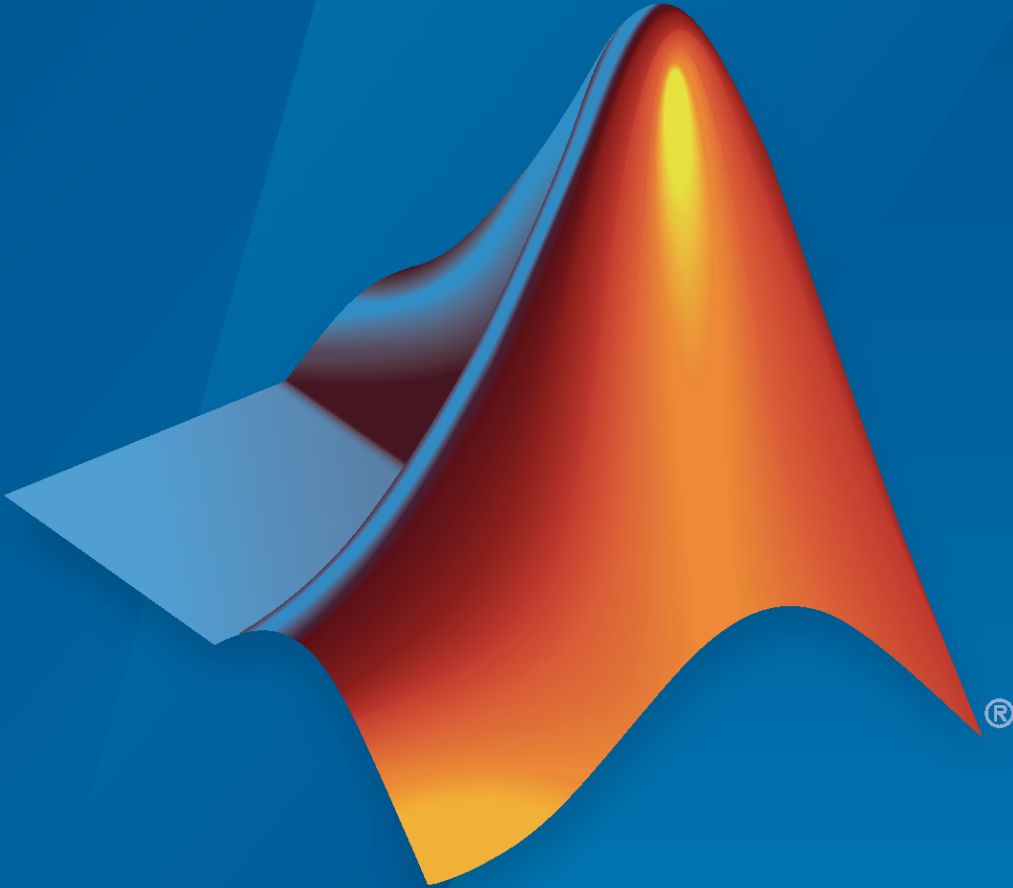


Lidar Toolbox™ Release Notes



MATLAB®



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://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*

© COPYRIGHT 2020–2023 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

## R2023a

<b>Lidar File Readers: Support for E57 files and enhancements to ousterFileReader</b> .....	1-2
<b>Lidar Sensor: Generate point cloud data by simulating a lidar sensor</b> ...	1-2
<b>Surface Mesh: Perform mesh smoothing, cluster connected mesh faces</b> .....	1-2
<b>removeHiddenPoints: Remove points hidden from a view in point cloud</b> .....	1-2
<b>Motion Compensation: Undistort point cloud affected due to ego motion</b> .....	1-2
<b>pointPillarsObjectDetector object: Support for hardware execution environment</b> .....	1-3
<b>Lidar Viewer Enhancements: Simultaneous views, updated edit and export options</b> .....	1-3
<b>Lidar Labeler Enhancements: Updated voxel and plane labeling</b> .....	1-3
<b>Lidar Examples: Explore new lidar workflows</b> .....	1-4
<b>Code Generation Support: Perform GPU and C/C++ code generation</b> ...	1-4
<b>Performance Improvement: detect object function of pointPillarsObjectDetector</b> .....	1-4
<b>MATLAB Online Support</b> .....	1-5

## R2022b

<b>3-D Surface Mesh: Create, read, write, and visualize surface mesh</b> .....	2-2
<b>pc2surfaceMesh: Convert point cloud data into 3-D surface mesh</b> .....	2-2
<b>Simultaneous Localization and Mapping (SLAM): Create 2-D occupancy map using lidar scan data</b> .....	2-2

<b>Curb Segmentation: Segment curb points from point cloud</b> .....	2-2
<b>Point Cloud Registration: Register two point clouds using fast global registration</b> .....	2-2
<b>Lidar Odometry and Mapping (LOAM): Create map from LOAM feature points</b> .....	2-2
<b>Lidar File Readers: Enhancements to lasFileReader</b> .....	2-2
<b>Lidar File Writers: Enhancements for lasFileWriter</b> .....	2-3
<b>lidarPointAttributes: WaveformData property has additional fields</b> .....	2-3
<b>Ouster Lidar Support: Stream point cloud data from Ouster 128 channel sensor models</b> .....	2-3
<b>Deep Learning Layers: Create point cloud input layer</b> .....	2-3
<b>Deep Learning Segmentation: Support for point cloud segmentation network, and segmentation in aerial lidar data</b> .....	2-3
<b>trainPointPillarsObjectDetector: Support for Experiment Manager app</b> .....	2-4
<b>Lidar Viewer Enhancements: Support for nonlinear colormaps, keyboard shortcuts</b> .....	2-4
<b>Lidar Labeler Enhancements</b> .....	2-4
Lidar Labeler Enhancements: Support for color data in point clouds, updated views .....	2-4
Functionality being removed or changed .....	2-5
<b>Lidar Examples: Motion compensation using sensor fusion</b> .....	2-5
<b>Functionality being removed or changed</b> .....	2-5
Several functions use new premultiply matrix convention .....	2-5
The readVLR function of the lasFileReader object returns both raw binary data and VLR data .....	2-6
The LaserReturns argument in the readPointCloud function of the lasFileReader object has been renamed to LaserReturn .....	2-6
The WaveformData property of the lidarPointAttributes object has a new field order .....	2-6

**R2022a**

<b>Lidar Sensor Model: Simulate lidar sensor and generate point cloud data</b> .....	3-2
--	-----

<b>Blocked Point Cloud: Process 3-D point clouds too large to fit in memory</b> .....	3-2
<b>Lidar Odometry and Mapping (LOAM): Detect and register LOAM feature points from lidar data</b> .....	3-2
<b>Lidar File Readers: Support for Ouster sensor, Hesai sensor, and enhancements to lasFileReader</b> .....	3-2
<b>Data Augmentation: Lidar point cloud ground truth data augmentation for object detection</b> .....	3-3
<b>Deep Learning Object Detectors: Create training data to train neural networks for point cloud object detection</b> .....	3-3
<b>Point Cloud to Scan Conversion: Convert 3-D lidar point cloud to 2-D lidar scan</b> .....	3-3
<b>Intrinsic Shape Signature (ISS) Features: Detect ISS feature points from point cloud data</b> .....	3-3
<b>Ouster Lidar Support: Stream point cloud data from Ouster sensor</b> .....	3-3
<b>Lidar Viewer Enhancements: Edit multiple point cloud frames, measure point cloud data</b> .....	3-4
<b>Lidar Labeler Enhancements: Create training data for object detection</b> .....	3-4
<b>Lidar Camera Calibrator Enhancements: Support for 16-bit intensity images</b> .....	3-4
<b>lidarPointAttributes Object: Store lidar point attributes</b> .....	3-5
<b>Lidar File Writer: Write point cloud data into LAS or LAZ format</b> .....	3-5
<b>estimateLidarCameraTransform Function: Compute transform with 2-D image corners</b> .....	3-5
<b>lidarParameters Object: Support for VLS-128, VLP-32C devices</b> .....	3-5
<b>Lidar Examples: Explore new lidar workflows</b> .....	3-5
<b>Deep Learning Examples: Explore new deep learning workflows</b> .....	3-5
<b>Code Generation Support: Perform GPU and C/C++ code generation</b> ...	3-6
Generate CUDA code using GPU Coder .....	3-6
Generate C/C++ code using MATLAB Coder .....	3-6
<b>Performance Improvement: matchScansGrid function</b> .....	3-6
<b>Functionality being removed or changed</b> .....	3-6
The readPointCloud function of the lasFileReader object returns a lidarPointAttributes object .....	3-6

The LaserReturns argument in the readPointCloud function of the lasFileReader object has been renamed to LaserReturn . . . . .	3-7
The CameraIntrinsic argument in the estimateLidarCameraTransform function is no longer required . . . . .	3-7

## R2021b

<b>Lidar Viewer: Visualize and analyze lidar data using app . . . . .</b>	<b>4-2</b>
<b>Unorganized to Organized Conversion: Convert unorganized point clouds to organized point clouds . . . . .</b>	<b>4-2</b>
<b>Object Detection Interface: Train and test PointPillars network to detect objects in lidar point clouds . . . . .</b>	<b>4-2</b>
<b>PointNet++ Network: Create PointNet++ network for segmentation . . . . .</b>	<b>4-2</b>
<b>Digital Elevation Model: Create digital elevation model of point cloud data . . . . .</b>	<b>4-2</b>
<b>Lidar Labeler Enhancements: SMRF ground segmentation algorithm . . . . .</b>	<b>4-3</b>
<b>Lidar Camera Calibrator Enhancements: Manually select checkerboard points for calibration . . . . .</b>	<b>4-3</b>
<b>Deep Learning Example: Code generation for PointNet++ network . . . . .</b>	<b>4-4</b>
<b>Lidar Examples: Explore new lidar workflows . . . . .</b>	<b>4-4</b>
<b>Code Generation Support: Generate C/C++ code using MATLAB Coder . . . . .</b>	<b>4-4</b>
<b>Performance Improvement: matchScans function . . . . .</b>	<b>4-4</b>

## R2021a

<b>Lidar Camera Calibrator: Interactively calibrate lidar and camera sensors to estimate rigid transformation using an app . . . . .</b>	<b>5-2</b>
<b>Lidar Labeler Enhancements: Define a custom point cloud reader and a region of interest in the point cloud . . . . .</b>	<b>5-2</b>
<b>Aerial Lidar Processing: Segment terrain in point cloud data . . . . .</b>	<b>5-3</b>
<b>Bounding Box Transfer: Transfer bounding boxes from the lidar coordinate frame to the camera coordinate frame . . . . .</b>	<b>5-3</b>

<b>Multiclass Object Detection and Segmentation: Multiclass support for PointPillars and SqueezeSegV2 networks</b> .....	5-3
<b>Unorganized to Organized Conversion: Convert unorganized point clouds to organized point clouds</b> .....	5-4
<b>Deep Learning Example: Data augmentations for object detection networks</b> .....	5-4
<b>Lane Detection Example: Lane detection in lidar data</b> .....	5-4
<b>Lidar Simultaneous Localization and Mapping (SLAM): Support for loop closure detection and localization</b> .....	5-4
<b>2-D SLAM Example: Map building from 2-D lidar scans</b> .....	5-4
<b>Code Generation Support: Generate C/C++ code using MATLAB Coder</b> .....	5-4

## R2020b

<b>Lidar Labeler App: Interactive, semi-automated, and custom automated labeling of lidar point clouds</b> .....	6-2
<b>Lidar-Camera Calibration: Calibrate lidar and camera sensors to estimate cross-sensor coordinate transform</b> .....	6-2
<b>Deep Learning for Lidar Point Cloud Processing: Use deep learning networks to detect and segment objects in lidar point cloud data</b> . . . .	6-3
<b>Shape Fitting: Fit shape and track detected objects in a lidar point cloud sequence</b> .....	6-3
<b>Feature Matching: Extract and match lidar point cloud features</b> .....	6-3
<b>2-D Lidar Processing: Simulate and process 2-D laser scan data and estimate the pose between two scans</b> .....	6-3
<b>Velodyne LiDAR Streaming: Connect and stream lidar point clouds from Velodyne LiDAR sensors</b> .....	6-4
<b>Lidar File Readers: Support for Ibeo sensor, LAS, and LAZ file formats</b> .....	6-4





# R2023a

---

**Version: 2.3**

**New Features**

**Bug Fixes**

## **Lidar File Readers: Support for E57 files and enhancements to `ousterFileReader`**

- Read point cloud data from an E57 file by using the `e57FileReader` object. You can also read coordinate reference system (CRS) data from the file.
- The `ousterFileReader` object has these enhancements.
  - The object has three new properties.
    - `ReturnMode` — Return modes in Ouster® PCAP file
    - `FirmwareVersion` — Firmware version of the Ouster sensor
    - `LidarUDPPProfile` — UDP profile configuration of the data packets
  - You can specify which return mode to read from the Ouster PCAP file by using the `ReadMode` argument of the `readFrame` object function.

## **Lidar Sensor: Generate point cloud data by simulating a lidar sensor**

Use the Lidar Sensor Simulink® block to generate point cloud data by simulating a lidar sensor. You can configure the sensor, as well as the actors, and simulate weather conditions using the block parameters.

You can now simulate weather conditions such as fog and rain by using the `lidarSensor` object also.

## **Surface Mesh: Perform mesh smoothing, cluster connected mesh faces**

Use the `smoothSurfaceMesh` function to perform a mesh smoothing operation using the average, Laplacian, or Taubin method.

The `clusterConnectedFaces` function clusters the connected triangular faces of the input surface mesh.

## **removeHiddenPoints: Remove points hidden from a view in point cloud**

Use the `removeHiddenPoints` function to remove points hidden from a specified point of view in the point cloud.

## **Motion Compensation: Undistort point cloud affected due to ego motion**

Use the `undistortEgoMotion` function to compensate point cloud distortion due to sensor motion. The function assumes that the sensor moves at a constant speed while recording the data.

---

## pointPillarsObjectDetector object: Support for hardware execution environment

You can now specify the hardware resource to process the point clouds by using the ExecutionEnvironment name-value argument in the detect object function of the pointPillarsObjectDetector object.

## Lidar Viewer Enhancements: Simultaneous views, updated edit and export options

The **Lidar Viewer** app has these enhancements:

- The app interface has been updated for a better user experience with easy access to tools, improved visualization, and performance.
- You can view and compare multiple point clouds using an overlay.
- Use the **History** pane of the **Edit Point Cloud** tab to delete or move your edit operations up and down. You can also update the parameters of your previous edits.
- You can now export point clouds as LAS or LAZ files.

## Lidar Labeler Enhancements: Updated voxel and plane labeling

This table describes enhancements for these labeling apps:

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

Feature	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler	Medical Image Labeler
New <b>Image Labeler</b> interface.	Yes	No	No	No	No
Use the <b>Point</b> ROI label to mark one or more keypoints in objects.	Yes	Yes	Yes	No	No
New naming scheme for pixel label file names when you export ground truth data. For more details, see the “Pixel Label File Naming: Unique filenames for image files related to pixel labels” release note.	Yes	Yes	Yes	No	No
Use the <b>Smart Voxel</b> tool to refine point cloud labeling by marking the foreground and background within a region of interest.	No	No	No	Yes	No

Feature	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler	Medical Image Labeler
Label a planar region within a point cloud by selecting any three points on the plane.	No	No	No	Yes	No
Configure, preview, and export 2-D and 3-D animations of medical image data.	No	No	No	No	Yes
View and navigate 2-D slices of a medical volume using crosshair navigation.	No	No	No	No	Yes

## Lidar Examples: Explore new lidar workflows

The “Determine Safe Landing Area for Aerial Vehicles” example shows how to identify a safe landing region for aerial vehicles such as helicopters and UAVs.

The “Export Point Cloud Segmentations to KML Files” example shows how to export point cloud data to a Keyhole Markup Language (KML) file. The example also shows how to read and visualize KML data.

The “Transform Point Cloud Using Lidar Viewer” example shows how to transform point clouds by using a custom preprocessing algorithm in the **Lidar Viewer** app.

## Code Generation Support: Perform GPU and C/C++ code generation

- These functions now support C/C++ code generation.
  - LOAMPoints
  - detectLOAMFeatures
  - pcregisterloam
  - pemaploam
- These functions now support both GPU and C/C++ code generation.
  - pointCloudInputLayer

## Performance Improvement: detect object function of pointPillarsObjectDetector

The `detect` object function of the `pointPillarsObjectDetector` object shows a reduction in execution time. For example, this code is about 2x times faster than the previous release with GPU environment.

```
function timingTest
% Load a pretrained PointPillars object detector
pretrainedDetector = load("pretrainedPointPillarsDetector.mat", ...
    "detector");
```

---

```
detector = pretrainedDetector.detector;  
  
% Read the input point cloud  
ptCloud= pcread('highwayScene.pcd');  
  
% Run the detector on the point cloud  
gputimeit(@() detect(detector,ptCloud))  
  
end
```

The approximate execution times are:

- **R2022b:** 0.11 s
- **R2023a:** 0.05 s

The code was timed on a Linux® 10, AMD® EPYC 7313 16-core processor system with NVIDIA® RTX A5000.

## **MATLAB Online Support**

MATLAB® Online™ now supports Lidar Toolbox™ software. For information about the product limitations, see Specifications and Limitations.



# R2022b

---

**Version: 2.2**

**New Features**

**Compatibility Considerations**

### **3-D Surface Mesh: Create, read, write, and visualize surface mesh**

Use the `surfaceMesh` object to construct a 3-D surface mesh by specifying vertices and faces.

You can read and write surface mesh data by using the `readSurfaceMesh` and `writeSurfaceMesh` functions, respectively. These functions support the STL and PLY formats.

To visualize the surface mesh, use the `surfaceMeshShow` function.

### **pc2surfaceMesh: Convert point cloud data into 3-D surface mesh**

To convert point cloud data into a 3-D surface mesh, use the `pc2surfacemesh` function.

### **Simultaneous Localization and Mapping (SLAM): Create 2-D occupancy map using lidar scan data**

Use the `lidarscanmap` object to build a map of an environment by using the scan data from a vehicle-mounted lidar sensor as the vehicle moves through the environment.

You can tune the scan matching and loop closure detection parameters of the SLAM algorithm by using the `detectLoopClosure` function. The algorithm generates a `poseGraph` and continuously optimizes edge constraints based on detected loop closures. As the algorithm detects more loop closures, you can continuously adjust for odometry drift while building a map of your environment.

### **Curb Segmentation: Segment curb points from point cloud**

Use the `segmentCurbPoints` function to segment curb points from a point cloud captured on a road. You can improve segmentation results by providing the road angles as an input to the function. Compute road angles by using the `detectRoadAngles` function.

### **Point Cloud Registration: Register two point clouds using fast global registration**

Compute the rigid transformation between point clouds, based on FPFH features, with the fast global registration (FGR) algorithm by using the `pcregisterfgr` function.

### **Lidar Odometry and Mapping (LOAM): Create map from LOAM feature points**

Use the `pcmaploam` object to create a map from the LOAM feature points detected in a point cloud. You can also use this object to optimize the absolute poses and visualize the map.

### **Lidar File Readers: Enhancements to lasFileReader**

The `lasFileReader` object has these enhancements.

- Support for color values of data type `uint16`.



- 
- You can now specify a user ID to the readVLR function to read variable length records (VLR) from the LAS file.

## **Lidar File Writers: Enhancements for lasFileWriter**

The `lasFileWriter` object has these enhancements.

- Support for point data record formats from 0 to 10.
- Support for color values of data type `uint16`.
- Use the `addVLR` function to write VLR, coordinate reference system (CRS), and waveform data into the LAS file.

## **lidarPointAttributes: WaveformData property has additional fields**

The `WaveformData` property of the `lidarPointAttributes` object has these additional structure fields.

- `WavePacketDescriptorIndex`
- `ByteOffsetToWaveformPacketData`
- `WaveformPacketSize`

## **Ouster Lidar Support: Stream point cloud data from Ouster 128 channel sensor models**

You can now stream data from these Ouster 128 channel device models using the Lidar Toolbox Support Package for Ouster Lidar Sensors.

- OS0-128
- OS1-128
- OS2-128

For more information, see the Lidar Toolbox Support Package for Ouster Lidar Sensors documentation.

## **Deep Learning Layers: Create point cloud input layer**

Create a point cloud input layer that inputs 3-D point clouds to a deep learning network by using the `pointCloudInputLayer` function.

## **Deep Learning Segmentation: Support for point cloud segmentation network, and segmentation in aerial lidar data**

Use the `pcsemanticseg` function to perform semantic segmentation on a point cloud.

Segment buildings and vegetation from aerial lidar data by using the `segmentAerialLidarBuildings` and `segmentAerialLidarVegetation` functions, respectively.

## trainPointPillarsObjectDetector: Support for Experiment Manager app

You can now specify the `ExperimentManager` value by using an `experiments.Monitor` (Deep Learning Toolbox) object as a name-value argument to the `trainPointPillarsObjectDetector` function. You can use the `experiments.Monitor` object with the **Experiment Manager** (Deep Learning Toolbox) app.

## Lidar Viewer Enhancements: Support for nonlinear colormaps, keyboard shortcuts

The **Lidar Viewer** app has these enhancements:

- You can now vary the colormap of the point cloud on a nonlinear scale, such as logarithmic or exponential. You can also create a custom colormap.
- The app supports various keyboard shortcuts. You can pan, zoom, and rotate point cloud data using these shortcuts.

## Lidar Labeler Enhancements

### Lidar Labeler Enhancements: Support for color data in point clouds, updated views

This table describes enhancements for these labeling apps:

- **Image Labeler**
- **Video Labeler**
- **Ground Truth Labeler** (Automated Driving Toolbox)
- **Lidar Labeler** (Lidar Toolbox)
- **Medical Image Labeler** (Medical Imaging Toolbox) — Introduced in R2022b

Feature	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler	Medical Image Labeler
The <code>cuboid2img</code> function returns rectangular projected cuboids to create data compatible with the labeling apps.	Yes	Yes	Yes	No	No
Use a function interface with labeler app to create a custom automation algorithm. For more details, see <a href="#">Create Automation Algorithm Function for Labeling</a>	Yes	Yes	Yes	No	No
Automate projected cuboid labeling with the temporal interpolator automation algorithm.	No	Yes	Yes	No	No

Feature	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler	Medical Image Labeler
The <code>boxLabelDatastore</code> object, the <code>gatherLabelData</code> (Automated Driving Toolbox), and the <code>objectDetectorTrainingData</code> object, used in creating training data for object detection, now support 2-D projected cuboid labels.	Yes	Yes	Yes	No	No
Visualize the color information of the point cloud using the updated <code>colormap</code> .	No	No	No	Yes	No
Add background color for the point cloud.	No	No	Yes	Yes	No
Visualize the <i>XY</i> , <i>YZ</i> , and <i>ZX</i> views of the point cloud.	No	No	Yes	Yes	No
Label 2-D and 3-D medical images for semantic segmentation.	No	No	No	No	Yes

### Functionality being removed or changed

#### Keyboard shortcuts to pan across a point cloud sequence has changed

*Behavior change*

Starting from R2022b, use **a**, **d**, **w**, and **s** as keyboard shortcuts for **Ground Truth Labeler** (Automated Driving Toolbox) and **Lidar Labeler** apps to pan across a point cloud frame.

Action	Keyboard Shortcut
Pan forward or backward	<b>w</b> — Forward <b>s</b> — Backward
Pan left or right	<b>a</b> — Left <b>d</b> — Right

### Lidar Examples: Motion compensation using sensor fusion

To learn more about how to correct the distortion in point cloud data due to the ego vehicle motion, see the Motion Compensation in 3-D Lidar Point Clouds Using Sensor Fusion example.

### Functionality being removed or changed

#### Several functions use new premultiply matrix convention

*Behavior change*

In R2022b, new geometric transformation objects, such as `rigidtfom3d` and `affinetform3d`, use a premultiply matrix convention. In previous releases, geometric transformation objects such as

`rigid3d` and `affine3d` used a postmultiply convention. There are no plans to remove these old geometric transformation objects at this time. However, you can streamline your geometric transformation workflows by switching to the new objects that use the premultiply convention. For more information, see [Migrate Geometric Transformations to Premultiply Convention](#).

These functions have been updated to support the premultiply matrix convention:

- `fuseCameraToLidar`
- `bboxCameraToLidar`
- `bboxLidarToCamera`
- `projectLidarPointsOnImage`
- `transformPointsForward` function of the `LOAMPoints` object
- `updateMap` and `findPose` functions of the `pcmapsegmatch` object
- `pcregisterloam`
- `estimateLidarCameraTransform`
- `pc2scan`

### **The `readVLR` function of the `lasFileReader` object returns both raw binary data and VLR data**

#### *Behavior change*

The `readVLR` function of the `lasFileReader` object now returns an output structure that contains both raw binary data and VLR data.

### **The `LaserReturns` argument in the `readPointCloud` function of the `lasFileReader` object has been renamed to `LaserReturn`**

#### *Errors*

The `LaserReturns` name-value argument of the `readPointCloud` function of the `lasFileReader` object has been renamed to `LaserReturn`. To update your code, replace all instances of the `LaserReturns` argument with `LaserReturn`.

### **The `WaveformData` property of the `lidarPointAttributes` object has a new field order**

The order of the `WaveformData` property structure fields has been updated to:

```
WavePacketDescriptorIndex: []
ByteOffsetToWaveformPacketData: []
WaveformPacketSize: []
ReturnPointLocation: []
    Xt: []
    Yt: []
    Zt: []
BitsPerSample: []
CompressionType: []
NumberOfSamples: []
TemporalSpacing: []
    DigitizerGain: []
    DigitizerOffset: []
```

---

Prior to R2022b, the order was:

```
Xt: []  
Yt: []  
Zt: []  
ReturnPointLocation: []  
  BitsPerSample: []  
  CompressionType: []  
  NumberOfSamples: []  
  TemporalSpacing: []  
  DigitizerGain: []  
  DigitizerOffset: []
```



# R2022a

---

**Version: 2.1**

**New Features**

**Compatibility Considerations**

## **Lidar Sensor Model: Simulate lidar sensor and generate point cloud data**

Generate lidar point cloud data by using the `lidarSensor` object. You can specify attributes such as sensor position and orientation, elevation angles, resolution, noise, and actor profiles to generate data.

## **Blocked Point Cloud: Process 3-D point clouds too large to fit in memory**

Read and process 3-D point clouds that are too large to fit in memory by using a `blockedPointCloud` object. Using a blocked point cloud, you can perform point cloud processing on smaller sections of large point clouds such as aerial point clouds.

To manage a collection of point cloud blocks that belong to one or more `blockedPointCloud` objects, use a `blockedPointCloudDatastore` object.

## **Lidar Odometry and Mapping (LOAM): Detect and register LOAM feature points from lidar data**

Detect LOAM surface points and edge points, based on curvature values in point clouds, by using the `detectLOAMFeatures` function, and store them as a `LOAMPoints` object.

You can use a `LOAMPoints` object to visualize, downsample, and apply geometric transformations to the LOAM feature points.

You can compute the rigid transformation between point clouds, or between LOAM points, by using the `pregisterloam` function.

## **Lidar File Readers: Support for Ouster sensor, Hesai sensor, and enhancements to `lasFileReader`**

- Use the `ousterFileReader` object to read lidar point cloud data from Ouster sensor PCAP files.
- Use the `hesaiFileReader` object to read lidar point cloud data from Hesai® sensor PCAP files.
- The `lasFileReader` object has these improvements.
  - Returns point cloud data of type `double`.
  - Supports intensity values of types `int16` and `uint16`.
  - The `readPointCloud` object function returns additional attributes from the LAS or LAZ file, such as user data, point source ID, scanner channel, and waveform data as a `lidarPointAttributes` object.
  - You can now read coordinate reference system (CRS) data and variable-length record (VLR) data.



---

## Data Augmentation: Lidar point cloud ground truth data augmentation for object detection

The `sampleLidarData` function samples 3-D bounding boxes and their respective labels from the input training data, and then returns them as a `fileDatastore` object and `boxLabelDatastore` object, respectively.

Using the `pcBboxOversample` function, you can augment ground truth data by adding the 3-D bounding boxes from a datastore or a table to the input point cloud.

## Deep Learning Object Detectors: Create training data to train neural networks for point cloud object detection

The `lidarObjectDetectorTrainingData` function creates training data from labeled ground truth data generated using the **Lidar Labeler** app or `groundTruthLidar` object. Use the training data returned by the `lidarObjectDetectorTrainingData` function to train deep learning networks, such as `PointPillars` using the `trainPointPillarsObjectDetector` function, for point cloud object detection.

## Point Cloud to Scan Conversion: Convert 3-D lidar point cloud to 2-D lidar scan

You can convert a 3-D lidar point cloud to a 2-D lidar scan by using the `pc2scan` function.

## Intrinsic Shape Signature (ISS) Features: Detect ISS feature points from point cloud data

Use the `detectISSFeatures` function to detect ISS feature points from the input point cloud data. An ISS feature point is a point with maximum ISS saliency in a specified neighborhood.

## Ouster Lidar Support: Stream point cloud data from Ouster sensor

Streaming point cloud data from Ouster sensors is now supported through the Lidar Toolbox Support Package for Ouster Lidar Sensors.

You can connect to and stream data from these Ouster sensor models.

- OS0-64
- OS1-64
- OS2-64

Use the `ousterLidar` object to access the point cloud data. You can preview and read point cloud data from the supported sensors.

For more information, see [Lidar Toolbox Support Package for Ouster Lidar Sensors documentation](#).

## Lidar Viewer Enhancements: Edit multiple point cloud frames, measure point cloud data

The **Lidar Viewer** app has these enhancements:

- You can now load point cloud data from any custom source into the app by using the custom reader.
- The app supports point cloud measurements such as distance, location, elevation, angle, and volume.
- You can segment ground from point cloud data and visualize point cloud clusters.
- Cropping is now more interactive. You can crop inside or outside a selected cuboidal region.
- You can edit multiple point cloud frames by using a custom temporal edit algorithm.
- Visualize the color information of the point cloud using the updated colormap.

## Lidar Labeler Enhancements: Create training data for object detection

The following table describes enhancements for these labeling apps:

- **Image Labeler**
- **Video Labeler**
- **Ground Truth Labeler**
- **Lidar Labeler** (Lidar Toolbox)

Enhancement	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler
Draw, visualize, and export semantic labels in the point cloud.	No	No	No	Yes
Create training data for object detection in point clouds by using the <code>lidarObjectDetectorTrainingData</code> function.	No	No	No	Yes
Import multiframe DICOM images.	Yes	No	No	No
Create 3-D line ROI for point cloud data.	No	No	Yes	Yes
Create voxel ROI for point cloud data.	No	No	No	Yes
Show or hide pixel labels in a labeled image or video.	Yes	Yes	Yes	No

## Lidar Camera Calibrator Enhancements: Support for 16-bit intensity images

The **Lidar Camera Calibrator** app has these enhancements.

- The app now supports 16-bit intensity images.

- 
- You can specify checkerboard square size in inches.
  - Use **Ctrl+A** to select all data pairs in the data browser.

## **lidarPointAttributes Object: Store lidar point attributes**

You can store additional lidar point data, such as classification, laser returns, and GPS time stamps, in a `lidarPointAttributes` object.

## **Lidar File Writer: Write point cloud data into LAS or LAZ format**

Use the `lasFileWriter` object to write lidar point cloud data into a LAS or LAZ file.

## **estimateLidarCameraTransform Function: Compute transform with 2-D image corners**

The `estimateLidarCameraTransform` function now accepts 2-D image corners and camera intrinsic parameters as inputs to compute the transform between a lidar sensor and a camera.

## **lidarParameters Object: Support for VLS-128, VLP-32C devices**

The `lidarParameters` object now supports the VLS-128 and VLP-32C Velodyne LiDAR® device models.

## **Lidar Examples: Explore new lidar workflows**

The Curb Detection in 3-D Lidar Point Cloud example shows you how to detect curb points in lidar point clouds. You can use the detected curb points for path planning and navigation.

The Automate Ground Truth Labeling for Lidar Point Cloud Semantic Segmentation Using Lidar Labeler example shows how to automate semantic labeling in a point cloud using a pretrained semantic segmentation network in the **Lidar Labeler** app.

The Create, Process, and Export Digital Surface Model from Lidar Data example shows you how to process aerial lidar data to generate a digital surface model (DSM) and export the DSM to a GeoTiff file.

The Build a Map with Lidar Odometry and Mapping (LOAM) Using Unreal Engine Simulation example shows you how to build a map with the Lidar Odometry and Mapping (LOAM) algorithm.

## **Deep Learning Examples: Explore new deep learning workflows**

The Lidar Object Detection Using Complex-YOLO v4 Network example shows you how to train a complex YOLO v4 network for object detection in point clouds.

The Code Generation for Lidar Object Detection Using SqueezeSegV2 Network example shows you how to generate CUDA® MEX code for lidar object detection using segmentation with a SqueezeSegV2 network followed by clustering and 3-D box fitting.

## Code Generation Support: Perform GPU and C/C++ code generation

### Generate CUDA code using GPU Coder

Generate optimized CUDA code for these additional functions:

- `pcfitcuboid`
- `segmentGroundSMRF`
- `pointPillarsObjectDetector`

### Generate C/C++ code using MATLAB Coder

The `pointPillarsObjectDetector` object supports C/C++ code generation.

## Performance Improvement: `matchScansGrid` function

The `matchScansGrid` function shows a reduction in execution time with the default search range. You can see a greater improvement with higher values of `Resolution`, `TranslationSearchRange`, `RotationSearchRange` and `MaxLidarRange` arguments.

For example, this code is about 1.8x faster than in the previous release:

```
function timingTest
% Laser scan data input
refRanges = [6*ones(1,100),7*ones(1,100),4*ones(1,100)];
refAngles = linspace(-pi/2,pi/2,300);
refScan = lidarScan(refRanges,refAngles);

% Generate the second laser scan by moving refScan 0.5 m along x
% direction and 0.2 m along y direction, and rotate the scan by 1.1
% radians, in the refScan's coordinates
currentScan = transformScan(refScan,[0.5 0.2 1.1]);

tic
% Match scans to get estimated relative pose
relPose = matchScansGrid(currentScan,refScan);
toc
end
```

The approximate execution times are:

- **R2021b:** 0.18 s
- **R2022a:** 0.10 s

The code was timed on a Windows 10, Intel® Xeon® W-2133 CPU @ 3.60GHz test system.

## Functionality being removed or changed

### The `readPointCloud` function of the `lasFileReader` object returns a `lidarPointAttributes` object

*Behavior change*

The `readPointCloud` function of the `lasFileReader` object now returns point attributes data as a `lidarPointAttributes` object. It previously returned point attributes data as a structure.

---

**The LaserReturns argument in the readPointCloud function of the lasFileReader object has been renamed to LaserReturn**

*Still runs*

The LaserReturns name-value argument in the readPointCloud function of the lasFileReader object has been renamed to LaserReturn. To update your code, replace all instances of the LaserReturns argument with LaserReturn.

**The CameraIntrinsic argument in the estimateLidarCameraTransform function is no longer required**

*Still runs*

The CameraIntrinsic name-value argument in the estimateLidarCameraTransform function is no longer required to specify the camera intrinsic parameters. You can now specify the parameters by using the intrinsics input.



# R2021b

---

**Version: 2.0**

**New Features**

**Bug Fixes**

## Lidar Viewer: Visualize and analyze lidar data using app

The **Lidar Viewer** app is an interactive tool that provides features to visualize, analyze, and preprocess lidar data. These features include:

- Load and visualizing point cloud data in various file formats, such as PLY, PCAP, LAS, LAZ, PCD, and rosbag files. You can export the processed point clouds as PLY or PCD files.
- Visualizing and analyzing the data using built-in camera views and color maps. You can also create and save custom camera views.
- Built-in preprocessing algorithms for ground removal, denoising, median filtering, cropping, and downsampling lidar data.
- Import of custom preprocessing algorithms for the lidar data. You can also create a user interface element to interactively tune the algorithm parameters.
- Export of the preprocessing operations performed on a point cloud as a function script.

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

Use the `pcorganize` function to convert unorganized point clouds into organized point clouds. You must specify the corresponding lidar sensor parameters, using the `lidarParameters` object, to convert the data.

For more information, see [Unorganized to Organized Conversion of Point Clouds Using Spherical Projection](#). You can also use the `pcorganize` function to reorganize an organized point cloud using a different set of sensor parameters.

## Object Detection Interface: Train and test PointPillars network to detect objects in lidar point clouds

Use the `pointPillarsObjectDetector` object to create a PointPillars network, or to configure the detector with a custom network. Use the `trainPointPillarsObjectDetector` function to train the object detector on training data. Then, use the trained detector to obtain detections on a point cloud using the `detect` function. For an example using a PointPillars network to detect objects, see [Lidar 3-D Object Detection Using PointPillars Deep Learning](#).

## PointNet++ Network: Create PointNet++ network for segmentation

Use the `pointnetplusLayers` function to create a PointNet++ network for point cloud segmentation. For more information, see [Aerial Lidar Semantic Segmentation Using PointNet++ Deep Learning](#).

## Digital Elevation Model: Create digital elevation model of point cloud data

Use the `pc2dem` function to generate a digital elevation model (DEM) of point cloud data using a local binning algorithm.



## Lidar Labeler Enhancements: SMRF ground segmentation algorithm

The following table describes enhancements for these labeling apps:

- **Image Labeler**
- **Video Labeler**
- **Ground Truth Labeler**
- **Lidar Labeler** (Lidar Toolbox)

Enhancement	Image Labeler	Video Labeler	Ground Truth Labeler	Lidar Labeler
Show or hide labels and sublabels of type Rectangle, Line, Polygon, and Projected cuboid in a labeled image or video.	Yes	Yes	Yes	No
Show or hide labels of type Cuboid in a labeled point cloud or point cloud sequence.	No	No	Yes	Yes
View and edit cuboid ROI labels using top, side, and front 2-D view projections by selecting <b>Projected View</b> .	No	No	Yes	Yes
Segment ground from lidar data using the simple morphological filter (SMRF) algorithm. For more information about the algorithm parameters, see the <code>segmentGroundSMRF</code> (Lidar Toolbox) function.	No	No	Yes (only with Lidar Toolbox license)	Yes
Extract video scenes and corresponding labels from a <code>groundTruth</code> or <code>groundTruthMultisignal</code> object.	No	Yes	Yes	No
Digital Imaging and Communication in Medicine (DICOM) image format.	Yes	No	No	No

## Lidar Camera Calibrator Enhancements: Manually select checkerboard points for calibration

The **Lidar Camera Calibrator** app has these enhancements:

- Manually select checkerboard points in the point cloud to improve detections. For more information, see [Select Checkerboard Region](#).
- The app now supports various keyboard shortcuts. For more information, see [Keyboard Shortcuts and Mouse Actions](#).

## **Deep Learning Example: Code generation for PointNet++ network**

The Code Generation For Aerial Lidar Semantic Segmentation Using PointNet++ Deep Learning example shows you how to generate CUDA® MEX for a multiclass PointNet++ network.

## **Lidar Examples: Explore new lidar workflows**

- Multi-Lidar Calibration — This example shows you how to calibrate multiple lidar sensors mounted on a vehicle to estimate the relative transformation between them.
- Extract Forest Metrics and Individual Tree Attributes from Aerial Lidar Data — This example shows you how to extract forest metrics, as well as the attributes of individual trees, from aerial lidar data.

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

The `segmentGroundSMRF` and `pc2dem` functions now support code generation.

## **Performance Improvement: matchScans function**

The `matchScans` function now executes 4-6 times faster.

# R2021a

---

**Version: 1.1**

**New Features**

**Bug Fixes**

## 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** (Lidar Toolbox)

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 <a href="#">Label Objects Using Polygons</a> .	Yes	Yes	Yes	No
Use superpixel automation to quickly pixel label regions of an image with similar pixel values. For more details, see <a href="#">Label Pixels Using Superpixel Tool</a> .	Yes	Yes	Yes	No
Automate the labeling of multiple signals together within a single automation run. For an example, see <a href="#">Automate Ground Truth Labeling Across Multiple Signals</a> .	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 (Lidar Toolbox).	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 (Lidar Toolbox).	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.