| Class attributes: | MAN(0, "distance\_measure\_manhattan"),  EUC(1, "distance\_measure\_euclidean")  distanceMeasureType: int  distanceMeasureName: String |
| Class methods: | DistanceMeasureTypeEnum(int distanceMeasureType, String distanceMeasureName)  public int getDistanceMeasureType()  public void setDistanceMeasureType(int distanceMeasureType)  public String getDistanceMeasureName()  public void setDistanceMeasureName(String distanceMeasureName) |

|  |  |
| --- | --- |
| Class name: | FPT |
| Class type: | Extends Algorithm class |
| Description: | This class is implementation of fingerprinting algorithm. |
| Class attributes: | outputFileName: String  seedSize: int  radioMap\_path: String  classifier: Classifier  grid\_size: double  margin\_left: double  margin\_right: double  maximum\_radiomap\_records\_for\_each: int  minimum\_radiomap\_records\_for\_each: int  k: int  positioningAlgorithmType: int  distanceMeasureType: int  dm: DistanceMeasure  outputFormatType: int  maximumSampleNumber: int  logarithmic: boolean  radiomap: Dataset |
| Class methods: | public FPT(String proPropPath, String txtInputPath, String txtOutputPath)  private Dataset buildRadioMap(double gs, int maxrr, int minrr, double ml, double mr)  private void exportRadioMap(Dataset data, String outputName)  private RadioMapRecord generateRadioMapRecord(ReferencePoint rp, Floor floor)  private void buildReferencePoints(double gs, double ml, double mr)  private List<ReferencePoint> generateReferencePointsForOneFloor(Floor floor, MBR2D mbr2d, double gs, double ml,double mr)  public int calAlgorithmForAll(ExecutorService threadPool)  public static String getRadioMap\_path()  public static void setRadioMap\_path(String radioMap\_path)  public Classifier getClassifier()  public void setClassifier(Classifier classifier)  public double getGrid\_size()  public void setGrid\_size(double grid\_size)  public double getMargin\_left()  public void setMargin\_left(double margin\_left)  public double getMargin\_right()  public void setMargin\_right(double margin\_right)  public int getMaximum\_radiomap\_records\_for\_each()  public void setMaximum\_radiomap\_records\_for\_each(int maximum\_radiomap\_records\_for\_each)  public int getMinimum\_radiomap\_records\_for\_each()  public void setMinimum\_radiomap\_records\_for\_each(int minimum\_radiomap\_records\_for\_each)  public int getK()  public void setK(int k)  public int getPositioningAlgorithmType()  public void setPositioningAlgorithmType(int positioningAlgorithmType)  public Dataset getRadiomap()  public void setRadiomap(Dataset radiomap)  public int getOutputFormatType()  public void setOutputFormatType(int outputFormatType)  public int getDistanceMeasureType()  public void setDistanceMeasureType(int distanceMeasureType)  public boolean isLogarithmic()  public void setLogarithmic(boolean logarithmic)  public int getMaximumSampleNumber()  public void setMaximumSampleNumber(int maximumSampleNumber) |

|  |  |
| --- | --- |
| Class name: | FPTForOne |
| Class type: | Extends AlgorithmForOne class |
| Description: | This class is implementation of fingerprinting algorithm for each moving object. |
| Class attributes: | classifier: Classifier  k: int  positioningAlgorithmType: int  outputFormatType: int  maximumSampleNumber: int |
| Class methods: | public FPTForOne(String inputFileName, String outputFileName)  public FPTForOne(String inputFileName, String outputFileName, int positioningAlgorithmType, Classifier classifier, int k, int outputFormatType, int maximumSampleNumber)  public void run()  public ArrayList<String> calculateTrajectory(ArrayList<String> records)  private IndoorLocation calculateLocationDeterWKNN(ArrayList<RSSIPackage> packs)  private IndoorLocation calculateLocationDeter(ArrayList<RSSIPackage> packs)  private SortedMap<Double, IndoorLocation> calculateLocationProb(ArrayList<RSSIPackage> packs)  public Classifier getClassifier()  public void setClassifier(Classifier classifier)  public int getOutputFormatType()  public void setOutputFormatType(int outputFormatType)  public int getPositioningAlgorithmType()  public void setPositioningAlgorithmType(int positioningAlgorithmType)  public int getMaximumSampleNumber()  public void setMaximumSampleNumber(int maximumSampleNumber) |

|  |  |
| --- | --- |
| Class name: | IndoorLocation |
| Class type: | Public class |
| Description: | This class is blueprint for indoor location object. |
| Class attributes: | x: double  y: double  floor: Floor  partition: Partition |
| Class methods: | public IndoorLocation()  public IndoorLocation(Point2D.Double point)  public IndoorLocation(Point2D.Double point, Floor floor)  public IndoorLocation(double x, double y)  public IndoorLocation(double x, double y, Floor floor)  public double getX()  public void setX(double x)  public double getY()  public void setY(double y)  public Floor getFloor()  public void setFloor(Floor floor)  public Partition getPartition()  public void setPartition(Partition partition)  public String toString() |

|  |  |
| --- | --- |
| Class name: | OutputFormatTypeEnum |
| Class type: | Enum class |
| Description: | This class is enumerate for output format. |
| Class attributes: | DETER(0, "output\_format\_deterministic"),  PROB(1, "output\_format\_probabilistic")  outputFormatType: int  outputFormatName: String |
| Class methods: | OutputFormatTypeEnum(int outputFormatType, String outputFormatName)  public int getOutputFormatType()  public void setOutputFormatType(int outputFormatType)  public String getOutputFormatName()  public void setOutputFormatName(String outputFormatName) |

|  |  |
| --- | --- |
| Class name: | ProximityRecord |
| Class type: | Public class |
| Description: | This class is blueprint for proximity algorithm. |
| Class attributes: | objectId: int  stationId: int  measurements: ArrayList<RSSIPackage> |
| Class methods: | public ProximityRecord(int objectId, int stationId)  public ProximityRecord(int objectId, int stationId, ArrayList<RSSIPackage> measurements)  public int getObjectId()  public void setObjectId(int objectId)  public ArrayList<RSSIPackage> getMeasurements()  public void setMeasurements(ArrayList<RSSIPackage> measurements)  public int getStationId()  public void setStationId(int stationId)  public int addMeasurements(RSSIPackage measurement)  public String toString() |

|  |  |
| --- | --- |
| Class name: | PXM |
| Class type: | Public class extends Algorithm class |
| Description: | This class used to measure proximity algorithm. |
| Class attributes: | outputFileName: String |
| Class methods: | public PXM(String proPropPath, String txtInputPath, String txtOutputPath)  public int calAlgorithmForAll(ExecutorService threadPool) |

|  |  |
| --- | --- |
| Class name: | PXMForOne |
| Class type: | Public class extends AlgorithmForOne class |
| Description: | This class used to measure proximity algorithm for each moving object. |
| Class attributes: | objectId: int  currentProximityRecords: Hashtable<Integer, ProximityRecord>() |
| Class methods: | public PXMForOne(String inputFileName, String outputFileName)  public void run()  public ArrayList<String> calculateTrajectory(ArrayList<String> records)  private RSSIPackage calculateStrongestPack(ArrayList<RSSIPackage> packs)  public int getObjectId()  public void setObjectId(int objectId) |

|  |  |
| --- | --- |
| Class name: | RadioMapRecord |
| Class type: | Public class |
| Description: | This class is blueprint for creating RadioMapRecord object. |
| Class attributes: | objectId: int  currentProximityRecords: Hashtable<Integer, ProximityRecord>() |
| Class methods: | public PXMForOne(String inputFileName, String outputFileName)  public void run()  public ArrayList<String> calculateTrajectory(ArrayList<String> records)  private RSSIPackage calculateStrongestPack(ArrayList<RSSIPackage> packs)  public int getObjectId()  public void setObjectId(int objectId) |

|  |  |
| --- | --- |
| Class name: | ReferencePoint |
| Class type: | Public class |
| Description: | This class is blueprint for creating ReferencePoint object. |
| Class attributes: | rp\_id: int  location: Point2D.Double |
| Class methods: | public ReferencePoint(Double location)  public ReferencePoint(int rp\_id, Double location)  public int getRp\_id()  public void setRp\_id(int rp\_id)  public Point2D.Double getLocation()  public void setLocation(Point2D.Double location) |

|  |  |
| --- | --- |
| Class name: | TRI |
| Class type: | Public class extends Algorithm class |
| Description: | This class is blueprint for trilateration algorithm. |
| Class attributes: | outputFileName: String  n: int  rssiAt1: double |
| Class methods: | public TRI(String proPropPath, String txtInputPath, String txtOutputPath)  public int calAlgorithmForAll(ExecutorService threadPool) |

|  |  |
| --- | --- |
| Class name: | TRIForOne |
| Class type: | Public class extends AlgorithmForOne class |
| Description: | This class used for calculating the indoor position using trilateration algorithm for each moving object. |
| Class attributes: | n: int  rssiAt1: double  lastPosition: IndoorLocation |
| Class methods: | public TRIForOne(String inputFileName, String outputFileName)  public TRIForOne(String inputFileName, String outputFileName, int n, double rssiAt1)  public void run()  public ArrayList<String> calculateTrajectory(ArrayList<String> records)  public IndoorLocation calculateLocation(ArrayList<RSSIPackage> packs)  public Point2D.Double calculateIntersect(RSSIPackage pack1, RSSIPackage pack2, RSSIPackage pack3) |

## com.database

|  |  |
| --- | --- |
| Class name: | DB\_Connection |
| Class type: | Public class |
| Description: | This class is blueprint for database connection. |
| Class attributes: | Comments: Boolean  in: FileInputStream  props: Properties |
| Class methods: | private static void loadProp(String dbProp)  private static Connection initiateCon(String dbProp) throws SQLException  public static Connection connectToDatabase(String dbProp) |

|  |  |
| --- | --- |
| Class name: | DB\_Create |
| Class type: | Public class |
| Description: | This class is blueprint for database creation. |
| Class attributes: | DB\_MOOVET: String  DB\_MOOVEW: String  T\_ITEM: String  ITEM\_NAME: String  ITEM\_ITEMID: String  ITEM\_GLOBALID: String  T\_FLOORITEM: String  FLOORITEM\_FLOORID: String  T\_FLOOR: String  FLOOR\_BUILDINGID: String  T\_BUILDING: String  BUILDING\_FILEID: String  T\_PARTITION: String  PART\_GEOM: String  T\_DECOMPREL: String  DECO\_ORIGINAL: String  DECO\_DECOMP: String  T\_ACCESSPOINT: String  AP\_TYPE: String  AP\_LOCATION: String  AP\_LINE: String  T\_CONNECTOR: String  CONN\_UPPERPOINT: String  CONN\_UPPERFLOOR: String  T\_APTOPART: String  A2P\_APID: String  A2P\_PARTID: String  T\_CONTOPART: String  C2P\_CONID: String  C2P\_PARTID: String  T\_CONNECTIVITY: String  CON\_APID: String  CON\_PART1ID: String  CON\_PART2ID: String  T\_ACCESSRULE: String  ACC\_ID: String  ACC\_CONID: String  ACC\_NAME: String  ACC\_DIRECTION: String |
| Class methods: | public static void RecreateMooveTempalte(Connection con) throws SQLException  public static void RecreateMooveWork(Connection con) throws SQLException  public static void CreateTablesForMooveTemplate(Connection con) throws SQLException |

|  |  |
| --- | --- |
| Class name: | DB\_CreateFunction |
| Class type: | Public class |
| Description: | This class used for creating random points in polygon. |
| Class attributes: | - |
| Class methods: | public static void CreateRandomPointsInPolygon(Connection con) throws SQLException  public static void main(String[] args) throws SQLException |

|  |  |
| --- | --- |
| Class name: | DB\_FileUploader |
| Class type: | Public class |
| Description: | This class used to upload IFC file into database. |
| Class attributes: | Connection: Connection  Pst: PreparedStatement  Fis: FileInputStream |
| Class methods: | public DB\_FileUploader()  public boolean saveObjectToDB(UploadObject object, File file) |

|  |  |
| --- | --- |
| Class name: | DB\_Import |
| Class type: | Public class |
| Description: | This class used to import IFC component into database. |
| Class attributes: | decompBeforeMapping: boolean |
| Class methods: | public static void importAll(Connection con, Integer fileID, File file) throws SQLException  private static void offsetAlignment(Connection con, ArrayList<Partition> partitions, ArrayList<Door> doors)  private static void importBuilding(Connection con, Integer fileID) throws SQLException  private static ArrayList<com.ifc.dataextraction.spatialobject.Floor> importFloors(Connection con) throws SQLException  private static ArrayList<Point2D.Double> checkIntermediate(Connection con, ArrayList<Point2D.Double> points)  private static ArrayList<Partition> importPartitions(Connection con) throws SQLException  private static ArrayList<Door> importDoors(Connection con) throws SQLException  public static void decompose(Connection con) throws SQLException  public static void mapD2P(Connection con) throws SQLException  public static void importStairs(Connection con,ArrayList<com.ifc.dataextraction.spatialobject.Floor> floors, ArrayList<Partition> partitions) throws SQLException  public static void importElevators(Connection con, ArrayList<com.ifc.dataextraction.spatialobject.Floor> floors, ArrayList<Partition> partitions) throws SQLException |

|  |  |
| --- | --- |
| Class name: | DB\_ WrapperDelete |
| Class type: | Public class extends DB\_Create class |
| Description: | This class used to empty value in table. |
| Class attributes: | - |
| Class methods: | public DB\_WrapperDelete()  public static void flushAP2PartTable(Connection con, Integer file\_id) throws SQLException  private static void flushConnectivityTable(Connection con, Integer file\_id) throws SQLException  public static void deletePartition(Connection con, Partition part) throws SQLException  public static void deleteAccessPoint(Connection con, AccessPoint ap) throws SQLException  public static void deleteFile(Connection con, UploadObject uo) throws SQLException |

|  |  |
| --- | --- |
| Class name: | DB\_WrapperInsert |
| Class type: | Public class extends DB\_Create class |
| Description: | This class used to insert value into table. |
| Class attributes: | - |
| Class methods: | public static void insertBuilding(Connection con, String name, String globalid, Integer fileID) throws SQLException  public static void insertFloor(Connection con, String name, String globalid, String buildingGlobalID) throws SQLException  public static Integer insertPartition(Connection con, String name, String globalid, String floorglobalid, ArrayList<Point2D.Double> polyline) throws SQLException  public static Integer insertPartition(Connection con, String name, String globalid, String floorglobalid, Polygon2D.Double polygon) throws SQLException  public static Integer insertAccessPointWithGeom(Connection con, String name, Point2D.Double placement, String globalid, String floorglobalid, Integer apType, Line2D.Double line) throws SQLException  public static Integer insertVirtualAccessPointWithGeom(Connection con, String name1, String name2, Point2D.Double placement, String globalid, String floorglobalid, Integer apType, Line2D.Double line) throws SQLException  public static void flushPartitionForAPConnections(Connection con, String globalidDoor) throws SQLException  public static void flushPartitionForConnectors(Connection con, String globalidStair) throws SQLException  public static Integer insertConnector(Connection con, String name, String globalid, Point2D.Double location, String globalidfloor, Point2D.Double upperPoint, String upperFloorID) throws SQLException  public static Integer insertAccessPointConnector(Connection con, String name, String pointGeom, String globalid, Integer floorid, Integer apType) throws SQLException  public static void connectPartAndAP(Connection con, Integer partID, Integer apID) throws SQLException  public static void connectPartAndCon(Connection con, Integer partID, Integer conID) throws SQLException  public static void connectPartAndCon(Connection con, String partGlobalID, Integer conID) throws SQLException  public static void connectPartAndAP(Connection con, String globalidPart, String globalidAP) throws SQLException  public static void insertDecompRel(Connection con, Integer partOriginalID, Integer partDecompID) throws SQLException  public static void insertDoorBetweenPart(Connection con, Partition from, Partition to) throws SQLException  private static Door createDoorForConRenewed(Partition from, Partition to) throws SQLException  private static Double findCommon(Connection con, Line2D.Double a, Line2D.Double b) throws SQLException |

|  |  |
| --- | --- |
| Class name: | DB\_WrapperLoad |
| Class type: | Public class extends DB\_Create class |
| Description: | This class used to run query and serialize the results. |
| Class attributes: | public static ArrayList<Floor> floorT = new ArrayList<Floor>();  public static ArrayList<Partition> partitionT = new ArrayList<Partition>();  public static ArrayList<Partition> partitionDecomposedT = new ArrayList<Partition>();  public static ArrayList<AccessPoint> accesspointT = new ArrayList<AccessPoint>()  decomprelT: ArrayList<DecompRel>()  aptopartT: ArrayList<ApToPart>()  contopartT: ArrayList<ConToPart>()  connectorT: ArrayList<Connector>()  accessruleT: ArrayList<AccessRule>()  connectivityT: ArrayList<Connectivity>()  accessPointConnectorT: ArrayList<AccessPoint>() |
| Class methods: | public DB\_WrapperDelete()  public static void flushAP2PartTable(Connection con, Integer file\_id) throws SQLException  private static void flushConnectivityTable(Connection con, Integer file\_id) throws SQLException  public static void deletePartition(Connection con, Partition part) throws SQLException  public static void deleteAccessPoint(Connection con, AccessPoint ap) throws SQLException  public static void deleteFile(Connection con, UploadObject uo) throws SQLException |