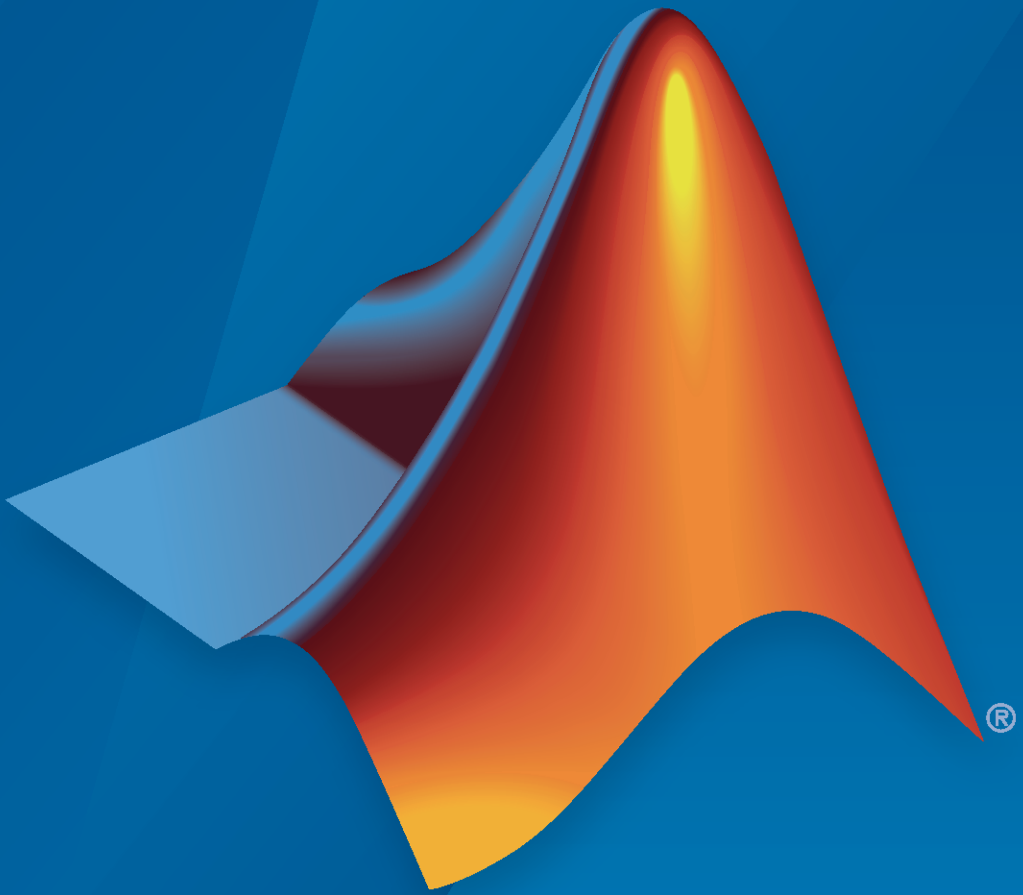
Lidar Toolbox™ Release Notes



MATLAB®

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
Phone:	508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Lidar Toolbox™ Release Notes

Contents

R2021a

Lidar Camera Calibrator: Interactively calibrate lidar and camera sensors to estimate rigid transformation using an app

The **Lidar Camera Calibrator** app estimates the rigid transformation between a lidar and camera sensors. Use the app to interactively perform the “Lidar and Camera Calibration” workflow. The app uses the underlying features of the workflow.

- Detect, extract, and visualize checkerboard features from image and point cloud data.
- Estimate the rigid transformation between the lidar and camera using feature detection results.
- Use calibration results to fuse data from both the sensors. You can visualize point cloud data projected onto the images, and color or grayscale information from the images fused with point cloud data.
- View the plotted calibration error metrics. You can remove outliers using a threshold line and recalibrate the remaining data.
- Define a region of interest (ROI) around the checkerboard to reduce the computation resources required by the transformation estimation process.
- Export the transformation and error metric data as workspace variables or MAT files. You can also create a MATLAB® script for the entire workflow.

Lidar Labeler Enhancements: Define a custom point cloud reader and a region of interest in the point cloud

The following table describes enhancements for these labeling apps:

- **Image Labeler**
- **Video Labeler**
- **Ground Truth Labeler**
- **Lidar Labeler**

Enhancement	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler
Label distinct instances of objects belonging to the same class using a polygon label. For more details, see “Label Objects Using Polygons”.	Yes	Yes	Yes	No
Use superpixel automation to quickly pixel label regions of an image with similar pixel values. For more details, see “Label Pixels Using Superpixel Tool”.	Yes	Yes	Yes	No
Automate the labeling of multiple signals together within a single automation run. For an example, see “Automate Ground Truth Labeling Across Multiple Signals” (Automated Driving Toolbox).	No	No	Yes	No

Enhancement	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler
Label very large images (with at least one dimension <8K) that previously could not be loaded into memory. Load these images as blocked images. For more details, see “Label Large Images in Image Labeler”.	Yes	No	No	No
Use a custom reader function to import any point cloud. For more details, see “Use Custom Point Cloud Source Reader for Labeling”.	No	No	No	Yes
Define and view a region of interest (ROI) in the point cloud and label objects in it. For more details, see “ROI View”.	No	No	No	Yes
Control the point dimension of the point cloud.	No	No	No	Yes

The **Lidar Labeler** enables you to create a custom algorithm to automate the labeling process. “Automate Ground Truth Labeling For Vehicle Detection Using PointPillars” workflow shows you how to create and apply such an algorithm. It uses a pre-trained PointPillars network to create a custom algorithm. To learn more about PointPillars, see “Lidar 3-D Object Detection Using PointPillars Deep Learning”.

Aerial Lidar Processing: Segment terrain in point cloud data

The `segmentGroundSMRF` function segments the input point cloud into ground and non-ground points using a simple morphological filter (SMRF). You can adjust the parameters to segment various types of terrain or steep slopes. The “Terrain Classification for Aerial Lidar Data” workflow uses this feature to segment ground, vegetation, and buildings in aerial point clouds.

The “Aerial Lidar Semantic Segmentation Using PointNet++ Deep Learning” workflow uses the PointNet++ network to perform semantic segmentation on aerial lidar data.

Bounding Box Transfer: Transfer bounding boxes from the lidar coordinate frame to the camera coordinate frame

The `bboxLidarToCamera` function estimates a 2-D bounding box in a camera coordinate frame using a 3-D bounding box in a lidar coordinate frame.

Multiclass Object Detection and Segmentation: Multiclass support for PointPillars and SqueezeSegV2 networks

The PointPillars and SqueezeSegV2 networks pre-trained on the PandaSet multiclass dataset are now available. For more information, see these examples:

- Lidar Point Cloud Semantic Segmentation Using SqueezeSegV2 Deep Learning Network

- Lidar 3-D Object Detection Using PointPillars Deep Learning

Unorganized to Organized Conversion: Convert unorganized point clouds to organized point clouds

The “Unorganized to Organized Conversion of Point Clouds Using Spherical Projection” example shows how to convert unorganized point clouds to organized format using spherical projection.

Deep Learning Example: Data augmentations for object detection networks

The “Data Augmentations for Lidar Object Detection Using Deep Learning” example shows typical data augmentation techniques used in 3-D object detection workflows with lidar data.

Lane Detection Example: Lane detection in lidar data

The “Lane Detection in 3-D Lidar Point Cloud” example shows how to detect lanes in 3-D lidar point clouds. It involves detection of the immediate left and right lanes, also known as ego vehicle lanes, with respect to the lidar sensor.

Lidar Simultaneous Localization and Mapping (SLAM): Support for loop closure detection and localization

Extract eigenvalue-based features from point cloud segments using the `extractEigenFeatures` function. The features are returned as a vector of `eigenFeature` objects. You can use these geometrical features for loop closure detection and localization in a target map.

The `pcmapsegmatch` object creates a map of segments and features for loop closure detection and localization using the segment matching (SegMatch) place recognition algorithm.

Use the new syntax for `pcshowMatchedFeatures` to visualize the segment matches.

The “Build Map and Localize Using Segment Matching” example shows how to build a map with lidar data and localize the position of a vehicle on the map using SegMatch.

2-D SLAM Example: Map building from 2-D lidar scans

The “Build Map from 2-D Lidar Scans Using SLAM” example shows how to implement the simultaneous localization and mapping (SLAM) algorithm on a series of 2-D lidar scans using scan processing algorithms and pose graph optimization (PGO). Use this example to estimate the trajectory of the robot and build a map of the environment.

Code Generation Support: Generate C/C++ code using MATLAB Coder

These functions now support code generation:

- `projectLidarPointsOnImage`
- `fuseCameraToLidar`

- `bboxCameraToLidar`

The “Code Generation For Lidar Object Detection Using PointPillars Deep Learning” shows how to generate CUDA® MEX for a PointPillars object detector with custom layers.

R2020b

Version: 1.0

New Features

Lidar Labeler App: Interactive, semi-automated, and custom automated labeling of lidar point clouds

The **Lidar Labeler** app enables you to label objects in a point cloud or a point cloud sequence. The app reads point cloud data from PLY, PCAP, LAS, LAZ, and PCD files. Using the app, you can:

- Define cuboid region of interest (ROI) labels. Use them to interactively label your ground truth data.
- Assign attributes to labels, and use them to further define the labels.
- Use built-in algorithms for clustering, ground plane segmentation, automated labelling, and tracking.
- Save label definitions, point cloud data, and ground truth data to a session file for future use.
- Import custom automation algorithms for automated labeling.
- Evaluate automation algorithm performance using the visual summary.
- Export the labeled ground truth as a `groundTruthLidar` object. You can use this object for system verification or training an object detector.

Lidar-Camera Calibration: Calibrate lidar and camera sensors to estimate cross-sensor coordinate transform

Use the lidar and camera calibration (LCC) workflow to estimate the rigid transformation between a lidar sensor and a camera. The workflow uses the checkerboard pattern calibration method. Lidar Toolbox™ introduces three new features to carry out this workflow:

1. `estimateCheckerboardCorners3d` – Estimate the world frame coordinates of the checkerboard corner points in an image.
2. `detectRectangularPlanePoints` – Detect a rectangular plane of the specified dimensions in a point cloud.
3. `estimateLidarCameraTransform` – Estimate the rigid transformation from a lidar sensor to a camera.

The toolbox also contains features to facilitate downstream applications:

1. `projectLidarPointsOnImage` – Project lidar point cloud data onto an image coordinate frame.
2. `fuseCameraToLidar` – Fuse color or grayscale information from an image to a point cloud.
3. `bboxCameraToLidar` – Estimate 3-D bounding boxes in a point cloud from 2-D bounding boxes in an image.

The Lidar and Camera Calibration example shows how to calibrate lidar and camera sensors by estimating the rigid transformation from the lidar sensor to the camera.

The Detect Vehicles in Lidar Using Image Labels example shows how to detect vehicles in lidar point cloud data using the detections from an image scene and the rigid transformation between the camera and lidar sensor.

Deep Learning for Lidar Point Cloud Processing: Use deep learning networks to detect and segment objects in lidar point cloud data

The Lidar Point Cloud Semantic Segmentation Using PointSeg Deep Learning Network example shows how to perform semantic segmentation of road objects in a highway traffic scene using a PointSeg deep learning network.

The Lidar Point Cloud Semantic Segmentation Using SqueezeSegV2 Deep Learning Network example shows how to perform semantic segmentation of organized lidar point cloud data using a SqueezeSegV2 deep learning network. You can use the `squeezesegv2Layers` function to create a the SqueezeSegV2 deep learning network.

The Code Generation for Lidar Point Cloud Segmentation Network example shows how to generate CUDA MEX code for a pretrained SqueezeSegV2 deep learning network that can segment lidar point cloud data.

The Lidar 3-D Object Detection Using PointPillars Deep Learning example shows how to detect objects in 3-D point cloud data using a PointPillars deep learning network.

Shape Fitting: Fit shape and track detected objects in a lidar point cloud sequence

Use the `pcfitcuboid` function to fit a cuboid bounding box over detected objects in organized point cloud data using the L-shape fitting algorithm.

You can fit cuboid bounding boxes and track detected objects in an organized lidar point cloud sequence using the `pcfitcuboid` function for shape fitting and a joint probabilistic data association (JPDA) tracker for tracking. For more information, see Detect, Classify, and Track Vehicles Using Lidar and Track Vehicles Using Lidar: From Point Cloud to Track List.

Feature Matching: Extract and match lidar point cloud features

Extract fast point feature histogram (FPFH) descriptors from point cloud data using the `extractFPFHFeatures` function. You can compare point descriptors (features) of different point clouds to check for correspondence between the two point clouds. Use the `pcmatchfeatures` function to find the matching points between two different point clouds and display the matched points by using `pcshowMatchedFeatures` function.

You can use feature extraction and matching to develop feature-based registration workflows for map building. For more information, see Feature-Based Map Building from Lidar Data.

The Aerial Lidar SLAM Using FPFH Descriptors example shows how to develop a 3-D simultaneous localization and mapping (SLAM) algorithm for aerial lidar data by using FPFH descriptors for feature-based registration.

2-D Lidar Processing: Simulate and process 2-D laser scan data and estimate the pose between two scans

You can simulate 2-D lidar sensors and sensor readings (scans), match scans, and estimate the pose between two scans. These features can be used in a 2-D object detection workflows.

The Collision Warning Using 2-D Lidar example shows how to detect obstacles and warn possible collisions using 2-D lidar scan data for automated guided vehicles.

Velodyne LiDAR Streaming: Connect and stream lidar point clouds from Velodyne LiDAR sensors

Support for Velodyne LiDAR® sensors is available through Lidar Toolbox Support Package for Velodyne LiDAR Sensors. You can connect to and stream point clouds from these Velodyne LiDAR sensor models:

- VLS-128 Alpha Puck
- VLP-32C Ultra Puck
- VLP-16 Puck Hi-Res
- VLP-16 Puck LITE
- VLP-16 Puck
- HDL-32E
- HDL-64E

You can preview and read point clouds from the supported lidar sensors by using the `velodynelidar` (Lidar Toolbox Support Package for Velodyne LiDAR Sensors) object.

For information about using Velodyne LiDAR sensors in MATLAB, see Lidar Toolbox Support Package for Velodyne LiDAR Sensors documentation.

Lidar File Readers: Support for Ibeo sensor, LAS, and LAZ file formats

Use the `ibeoLidarReader` object to read lidar point cloud data from Ibeo data container (IDC) files.

Use the `lasFileReader` object to read lidar point cloud data from LAS or LAZ files.