Pearl Class

The Pearl class implements the PEARL algorithm, that is an improvement from the RANSAC algorithm. The RANSAC is an iteration algorithm that chooses a random data-set, and from this, finds a line. If the lines found don't meet the requirements, iterate again. Pearl algorithm does basically the same thing but keeps the lines found in the previous iteration to calculate the next set of models. To search for better results, the algorithm tries to minimize a quantity called "Energy" of the set. First, some definitions and recapitulations.

- Point is a (x,y) pair of coordinates and can be attached to any model, or neither, being an outlier
- A model is a line in the form y = ax + b.
- A set is a combination of all models and the outliers.

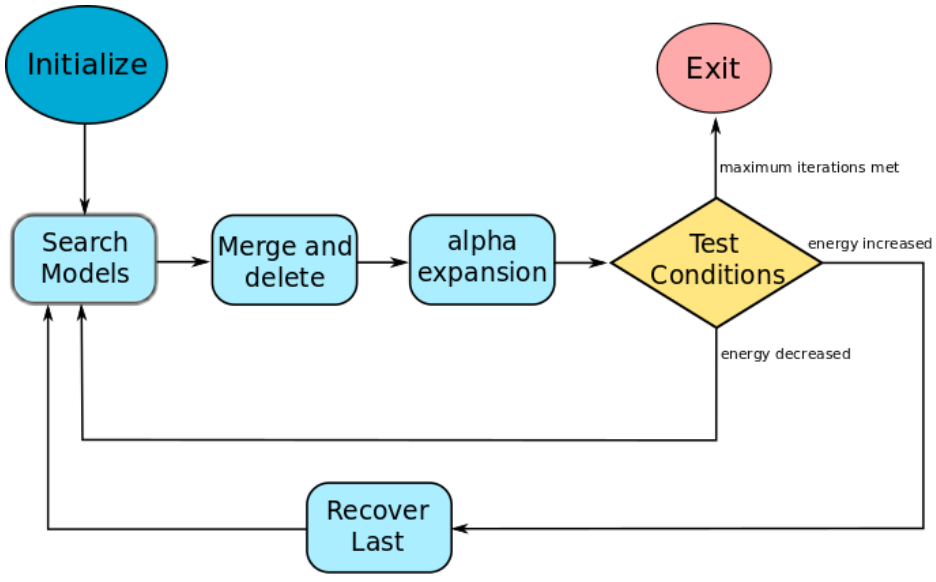
We will divide the set's energy calculation into three parts.

1. The model's energy being the sum of distances from a model to all its attached points $E_{model}=\sum_{i=1}^{n}\frac{|ax_i-y_i+b|}{\sqrt{a^2+1}}$ where n is the number of points in the model

2. The outliers' energy is a fixed value for each point that is not attached to any model $E_{outliers}=\rho n_{outliers}$ where ρ is a constant value and $n_{outliers}$ is the number of outliers in the set

3. The penalty energy that is a way to penalize two close points that are in a different model. $E_{penalty}=\lambda\cdot\sum_{p\in m_1}\sum_{q\in m_2}\omega_{pq}$ where λ is a constant to scale the sum, m_1 *and* m_2 are two different models in the set, p and q are the points that are attached to models m_1 *and* m_2 respectively and ω_{pq} the function to calculate the additional energy between points. For this algorithm, we chose $\omega_{pq}=e^{-\frac{((p_x-q_x)^2+(p_y-q_y)^2)}{\xi^2}}$ to be that function.

This way, the set's total energy is $E_{set}=E_{outliers}+E_{penalty}+\sum_{m\in set}E_m$ and the quantity the algorithm will try to minimize. To achieve this, the algorithm will proceed as described in this flow chart :



- Initialize: Sets all variables and populate outlier field with point
- Search models: it picks 3 random points in the field, and calculate the linear fit to find the best line. Repeating this process until half of outliers are attached to some model will give us a good amount of models
- Merge and delete: First, it fuses models that are similar, having both slope and intercept reasonably close, combining their points and recalculating the linear fit. Then, using an Energy/NofPoints ratio creteria, we remove models that are considered not good enough.
- Alpha expansion: Detaches every point from its model, and reattaches it to the closest model in the set, calculating set's total energy in the process.
- Test Conditions: During iterations, we keep track of the best set and its energy, this way we can recover the best state to try again until the algorithm performs its maximum number of iterations.
- After all iterations the algorithm will return all models that survived, then another algorithm will chose witch one, but this is for later.

Public Methods

◆ Pearl( )

Pearl::Pearl ( )

Contructor that creates all the vectors.

◆ populateOutliers( )

void Pearl::populateOutliers (const sensor_msgs::LaserScan & msg)

Populate outliers' vector with all the points in the ros message

Parameters

msg is the Ros message containing the LIDAR's data

◆ findLines( )

std::vector<Model> Pearl::findLines ( )

Uses the PEARL algorithm to find the lines into the cloud of points, then return the best result.

- getInitialField( )
  std::vector<Point> Pearl::getInitialField ( ) const

  Gets the initial field of points, this function will be more useful in later versions.

Protected Methods

- removeModel( )
  void Pearl::removeModel (const int modelIndex)

  Remove the model in the index ‘modelIndex’

  Parameters
  modelIndex is the index to be deleted

- removePointsInModels( )
  void Pearl::removePointsInModels ( )

  Detach the points in all models contained in the set

- randomPointsInField( )
  std::vector<Point> Pearl::randomPointsInField (const int num)

  Return a vector containing ‘num’ different points in the outliers vector

  Parameters
  num is the amount of points to be selected

- searchModels( )
  void Pearl::searchModels (const int nbOfModels)

  Search for ‘nbOfModels’ models using the points in the outliers

  Parameters
  nbOfModels is the amount of models to be searched

- redistributePoints( )
  double Pearl::redistributePoints ( )

  Detaches points from all models and attaches it to the closest model, returns the sum of all models’ energy

- ## removeTinyModels( )

  double Pearl::removeTinyModels (const int points = 3)

  Remove models that have ‘points’ or less. Returns the difference in models’ energy, since some models were deleted and its points sent to outliers

  Parameters

  points is the amount of point that, if a model have this amount or less, will be removed

- ## calculateAdditionalEnergy( )

  double Pearl::calculateAdditionalEnergy ( ) const

  Calculates the $E_{penalty}$ for the set, and returns the additional energy.

- ## expansionEnergy( )

  double Pearl::expansionEnergy ( )

  Does the alpha-expansion, redistributing every point to the closest one, removing tiny models and returns set's final energy.

- ## reEstimation( )

  void Pearl::reEstimation ( )

  Recalculates the linear fit for every model in the set, using their attached points

- ## eraseBadModels( )

  void Pearl::eraseBadModels (const double threshRatio)

  Remove all models that have the ratio Energy / Number of points bigger than ‘threshRatio’

  Parameters

  threshRatio is the minimum ratio allowed

- ## fuseEqualModels( )

  void Pearl::fuseEqualModels ( )

  Fuse models, two by two, that are considered close by having both slope and both intercept reasonably close

- friend operator << ( )

  std::ostream & operator << (std::ostream & out, const Pearl & p)

  Print pearl object in terminal.

  Parameters

  out is where to print, normally terminal

  p is the object to be printed