

---

# **Cartographer Documentation**

**The Cartographer Authors**

**Sep 14, 2018**



<b>1</b>	<b>Configuration</b>	<b>1</b>
1.1	cartographer.common.proto.CeresSolverOptions . . . . .	1
1.2	cartographer.mapping.pose_graph.proto.ConstraintBuilderOptions . . . . .	1
1.3	cartographer.mapping.pose_graph.proto.OptimizationProblemOptions . . . . .	2
1.4	cartographer.mapping.proto.MapBuilderOptions . . . . .	2
1.5	cartographer.mapping.proto.MotionFilterOptions . . . . .	2
1.6	cartographer.mapping.proto.PoseGraphOptions . . . . .	3
1.7	cartographer.mapping.proto.TrajectoryBuilderOptions . . . . .	3
1.8	cartographer.mapping_2d.proto.LocalTrajectoryBuilderOptions . . . . .	3
1.9	cartographer.mapping_2d.proto.RangeDataInserterOptions . . . . .	4
1.10	cartographer.mapping_2d.proto.SubmapsOptions . . . . .	4
1.11	cartographer.mapping_2d.scan_matching.proto.CeresScanMatcherOptions . . . . .	4
1.12	cartographer.mapping_2d.scan_matching.proto.FastCorrelativeScanMatcherOptions . . . . .	4
1.13	cartographer.mapping_2d.scan_matching.proto.RealTimeCorrelativeScanMatcherOptions . . . . .	5
1.14	cartographer.mapping_3d.proto.LocalTrajectoryBuilderOptions . . . . .	5
1.15	cartographer.mapping_3d.proto.RangeDataInserterOptions . . . . .	6
1.16	cartographer.mapping_3d.proto.SubmapsOptions . . . . .	6
1.17	cartographer.mapping_3d.scan_matching.proto.CeresScanMatcherOptions . . . . .	6
1.18	cartographer.mapping_3d.scan_matching.proto.FastCorrelativeScanMatcherOptions . . . . .	6
1.19	cartographer.sensor.proto.AdaptiveVoxelFilterOptions . . . . .	7
<b>2</b>	<b>Evaluation</b>	<b>9</b>
2.1	Concept . . . . .	9
2.2	Advantages & Limitations . . . . .	10
2.3	How-To . . . . .	10
2.4	References . . . . .	11
<b>3</b>	<b>Terminology</b>	<b>13</b>
3.1	Frames . . . . .	13
3.2	Transforms . . . . .	13
<b>4</b>	<b>Cost functions</b>	<b>15</b>
4.1	Relative Transform Error 2D . . . . .	15
4.2	Landmark Cost Function . . . . .	15
<b>5</b>	<b>Migration tool for pbstream files</b>	<b>17</b>

<b>6</b>	<b>Technical Overview</b>	<b>19</b>
<b>7</b>	<b>Getting started</b>	<b>21</b>
7.1	Getting started with ROS . . . . .	21
7.2	Getting started without ROS . . . . .	21
<b>8</b>	<b>System Requirements</b>	<b>23</b>
8.1	Known Issues . . . . .	23
<b>9</b>	<b>How to cite us</b>	<b>25</b>

### 1.1 `cartographer.common.proto.CeresSolverOptions`

**bool use\_nonmonotonic\_steps** Configure the Ceres solver. See the Ceres documentation for more information: <https://code.google.com/p/ceres-solver/>

**int32 max\_num\_iterations** Not yet documented.

**int32 num\_threads** Not yet documented.

### 1.2 `cartographer.mapping.pose_graph.proto.ConstraintBuilderOptions`

**double sampling\_ratio** A constraint will be added if the proportion of added constraints to potential constraints drops below this number.

**double max\_constraint\_distance** Threshold for poses to be considered near a submap.

**double min\_score** Threshold for the scan match score below which a match is not considered. Low scores indicate that the scan and map do not look similar.

**double global\_localization\_min\_score** Threshold below which global localizations are not trusted.

**double loop\_closure\_translation\_weight** Weight used in the optimization problem for the translational component of loop closure constraints.

**double loop\_closure\_rotation\_weight** Weight used in the optimization problem for the rotational component of loop closure constraints.

**bool log\_matches** If enabled, logs information of loop-closing constraints for debugging.

**cartographer.mapping\_2d.scan\_matching.proto.FastCorrelativeScanMatcherOptions fast\_correlative\_scan\_matcher\_options** Options for the internally used scan matchers.

**cartographer.mapping\_2d.scan\_matching.proto.CeresScanMatcherOptions ceres\_scan\_matcher\_options** Not yet documented.

**cartographer.mapping\_3d.scan\_matching.proto.FastCorrelativeScanMatcherOptions fast\_correlative\_scan\_matcher\_options\_3d**  
Not yet documented.

**cartographer.mapping\_3d.scan\_matching.proto.CeresScanMatcherOptions ceres\_scan\_matcher\_options\_3d**  
Not yet documented.

### 1.3 cartographer.mapping.pose\_graph.proto.OptimizationProblemOptions

**double huber\_scale** Scaling parameter for Huber loss function.

**double acceleration\_weight** Scaling parameter for the IMU acceleration term.

**double rotation\_weight** Scaling parameter for the IMU rotation term.

**double local\_slam\_pose\_translation\_weight** Scaling parameter for translation between consecutive nodes based on the local SLAM pose.

**double local\_slam\_pose\_rotation\_weight** Scaling parameter for rotation between consecutive nodes based on the local SLAM pose.

**double odometry\_translation\_weight** Scaling parameter for translation between consecutive nodes based on the odometry.

**double odometry\_rotation\_weight** Scaling parameter for rotation between consecutive nodes based on the odometry.

**double fixed\_frame\_pose\_translation\_weight** Scaling parameter for the FixedFramePose translation.

**double fixed\_frame\_pose\_rotation\_weight** Scaling parameter for the FixedFramePose rotation.

**bool log\_solver\_summary** If true, the Ceres solver summary will be logged for every optimization.

**cartographer.common.proto.CeresSolverOptions ceres\_solver\_options** Not yet documented.

### 1.4 cartographer.mapping.proto.MapBuilderOptions

**bool use\_trajectory\_builder\_2d** Not yet documented.

**bool use\_trajectory\_builder\_3d** Not yet documented.

**int32 num\_background\_threads** Number of threads to use for background computations.

**cartographer.mapping.proto.PoseGraphOptions pose\_graph\_options** Not yet documented.

### 1.5 cartographer.mapping.proto.MotionFilterOptions

**double max\_time\_seconds** Threshold above which range data is inserted based on time.

**double max\_distance\_meters** Threshold above which range data is inserted based on linear motion.

**double max\_angle\_radians** Threshold above which range data is inserted based on rotational motion.

## 1.6 cartographer.mapping.proto.PoseGraphOptions

**int32 optimize\_every\_n\_nodes** Online loop closure: If positive, will run the loop closure while the map is built.

**cartographer.mapping.pose\_graph.proto.ConstraintBuilderOptions constraint\_builder\_options** Options for the constraint builder.

**double matcher\_translation\_weight** Weight used in the optimization problem for the translational component of non-loop-closure scan matcher constraints.

**double matcher\_rotation\_weight** Weight used in the optimization problem for the rotational component of non-loop-closure scan matcher constraints.

**cartographer.mapping.pose\_graph.proto.OptimizationProblemOptions optimization\_problem\_options** Options for the optimization problem.

**int32 max\_num\_final\_iterations** Number of iterations to use in ‘optimization\_problem\_options’ for the final optimization.

**double global\_sampling\_ratio** Rate at which we sample a single trajectory’s nodes for global localization.

**bool log\_residual\_histograms** Whether to output histograms for the pose residuals.

**double global\_constraint\_search\_after\_n\_seconds** If for the duration specified by this option no global constraint has been added between two trajectories, loop closure searches will be performed globally rather than in a smaller search window.

## 1.7 cartographer.mapping.proto.TrajectoryBuilderOptions

**cartographer.mapping\_2d.proto.LocalTrajectoryBuilderOptions trajectory\_builder\_2d\_options** Not yet documented.

**cartographer.mapping\_3d.proto.LocalTrajectoryBuilderOptions trajectory\_builder\_3d\_options** Not yet documented.

**bool pure\_localization** Not yet documented.

## 1.8 cartographer.mapping\_2d.proto.LocalTrajectoryBuilderOptions

**float min\_range** Rangefinder points outside these ranges will be dropped.

**float max\_range** Not yet documented.

**float min\_z** Not yet documented.

**float max\_z** Not yet documented.

**float missing\_data\_ray\_length** Points beyond ‘max\_range’ will be inserted with this length as empty space.

**int32 num\_accumulated\_range\_data** Number of range data to accumulate into one unwrapped, combined range data to use for scan matching.

**float voxel\_filter\_size** Voxel filter that gets applied to the range data immediately after cropping.

**cartographer.sensor.proto.AdaptiveVoxelFilterOptions adaptive\_voxel\_filter\_options** Voxel filter used to compute a sparser point cloud for matching.

**cartographer.sensor.proto.AdaptiveVoxelFilterOptions loop\_closure\_adaptive\_voxel\_filter\_options** Voxel filter used to compute a sparser point cloud for finding loop closures.

**bool use\_online\_correlative\_scan\_matching** Whether to solve the online scan matching first using the correlative scan matcher to generate a good starting point for Ceres.

**cartographer.mapping\_2d.scan\_matching.proto.RealTimeCorrelativeScanMatcherOptions real\_time\_correlative\_scan\_matcher\_options** Not yet documented.

**cartographer.mapping\_2d.scan\_matching.proto.CeresScanMatcherOptions ceres\_scan\_matcher\_options** Not yet documented.

**cartographer.mapping\_2d.proto.MotionFilterOptions motion\_filter\_options** Not yet documented.

**double imu\_gravity\_time\_constant** Time constant in seconds for the orientation moving average based on observed gravity via the IMU. It should be chosen so that the error 1. from acceleration measurements not due to gravity (which gets worse when the constant is reduced) and 2. from integration of angular velocities (which gets worse when the constant is increased) is balanced.

**cartographer.mapping\_2d.proto.SubmapsOptions submaps\_options** Not yet documented.

**bool use\_imu\_data** True if IMU data should be expected and used.

## 1.9 cartographer.mapping\_2d.proto.RangeDataInserterOptions

**double hit\_probability** Probability change for a hit (this will be converted to odds and therefore must be greater than 0.5).

**double miss\_probability** Probability change for a miss (this will be converted to odds and therefore must be less than 0.5).

**bool insert\_free\_space** If 'false', free space will not change the probabilities in the occupancy grid.

## 1.10 cartographer.mapping\_2d.proto.SubmapsOptions

**double resolution** Resolution of the map in meters.

**int32 num\_range\_data** Number of range data before adding a new submap. Each submap will get twice the number of range data inserted: First for initialization without being matched against, then while being matched.

**cartographer.mapping\_2d.proto.RangeDataInserterOptions range\_data\_inserter\_options** Not yet documented.

## 1.11 cartographer.mapping\_2d.scan\_matching.proto.CeresScanMatcherOptions

**double occupied\_space\_weight** Scaling parameters for each cost functor.

**double translation\_weight** Not yet documented.

**double rotation\_weight** Not yet documented.

**cartographer.common.proto.CeresSolverOptions ceres\_solver\_options** Configure the Ceres solver. See the Ceres documentation for more information: <https://code.google.com/p/ceres-solver/>

## 1.12 cartographer.mapping\_2d.scan\_matching.proto.FastCorrelativeScanMatcherOptions

**double linear\_search\_window** Minimum linear search window in which the best possible scan alignment will be found.



**double angular\_search\_window** Minimum angular search window in which the best possible scan alignment will be found.

**int32 branch\_and\_bound\_depth** Number of precomputed grids to use.

## 1.13 cartographer.mapping\_2d.scan\_matching.proto.RealTimeCorrelativeScan

**double linear\_search\_window** Minimum linear search window in which the best possible scan alignment will be found.

**double angular\_search\_window** Minimum angular search window in which the best possible scan alignment will be found.

**double translation\_delta\_cost\_weight** Weights applied to each part of the score.

**double rotation\_delta\_cost\_weight** Not yet documented.

## 1.14 cartographer.mapping\_3d.proto.LocalTrajectoryBuilderOptions

**float min\_range** Rangefinder points outside these ranges will be dropped.

**float max\_range** Not yet documented.

**int32 num\_accumulated\_range\_data** Number of range data to accumulate into one unwarped, combined range data to use for scan matching.

**float voxel\_filter\_size** Voxel filter that gets applied to the range data immediately after cropping.

**cartographer.sensor.proto.AdaptiveVoxelFilterOptions high\_resolution\_adaptive\_voxel\_filter\_options** Voxel filter used to compute a sparser point cloud for matching.

**cartographer.sensor.proto.AdaptiveVoxelFilterOptions low\_resolution\_adaptive\_voxel\_filter\_options** Not yet documented.

**bool use\_online\_correlative\_scan\_matching** Whether to solve the online scan matching first using the correlative scan matcher to generate a good starting point for Ceres.

**cartographer.mapping\_2d.scan\_matching.proto.RealTimeCorrelativeScanMatcherOptions real\_time\_correlative\_scan\_matcher\_options** Not yet documented.

**cartographer.mapping\_3d.scan\_matching.proto.CeresScanMatcherOptions ceres\_scan\_matcher\_options** Not yet documented.

**cartographer.mapping.proto.MotionFilterOptions motion\_filter\_options** Not yet documented.

**double imu\_gravity\_time\_constant** Time constant in seconds for the orientation moving average based on observed gravity via the IMU. It should be chosen so that the error 1. from acceleration measurements not due to gravity (which gets worse when the constant is reduced) and 2. from integration of angular velocities (which gets worse when the constant is increased) is balanced.

**int32 rotational\_histogram\_size** Number of histogram buckets for the rotational scan matcher.

**cartographer.mapping\_3d.proto.SubmapsOptions submaps\_options** Not yet documented.

## 1.15 cartographer.mapping\_3d.proto.RangeDataInserterOptions

**double hit\_probability** Probability change for a hit (this will be converted to odds and therefore must be greater than 0.5).

**double miss\_probability** Probability change for a miss (this will be converted to odds and therefore must be less than 0.5).

**int32 num\_free\_space\_voxels** Up to how many free space voxels are updated for scan matching. 0 disables free space.

## 1.16 cartographer.mapping\_3d.proto.SubmapsOptions

**double high\_resolution** Resolution of the 'high\_resolution' map in meters used for local SLAM and loop closure.

**double high\_resolution\_max\_range** Maximum range to filter the point cloud to before insertion into the 'high\_resolution' map.

**double low\_resolution** Resolution of the 'low\_resolution' version of the map in meters used for local SLAM only.

**int32 num\_range\_data** Number of range data before adding a new submap. Each submap will get twice the number of range data inserted: First for initialization without being matched against, then while being matched.

**cartographer.mapping\_3d.proto.RangeDataInserterOptions range\_data\_inserter\_options** Not yet documented.

## 1.17 cartographer.mapping\_3d.scan\_matching.proto.CeresScanMatcherOptions

**double occupied\_space\_weight** Scaling parameters for each cost functor.

**double translation\_weight** Not yet documented.

**double rotation\_weight** Not yet documented.

**bool only\_optimize\_yaw** Whether only to allow changes to yaw, keeping roll/pitch constant.

**cartographer.common.proto.CeresSolverOptions ceres\_solver\_options** Configure the Ceres solver. See the Ceres documentation for more information: <https://code.google.com/p/ceres-solver/>

## 1.18 cartographer.mapping\_3d.scan\_matching.proto.FastCorrelativeScanMatcherOptions

**int32 branch\_and\_bound\_depth** Number of precomputed grids to use.

**int32 full\_resolution\_depth** Number of full resolution grids to use, additional grids will reduce the resolution by half each.

**double min\_rotational\_score** Minimum score for the rotational scan matcher.

**double min\_low\_resolution\_score** Threshold for the score of the low resolution grid below which a match is not considered. Only used for 3D.

**double linear\_xy\_search\_window** Linear search window in the plane orthogonal to gravity in which the best possible scan alignment will be found.

**double linear\_z\_search\_window** Linear search window in the gravity direction in which the best possible scan alignment will be found.

**double angular\_search\_window** Minimum angular search window in which the best possible scan alignment will be found.

## 1.19 cartographer.sensor.proto.AdaptiveVoxelFilterOptions

**float max\_length** 'max\_length' of a voxel edge.

**float min\_num\_points** If there are more points and not at least 'min\_num\_points' remain, the voxel length is reduced trying to get this minimum number of points.

**float max\_range** Points further away from the origin are removed.



Performing evaluation is a crucial part of developing a SLAM system. For this purpose, Cartographer offers built-in tools that can aid the tuning process or can be used for quality assurance purposes.

These tools can be used to assess the SLAM result even when no dedicated ground truth is available. This is in contrast to public SLAM benchmarks like e.g the KITTI dataset<sup>1</sup> or the TUM RGB-D dataset<sup>2</sup>, where highly-precise ground truth states (GPS-RTK, motion capture) are available as a reference.

## 2.1 Concept

The process comprises two steps:

1. auto-generation of “ground truth” relations
2. evaluation of the test data against the generated ground truth

The evaluation is based on the pose relations metric proposed in<sup>3</sup>. Rather than comparing the pose of a trajectory node directly to the corresponding ground truth pose, it compares the relative poses between two trajectory nodes in the probe data to the corresponding relation of two trajectory nodes in the ground truth trajectory.

In Cartographer, we can generate such ground truth relations from trajectories with loop closures. Let an optimized trajectory with loop closures be the input for the ground truth generation. We select the relations from loop closure constraints that satisfy the following criteria:

- `min_covered_distance`: Minimum covered distance in meters before a loop closure is considered a candidate for autogenerated ground truth.
- `outlier_threshold_meters`: Distance in meters beyond which constraints are considered outliers.
- `outlier_threshold_radians`: Distance in radians beyond which constraints are considered outliers.

<sup>1</sup> Andreas Geiger, Philip Lenz and Raquel Urtasun. *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. CVPR, 2012.

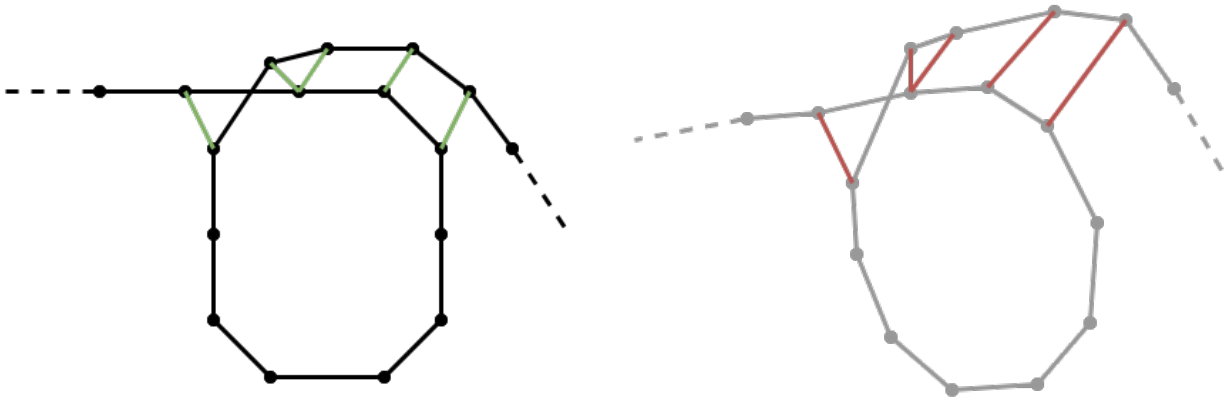
<sup>2</sup> Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard and Daniel Cremers. *A Benchmark for the Evaluation of RGB-D SLAM Systems*. IROS, 2012.

<sup>3</sup> Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss and Alexander Kleiner. *On measuring the accuracy of SLAM algorithms*. Autonomous Robots 27(4), pp.387-407, 2009.

We can assume the pose relations of neighboring trajectory nodes fulfilling these requirements to be locally correct in a fully optimized trajectory. Although this is not a ground truth in the sense of an independent input from another source, we can now use it to evaluate the quality of local SLAM results that were generated without loop closure optimization.

The following figure illustrates the concept. On the left side, the ground truth relations are visualized as green connections between trajectory nodes of a fully optimized trajectory. On the right side, the corresponding relations in a non-optimized trajectory are shown in red.

The actual metric that is computed is the difference between the ground truth (green) and the probe (red) relations.



## 2.2 Advantages & Limitations

The first obvious advantage is the easier data collection process compared to a cumbersome ground truth setup. Another great advantage of this methodology is that the SLAM system can be evaluated in any custom sensor configuration (compared to public benchmarks where we are restricted to the data and the sensor configuration of the authors).

However, this type of self-evaluation is not suitable for measuring the accuracy of the full SLAM system with all optimizations enabled - only an evaluation with *real* ground truth states can provide that. Furthermore, trajectory nodes outside of loop closure areas can't be considered.

## 2.3 How-To

Given a serialized state of a fully optimized trajectory (here: `optimized.pbstream` file), the ground truth relations can be generated with the following command:

```
cd <build> # (directory where Cartographer's binaries are located)
./cartographer_autogenerate_ground_truth -pose_graph_filename optimized.pbstream -
  ↳output_filename relations.pbstream -min_covered_distance 100 -outlier_threshold_
  ↳meters 0.15 -outlier_threshold_radians 0.02
```

Then, a non-optimized trajectory `test.pbstream` can be evaluated against the generated relations with:

```
./cartographer_compute_relations_metrics -relations_filename relations.pbstream -pose_
  ↳graph_filename test.pbstream
```

This will produce output in this form:

Abs translational error 0.01944 +/- 0.01819 m
Sqr translational error 0.00071 +/- 0.00189 m^2
Abs rotational error 0.11197 +/- 0.12432 deg
Sqr rotational error 0.02799 +/- 0.07604 deg^2

---

## 2.4 References





This documents a few common patterns that exist in the Cartographer codebase.

### 3.1 Frames

**global map frame** This is the frame in which global SLAM results are expressed. It is the fixed map frame including all loop closure and optimization results. The transform between this frame and any other frame can jump when new optimization results are available. Its z-axis points upwards, i.e. the gravitational acceleration vector points in the -z direction, i.e. the gravitational component measured by an accelerometer is in the +z direction.

**local map frame** This is the frame in which local SLAM results are expressed. It is the fixed map frame excluding loop closures and the pose graph optimization. For a given point in time, the transform between this and the global map frame may change, but the transform between this and all other frames does not change.

**submap frame** Each submap has a separate fixed frame.

**tracking frame** The frame in which sensor data is expressed. It is not fixed, i.e. it changes over time. It is also different for different trajectories.

**gravity-aligned frame** Only used in 2D. A frame colocated with the tracking frame but with a different orientation that is approximately aligned with gravity, i.e. the gravitational acceleration vector points approximately in the -z direction. No assumption about yaw (rotation around the z axis between this and the tracking frame) should be made. A different gravity-aligned frame is used for different trajectory nodes, e.g. yaw can change arbitrarily between gravity-aligned frames of consecutive nodes.

### 3.2 Transforms

**local\_pose** Transforms data from the tracking frame (or a submap frame, depending on context) to the local map frame.

**global\_pose** Transforms data from the tracking frame (or a submap frame, depending on context) to the global map frame.

**local\_submap\_pose** Transforms data from a submap frame to the local map frame.

**global\_submap\_pose** Transforms data from a submap frame to the global map frame.

### 4.1 Relative Transform Error 2D

Given two poses  $\mathbf{p}_i = [\mathbf{x}_i; \theta_i] = [x_i, y_i, \theta_i]^T$  and  $\mathbf{p}_j = [\mathbf{x}_j; \theta_j] = [x_j, y_j, \theta_j]^T$  the transformation  $\mathbf{T}$  from the coordinate frame  $j$  to the coordinate frame  $i$  has the following form

$$\mathbf{T}(\mathbf{p}_i, \mathbf{p}_j) = \begin{bmatrix} R(\theta_i)^T(\mathbf{x}_j - \mathbf{x}_i) \\ \theta_j - \theta_i \end{bmatrix}$$

where  $R(\theta_i)^T$  is the rotation matrix of  $\theta_i$ .

The weighted error  $f : \mathbb{R}^6 \mapsto \mathbb{R}^3$  between  $\mathbf{T}$  and the measured transformation  $\mathbf{T}_{ij}^m = [\mathbf{x}_{ij}^m; \theta_{ij}^m]$  from the coordinate frame  $j$  to the coordinate frame  $i$  can be computed as

$$\mathbf{f}_{\text{relative}}(\mathbf{p}_i, \mathbf{p}_j) = [w_t \ w_r] (\mathbf{T}_{ij}^m - \mathbf{T}(\mathbf{p}_i, \mathbf{p}_j)) = \begin{bmatrix} w_t (\mathbf{x}_{ij}^m - R(\theta_i)^T(\mathbf{x}_j - \mathbf{x}_i)) \\ w_r (\text{clamp}(\theta_{ij}^m - (\theta_j - \theta_i))) \end{bmatrix}$$

where  $w_t$  and  $w_r$  are weights for translation and rotation respectively and  $\text{clamp} : \mathbb{R} \mapsto [-\pi, \pi]$  normalizes the angle difference.

Jacobian matrix  $J_f$  is given by:

$$\begin{aligned} J_f(\mathbf{p}_i, \mathbf{p}_j) &= \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_i} & \frac{\partial \mathbf{f}}{\partial y_i} & \frac{\partial \mathbf{f}}{\partial \theta_i} & \frac{\partial \mathbf{f}}{\partial x_j} & \frac{\partial \mathbf{f}}{\partial y_j} & \frac{\partial \mathbf{f}}{\partial \theta_j} \end{bmatrix} \\ &= \begin{bmatrix} w_t R^T(\theta_i) & -w_t \frac{dR^T(\theta_i)}{d\theta}(\mathbf{x}_j - \mathbf{x}_i) & -w_t R^T(\theta_i) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & w_r & \mathbf{0}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \end{aligned} \quad (4.1)$$

### 4.2 Landmark Cost Function

Let  $\mathbf{p}_o$  denote the global pose of the SLAM tracking frame at which a landmark with the global pose  $\mathbf{p}_l$  is observed. The landmark observation itself is the measured transformation  $\mathbf{T}_{ol}^m$  that was observed at time  $t_o$ .

As the landmark can be observed asynchronously, the pose of observation  $\mathbf{p}_o$  is modeled in between two regular, consecutive trajectory nodes  $\mathbf{p}_i, \mathbf{p}_j$ . It is interpolated between  $\mathbf{p}_i$  and  $\mathbf{p}_j$  at the observation time  $t_o$  using a linear interpolation for the translation and a quaternion SLERP for the rotation:

$$\mathbf{p}_o = \text{interpolate}(\mathbf{p}_i, \mathbf{p}_j, t_o)$$

Then, the full weighted landmark cost function can be written as:

$$\begin{aligned} \mathbf{f}_{\text{landmark}}(\mathbf{p}_l, \mathbf{p}_i, \mathbf{p}_j) &= \mathbf{f}_{\text{relative}}(\mathbf{p}_l, \mathbf{p}_o) \\ &= [w_t \ w_r] (\mathbf{T}_{ol}^m - \mathbf{T}(\mathbf{p}_o, \mathbf{p}_l)) \end{aligned} \quad (4.4)$$

The translation and rotation weights  $w_t, w_r$  are part of the landmark observation data that is fed into Cartographer.

---

## Migration tool for pbstream files

---

With the update of the pbstream serialization format as discussed in [RFC-0021](#), previously serialized pbstream files are not loadable in Cartographer 1.0 anymore.

In order to enable users to reuse previously generated pbstream files, we provide a migration tool which converts pbstreams from Cartographer 0.3 to the new serialization format used in Cartographer 1.0.

The tool is shipped as part of Cartographer's pbstream tool ([source](#)) and once built can be invoked as follows::

```
cartographer_pbstream migrate old.pbstream new.pbstream
```

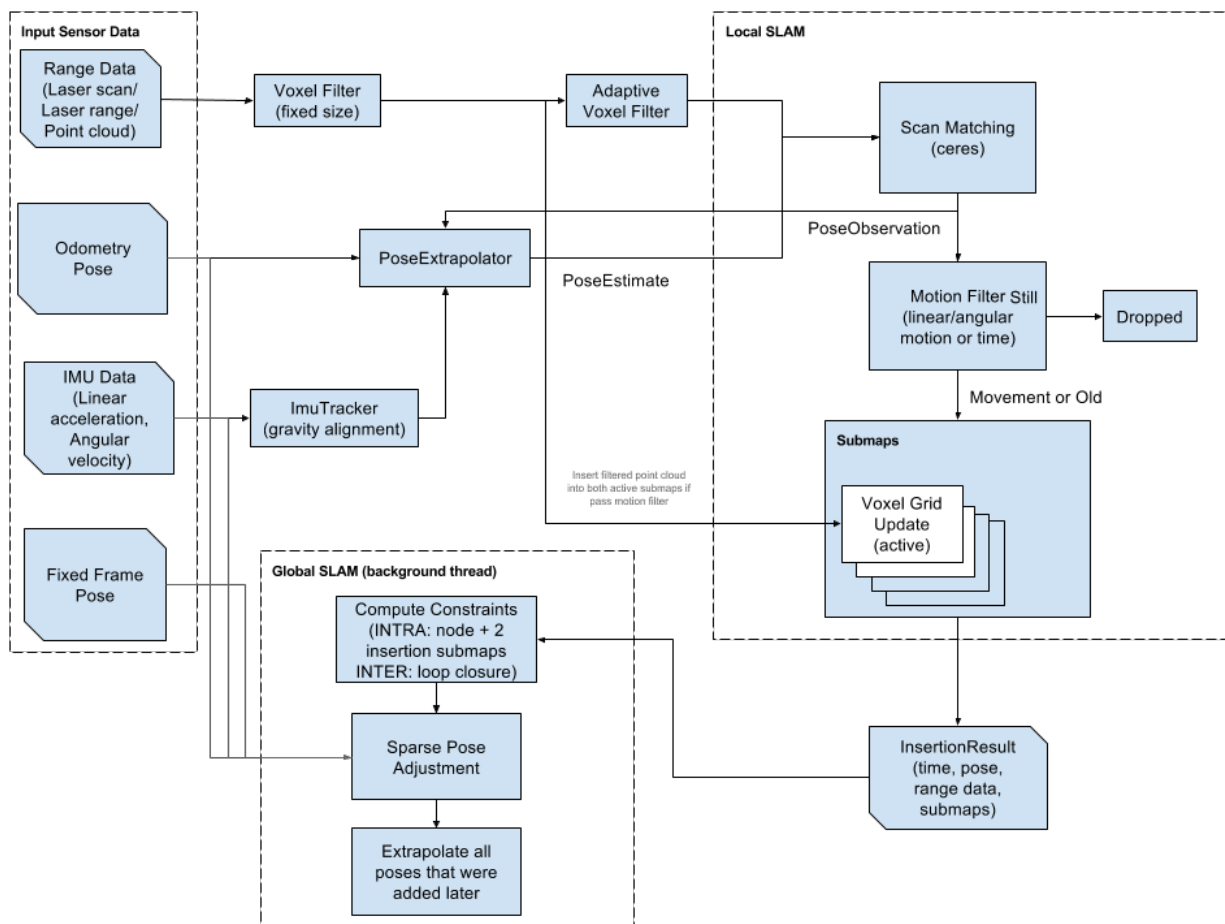
The tool assumes that the first pbstream provided as commandline argument, follows the serialization format of Cartographer 0.3. The resulting 1.0 pbstream will be saved to the second commandline argument location.

[Cartographer](#) is a system that provides real-time simultaneous localization and mapping ([SLAM](#)) in 2D and 3D across multiple platforms and sensor configurations.



## Technical Overview

- High level system overview of Cartographer







Cartographer is a standalone C++ library. To get started quickly, use our [ROS](#) integration.

### 7.1 Getting started with ROS

ROS integration is provided by the [Cartographer ROS repository](#). You will find complete documentation for using Cartographer with ROS at the [Cartographer ROS Read the Docs site](#).

### 7.2 Getting started without ROS

Please see our ROS integration as a starting point for integrating your system with the standalone library. Currently, it is the best available reference.

On Ubuntu 14.04 (Trusty):

```
# Install the required libraries that are available as debs.
sudo apt-get update

# Install CMake 3.2 for Ubuntu Trusty and Debian Jessie.
sudo apt-get install lsb-release -y
if [[ "$(lsb_release -sc)" = "trusty" ]]
then
    sudo apt-get install cmake3 -y
elif [[ "$(lsb_release -sc)" = "jessie" ]]
then
    sudo sh -c "echo 'deb http://ftp.debian.org/debian jessie-backports main' >> /etc/
↪apt/sources.list"
    sudo apt-get update
    sudo apt-get -t jessie-backports install cmake -y
else
    sudo apt-get install cmake -y
```

(continues on next page)

(continued from previous page)

```
fi

sudo apt-get install -y \
    g++ \
    git \
    google-mock \
    libboost-all-dev \
    libcairo2-dev \
    libeigen3-dev \
    libgflags-dev \
    libgoogle-glog-dev \
    liblua5.2-dev \
    libsuitesparse-dev \
    ninja-build \
    python-sphinx
```

```
VERSION="1.13.0"

# Build and install Ceres.
git clone https://ceres-solver.googlesource.com/ceres-solver
cd ceres-solver
git checkout tags/${VERSION}
mkdir build
cd build
cmake .. -G Ninja -DCXX11=ON
ninja
CTEST_OUTPUT_ON_FAILURE=1 ninja test
sudo ninja install
```

```
VERSION="v3.4.1"

# Build and install proto3.
git clone https://github.com/google/protobuf.git
cd protobuf
git checkout tags/${VERSION}
mkdir build
cd build
cmake -G Ninja \
    -DCMAKE_POSITION_INDEPENDENT_CODE=ON \
    -DCMAKE_BUILD_TYPE=Release \
    -Dprotobuf_BUILD_TESTS=OFF \
    ../cmake
ninja
sudo ninja install
```

```
# Build and install Cartographer.
cd cartographer
mkdir build
cd build
cmake .. -G Ninja
ninja
CTEST_OUTPUT_ON_FAILURE=1 ninja test
sudo ninja install
```

---

### System Requirements

---

Although Cartographer may run on other systems, it is confirmed to be working on systems that meet the following requirements:

- 64-bit, modern CPU (e.g. 3rd generation i7)
- 16 GB RAM
- Ubuntu 14.04 (Trusty) and 16.04 (Xenial)
- gcc version 4.8.4 and 5.4.0

### 8.1 Known Issues

- 32-bit builds have libeigen alignment problems which cause crashes and/or memory corruptions.



## CHAPTER 9

---

### How to cite us

---

Background about the algorithms developed for Cartographer can be found in the following publication. If you use Cartographer for your research, we would appreciate it if you cite our paper.

W. Hess, D. Kohler, H. Rapp, and D. Andor, [Real-Time Loop Closure in 2D LIDAR SLAM](#), in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016. pp. 1271–1278.