

本页采用了机器翻译。[点击此处可查看英文原文。](#)

使用 RoadRunner 高清地图根据自定义数据构建场景

此示例展示如何使用 RoadRunner 高清地图协议缓冲区将自定义数据导入 RoadRunner 并构建场景。在此示例中，您：

- 从自定义数据生成 RoadRunner 高清地图文件。
- 将生成的文件导入 RoadRunner 并预览 RoadRunner 高清地图数据。
- 从导入的数据构建 RoadRunner 场景。要构建场景，您必须拥有 RoadRunner Scene Builder 许可证。

此示例使用 C++ 通过协议缓冲区实现 RoadRunner 高清地图。

介绍

RoadRunner 高清地图是一种道路数据模型，用于在 RoadRunner 场景中表示高清 (HD) 地图数据。该模型定义了一个简单的数据结构来表示使用车道、车道边界、车道标记和交叉口的道路布局。该模型还定义了其他场景元素的数据结构，例如隔离物、标志和不同类型的静态目标。该模型中的数据使用协议缓冲区进行序列化，并以二进制格式保存，扩展名为 `.rrhd`。RoadRunner 高清地图二进制文件由这些数据部分组成：

- Varint 标头大小 - 指定标头消息的大小。有关详细信息，请参阅 Google® 协议缓冲区指南中的 [Base 128 Varints](#)。
- 标题消息 - 指定有关文件作者、地图投影和几何数据的空间边界的信息。
- RoadRunner 高清地图消息 - 指定车道、车道边界、车道标记和交叉口的属性。

您可以使用协议缓冲区架构文件为自定义高清地图数据创建 RoadRunner 高清地图二进制文件。这些模式文件描述了 RoadRunner 高清地图数据模型。协议缓冲区支持各种编程语言，因此您可以在实现 RoadRunner 高清地图时选择使用哪种语言。此示例使用 C++。

为 RoadRunner 高清地图编译协议缓冲区

您本地安装的 RoadRunner 包含此位置的协议缓冲区架构文件，在此示例中称为 `RoadRunnerProtoPath`。

```
RRInstallFolder\bin\platform\Proto
```

- `RRInstallFolder` 是您的本地 RoadRunner 安装文件夹。
- `platform` 是您的操作系统平台的文件夹名称。

在 Proto 文件夹内，顶级 `mathworks` 文件夹包含 `protobuf` 文件的层次结构。此表显示了您需要编译的 `protobuf` 文件及其位置。

要编译的 Protobuf 文件	Protobuf 文件位置
hd_map_header.proto	<code>RoadRunnerProtoPath\mathworks\scenario\scene\hd</code>
hd_map.proto	
hd_lanes.proto	
hd_lane_markings.proto	
hd_junctions.proto	
hd_barriers.proto	
hd_signs.proto	

要编译的 Protobuf 文件	Protobuf 文件位置
hd_static_objects.proto	
hd_phases.proto	
hd_traffic_signals.proto	
hd_curve_markings.proto	
hd_stencil_markings.proto	
hd_parking_spaces.proto	
common_attributes.proto	
geometry.proto	<i>RoadRunnerProtoPath\mathworks\scenario\common</i>

要使用 C++ 编译协议缓冲区，请按照下列步骤操作：

1. 下载并安装协议编译器 protoc（最低版本 3.8.0）和 protobuf C++ 运行时。按照 <https://github.com/protocolbuffers/protobuf> 上的说明进行操作。

安装会在安装位置创建这些文件夹：

- bin - 包含 protoc 编译器。
- include - 包含编译所需的附加头文件和 protobuf 文件。
- lib - 包含链接库。

2. 运行编译器。指定这些文件夹的文件路径：

- include - 在上一步中创建的文件夹。
- *RoadRunnerProtoPath* - 包含 RoadRunner 高清地图 protobuf 文件的文件夹。
- *DestinationPath* - 生成代码的目标目录。
- 源 protobuf 文件的绝对文件路径。

例如，使用此命令编译 `hd_map_header.proto`，其中 *DestinationPath* 指定目标目录。

```
protoc -I="..\include" -I="RoadRunnerProtoPath" --cpp_out="DestinationPath" ^
"RoadRunnerProtoPath\mathworks\scenario\scene\hd\hd_map_header.proto"
```

3. 验证编译器是否为 .proto 文件创建了这些文件：

- 扩展名为 .pb.h 的头文件。
- 扩展名为 .pb.cc 的实现文件。

4. 重复步骤 2 和 3 来编译其他 protobuf 文件。

准备自定义数据

在实践中，您必须探索要导入的自定义场景的数据以确定其车道属性。此示例使用合成数据创建一个由两个纵向连接的车道段组成的场景。每条车道有两个车道边界。该表显示了这些车道和车道边界的属性。

属性	车道 1 的值	车道 2 的值
车道 ID	Lane1	Lane2
几何形状	[(0.782, -1.56) (50.78, 23.43)]	[(50.78, 23.43) (100.78, 48.43)]
类型	Driving	Driving

属性	车道 1 的值	车道 2 的值
行车方向	Forward	Forward
前驱 ID	-	Lane1
前驱对齐	-	Forward
后继 ID	Lane2	-
后继对齐	Forward	-
左车道边界 ID	LaneBoundary1	LaneBoundary4
左车道边界几何形状	[(0,0) (50,25)]	[(50,25) (100,50)]
右车道边界 ID	LaneBoundary2	LaneBoundary3
右车道边界几何形状	[(1.565,-3.13) (51.565,21.864)]	[(51.565,21.864) (101.565,46.869)]

该示例还展示了如何表示其他场景元素（例如隔离物、标志、树木和建筑物）的数据。

从自定义数据创建 RoadRunner 高清地图二进制文件

编写 C++ 代码，编译后将自定义数据添加到生成的 RoadRunner 高清地图协议缓冲区类中。

▶ 要复制的 C++ 代码

下表描述了此示例中的代码。

代码

```
#include "mathworks/scenario/scene/hd/hd_map.pb.h"
#include "mathworks/scenario/scene/hd/hd_map_header.pb.h"
#include <google/protobuf/util/delimited_message_util.h>

#include <iostream>
#include <fstream>
#include <string>

using namespace std;
using namespace mathworks::scenario::scene::hdmap;

void WriteToRRHD(const string &filepath,
    const google::protobuf::MessageLite &headerMessage,
    const google::protobuf::MessageLite &HDMAP);

int main(int argc, char **argv)
{
    HDMAP myModel;

    // Add Lane1 to the model
    Lane* lane1 = myModel.add_lanes();

    lane1->set_id("Lane1");
```

代码

```
auto lane1Geometry = lane1->mutable_geometry();

// Start Point of Lane1
auto startPoint_lane1 = lane1Geometry->add_values();
startPoint_lane1->set_x(0.782);
startPoint_lane1->set_y(-1.56);

// End point of Lane1
auto endPoint_lane1 = lane1Geometry->add_values();
endPoint_lane1->set_x(50.78);
endPoint_lane1->set_y(23.43);

// Set type and direction for Lane1
lane1->set_lane_type(LaneType::LANE_TYPE_DRIVING);
lane1->set_travel_dir(TravelDir::TRAVEL_DIR_FORWARD);

// Add Lane2 to the Model
Lane* lane2 = myModel.add_lanes();

lane2->set_id("Lane2");
auto lane2Geometry = lane2->mutable_geometry();

// Start Point of Lane2
auto startPoint_lane2 = lane2Geometry->add_values();
startPoint_lane2->set_x(50.78);
startPoint_lane2->set_y(23.43);

// End point of Lane2
auto endPoint_lane2 = lane2Geometry->add_values();
endPoint_lane2->set_x(100.78);
endPoint_lane2->set_y(48.43);

// Set type and direction for Lane2
lane2->set_lane_type(LaneType::LANE_TYPE_DRIVING);
lane2->set_travel_dir(TravelDir::TRAVEL_DIR_FORWARD);

// Add connectivity information of Lane1 and Lane2
auto predecessor = lane2->add_predecessors();
auto predecessor_ref = predecessor->mutable_reference();
predecessor_ref->set_id("Lane1");
predecessor->set_alignment(Alignment::ALIGNMENT_FORWARD);

auto successor = lane1->add_successors();
auto successor_ref = successor->mutable_reference();
successor_ref->set_id("Lane2");
successor->set_alignment(Alignment::ALIGNMENT_FORWARD);

// Add lane boundaries to the model
auto laneBoundary1 = myModel.add_lane_boundaries();
laneBoundary1->set_id("LaneBoundary1");
auto lb1Geometry = laneBoundary1->mutable_geometry();
```

代码

```
// Define the start point of LaneBoundary1
auto startPoint_lb1 = lb1Geometry->add_values();
startPoint_lb1->set_x(0);
startPoint_lb1->set_y(0);

// Define the end point of LaneBoundary1
auto endPoint_lb1 = lb1Geometry->add_values();
endPoint_lb1->set_x(50);
endPoint_lb1->set_y(25);

auto laneBoundary2 = myModel.add_lane_boundaries();
laneBoundary2->set_id("LaneBoundary2");
auto lb2Geometry = laneBoundary2->mutable_geometry();

// Define the start point of LaneBoundary2
auto startPoint_lb2 = lb2Geometry->add_values();
startPoint_lb2->set_x(1.565);
startPoint_lb2->set_y(-3.13);

// Define the end point of LaneBoundary2
auto endPoint_lb2 = lb2Geometry->add_values();
endPoint_lb2->set_x(51.565);
endPoint_lb2->set_y(21.864);

auto laneBoundary3 = myModel.add_lane_boundaries();
laneBoundary3->set_id("LaneBoundary3");
auto lb3Geometry = laneBoundary3->mutable_geometry();

// Define the start point of LaneBoundary3
auto startPoint_lb3 = lb3Geometry->add_values();
startPoint_lb3->set_x(51.565);
startPoint_lb3->set_y(21.864);

// Define the end point of LaneBoundary3
auto endPoint_lb3 = lb3Geometry->add_values();
endPoint_lb3->set_x(101.565);
endPoint_lb3->set_y(46.869);

auto laneBoundary4 = myModel.add_lane_boundaries();
laneBoundary4->set_id("LaneBoundary4");
auto lb4Geometry = laneBoundary4->mutable_geometry();

// Define the start point of LaneBoundary4
auto startPoint_lb4 = lb4Geometry->add_values();
startPoint_lb4->set_x(50);
startPoint_lb4->set_y(25);

// Define the end point of LaneBoundary4
auto endPoint_lb4 = lb4Geometry->add_values();
```

代码

```
endPoint_lb4->set_x(100);
endPoint_lb4->set_y(50);

// Connect lane boundaries to lane
auto referenceBoundary1 = lane1->mutable_left_lane_boundary();
auto referenceBoundary1_ref = referenceBoundary1->mutable_reference();
referenceBoundary1_ref->set_id("LaneBoundary1");
referenceBoundary1->set_alignment(Alignment::ALIGNMENT_FORWARD);

auto referenceBoundary2 = lane1->mutable_right_lane_boundary();
auto referenceBoundary2_ref = referenceBoundary2->mutable_reference();
referenceBoundary2_ref->set_id("LaneBoundary2");
referenceBoundary2->set_alignment(Alignment::ALIGNMENT_FORWARD);

auto referenceBoundary3 = lane2->mutable_right_lane_boundary();
auto referenceBoundary3_ref = referenceBoundary3->mutable_reference();
referenceBoundary3_ref->set_id("LaneBoundary3");
referenceBoundary3->set_alignment(Alignment::ALIGNMENT_FORWARD);

auto referenceBoundary4 = lane2->mutable_left_lane_boundary();
auto referenceBoundary4_ref = referenceBoundary4->mutable_reference();
referenceBoundary4_ref->set_id("LaneBoundary4");
referenceBoundary4->set_alignment(Alignment::ALIGNMENT_FORWARD);

// Add barrier type to the model
auto barrierType1 = myModel.add_barrier_types();
auto barrierTypeID = "BarrierType1";
barrierType1->set_id(barrierTypeID);
auto extrusionPath1 = barrierType1->mutable_extrusion_path();
extrusionPath1->set_asset_path("Assets/Extrusions/GuardRail.rrect");

// Add barrier to the model
auto barrier1 = myModel.add_barriers();
barrier1->set_id("Barrier1");
auto barrier1Geometry = barrier1->mutable_geometry();

// Define the geometry points of barrier
auto barrier1_point1 = barrier1Geometry->add_values();
barrier1_point1->set_x(1.565);
barrier1_point1->set_y(-3.13);
auto barrier1_point2 = barrier1Geometry->add_values();
barrier1_point2->set_x(51.565);
barrier1_point2->set_y(21.864);
auto barrier1_point3 = barrier1Geometry->add_values();
barrier1_point3->set_x(101.565);
barrier1_point3->set_y(46.869);
auto barrier1_ref = barrier1->mutable_barrier_type_ref();
barrier1_ref->set_id(barrierTypeID);
barrier1->flip_laterally();
```

代码

```
// Add sign type to the model
auto signType1 = myModel.add_sign_types();
auto signTypeID = "SignType1";
signType1->set_id(signTypeID);
auto signType1_asset_path = signType1->mutable_asset_path();
signType1_asset_path->set_asset_path("Assets/Signs/US/Regulatory Signs/Sign_R2-1(30).

// Add sign to the model
auto sign1 = myModel.add_signs();
sign1->set_id("Sign1");
auto sign1Geometry = sign1->mutable_geometry();
auto sign1center = sign1Geometry->mutable_center();
sign1center->set_x(8.62);
sign1center->set_y(-3.70);
sign1center->set_z(2);
auto sign1dimension = sign1Geometry->mutable_dimension();
sign1dimension->set_length(0);
sign1dimension->set_width(0.5);
sign1dimension->set_height(0.5);
auto sign1geoOrientation = sign1Geometry->mutable_geo_orientation();
auto sign1geoAngle = sign1geoOrientation->mutable_geo_angle();
sign1geoAngle->set_roll(0);
sign1geoAngle->set_pitch(0);
sign1geoAngle->set_heading(-3383);
auto sign1TypeRef = sign1->mutable_sign_type_ref();
sign1TypeRef->set_id(signTypeID);

// Add static objects to the model
// Define pole type of static object
auto staticObjectType1 = myModel.add_static_object_types();
auto staticObjectTypeID1 = "StaticObjectType1";
staticObjectType1->set_id(staticObjectTypeID1);
auto staticObjectType1_asset_path = staticObjectType1->mutable_asset_path();
staticObjectType1_asset_path->set_asset_path("Assets/Props/Signals/WoodPost_10ft.fbx"

// Add pole to the model
pair<double, double> pole = { 8.689,-3.693 };
auto so1 = myModel.add_static_objects();
so1->set_id("Pole1");
auto so1Geometry = so1->mutable_geometry();
auto so1center = so1Geometry->mutable_center();
so1center->set_x(pole.first);
so1center->set_y(pole.second);
so1center->set_z(1.4);
auto so1dimension = so1Geometry->mutable_dimension();
so1dimension->set_length(0.10 / 2.0);
so1dimension->set_width(0.10/2.0);
so1dimension->set_height(3.3/2.0);
auto so1geoOrientation = so1Geometry->mutable_geo_orientation();
auto so1geoAngle = so1geoOrientation->mutable_geo_angle();
```

代码

```

so1geoAngle->set_roll(0);
so1geoAngle->set_pitch(0);
so1geoAngle->set_heading(0);
auto so1TypeRef = so1->mutable_object_type_ref();
so1TypeRef->set_id(staticObjectTypeID1);

// Define tree type of static object
auto staticObjectType2 = myModel.add_static_object_types();
auto staticObjectTypeID2 = "StaticObjectType2";
staticObjectType2->set_id(staticObjectTypeID2);
auto staticObjectType2_asset_path = staticObjectType2->mutable_asset_path();
staticObjectType2_asset_path->set_asset_path("Assets/Props/Trees/Eucalyptus_Sm01.fbx"

// Add trees to the model
double treePositions[15][2] = {
    { 20.6743,2.2571 },
    { 32.7286,8.2143 },
    { 44.7829,14.1714 },
    { 56.8371,20.1286 },
    { 68.8914,26.0857 },
    { 80.9457,32.0429 },
    { 93,38 },
    { 87,48 },
    { 5,8 },
    { 16.7143,13.7857 },
    { 28.4286,19.5714 },
    { 40.1429,25.3571 },
    { 51.8571,31.1429 },
    { 63.5714,36.9286 },
    { 75.2857,42.7143 }
};

for (int i = 0; i < 15; i++)
{
    auto so2 = myModel.add_static_objects();
    so2->set_id("Tree" + std::to_string(i));
    auto so2Geometry = so2->mutable_geometry();
    auto so2center = so2Geometry->mutable_center();
    so2center->set_x(treePositions[i][0]);
    so2center->set_y(treePositions[i][1]);
    so2center->set_z(1.4);
    auto so2dimension = so2Geometry->mutable_dimension();
    so2dimension->set_length(1);
    so2dimension->set_width(1);
    so2dimension->set_height(1.5);
    auto so2geoOrientation = so2Geometry->mutable_geo_orientation();
    auto so2geoAngle = so2geoOrientation->mutable_geo_angle();
    so2geoAngle->set_roll(0);
    so2geoAngle->set_pitch(0);
    so2geoAngle->set_heading(0);
    auto so2TypeRef = so2->mutable_object_type_ref();

```

代码

```

        so2TypeRef->set_id(staticObjectTypeID2);
    }

// Define building type of static object
auto staticObjectType3 = myModel.add_static_object_types();
auto staticObjectTypeID3 = "StaticObjectType3";
staticObjectType3->set_id(staticObjectTypeID3);
auto staticObjectType3_asset_path = staticObjectType3->mutable_asset_path();
staticObjectType3_asset_path->set_asset_path("Assets/Buildings/Downtown_30mX30m_6stor

// Add building to the model
pair<double, double> building = { 31.670,50.01 };
auto so3 = myModel.add_static_objects();
so3->set_id("Building1");
auto so3Geometry = so3->mutable_geometry();
auto so3center = so3Geometry->mutable_center();
so3center->set_x(building.first);
so3center->set_y(building.second);
so3center->set_z(13.3915);
auto so3dimension = so3Geometry->mutable_dimension();
so3dimension->set_length(31.5804 / 2.0);
so3dimension->set_width(30.6722 / 2.0);
so3dimension->set_height(38.7831 / 2.0);
auto so3geoOrientation = so3Geometry->mutable_geo_orientation();
auto so3geoAngle = so3geoOrientation->mutable_geo_angle();
so3geoAngle->set_roll(0);
so3geoAngle->set_pitch(0);
so3geoAngle->set_heading(0.4697);
auto so3TypeRef = so3->mutable_object_type_ref();
so3TypeRef->set_id(staticObjectTypeID3);

Header headerMessage;

// Set the author of the file
headerMessage.set_author("Author Name");

// Set the projection of the file
auto proj = headerMessage.mutable_projection();
proj->set_projection("+proj=tmerc +datum=WGS84");

auto geoBounds = headerMessage.mutable_geographic_boundary();
auto bounds = geoBounds->mutable_bounds();

// Set the maximum bounds
auto boundsMax = bounds->mutable_max();
boundsMax->set_x(107.57);
boundsMax->set_y(119.04);
boundsMax->set_z(0);

// Set the minimum bounds
auto boundsMin = bounds->mutable_min();

```

代码

```

boundsMin->set_x(-3.700);
boundsMin->set_y(-18.830);
boundsMin->set_z(0);

// Specify the output file name
string filepath = "example.rrhd";

// Write the RRHD file to disk
WriteToRRHD(filepath, headerMessage, myModel);

return 0;
}

void WriteToRRHD(const string &filepath,
    const google::protobuf::MessageLite &headerMessage,
    const google::protobuf::MessageLite &HDMap)
{
    // Open the file as output binary
    fstream fileStream(filepath, ios::out | ios::binary);

    // Write the delimited header message to the buffer
    if (!google::protobuf::util::SerializeDelimitedToOstream(
        headerMessage, &fileStream))
        cerr << "Error writing the header message" << endl;

    // Write the RoadRunner HD Map message to buffer
    if (!HDMap.SerializeToOstream(&fileStream))
        cerr << "Error writing the RoadRunner HD Map message" << endl;
}

```

将高清地图文件导入 RoadRunner

要将 RoadRunner 高清地图二进制文件导入 RoadRunner，请按照以下步骤操作：

1. 在 RoadRunner 中，将高清地图文件添加到 **Library Browser** 中的文件夹中。
2. 选择 **Library Browser** 中的文件即可在 **Attributes** 窗格中查看该文件的属性。
3. 如果 **Attributes** 窗格没有显示投影数据，请选择 **Set Custom Projection**，指定文件中 (0,0) 点的纬度和经度，然后选择 **Use Transverse Mercator At**。然后点击 **OK**。

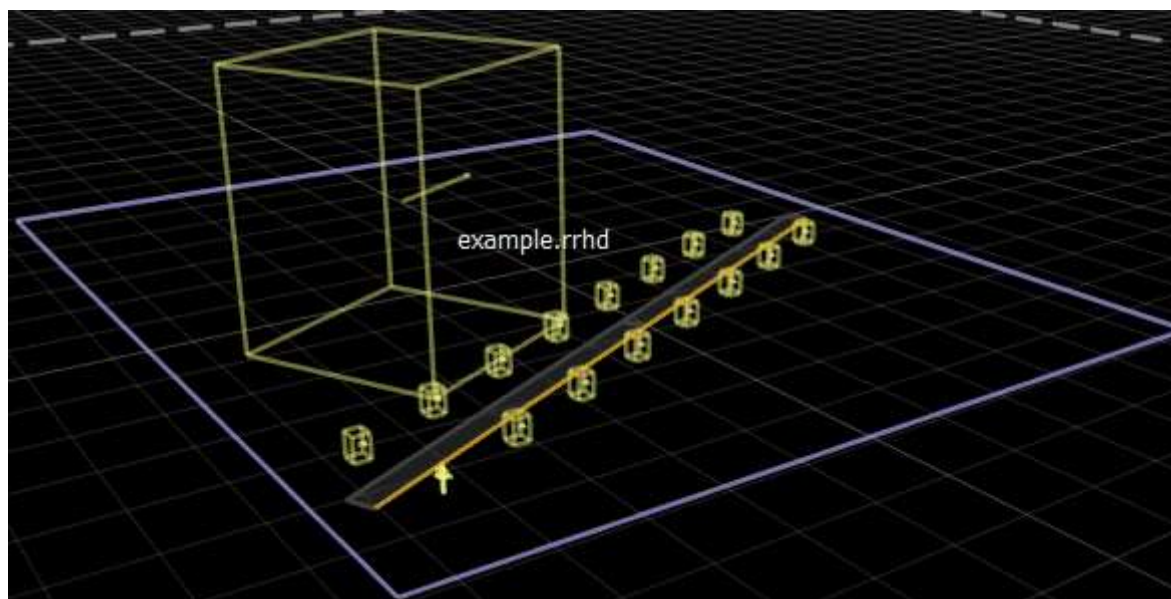
注意

当您导入包含投影数据的路径规划器高清地图文件且不设置场景原点时，路径规划器的默认行为取决于路径规划器高清地图文件的投影类型。如果文件使用横轴墨卡托投影，则路径规划器会将场景原点与导入文件的原点进行匹配。否则，路径规划器将场景原点设置为数据集的中心。但是，如果在导入文件之前设置场景原点，路径规划器将覆盖输入文件的场景原点信息。

4. 将文件从 **Library Browser** 拖到场景中。此操作切换到场景构建器工具。



场景编辑画布显示场景的 RoadRunner 高清地图。通过从 **Attributes** 窗格中选择控制点、车道和车道边界来验证导入的数据。



构建场景

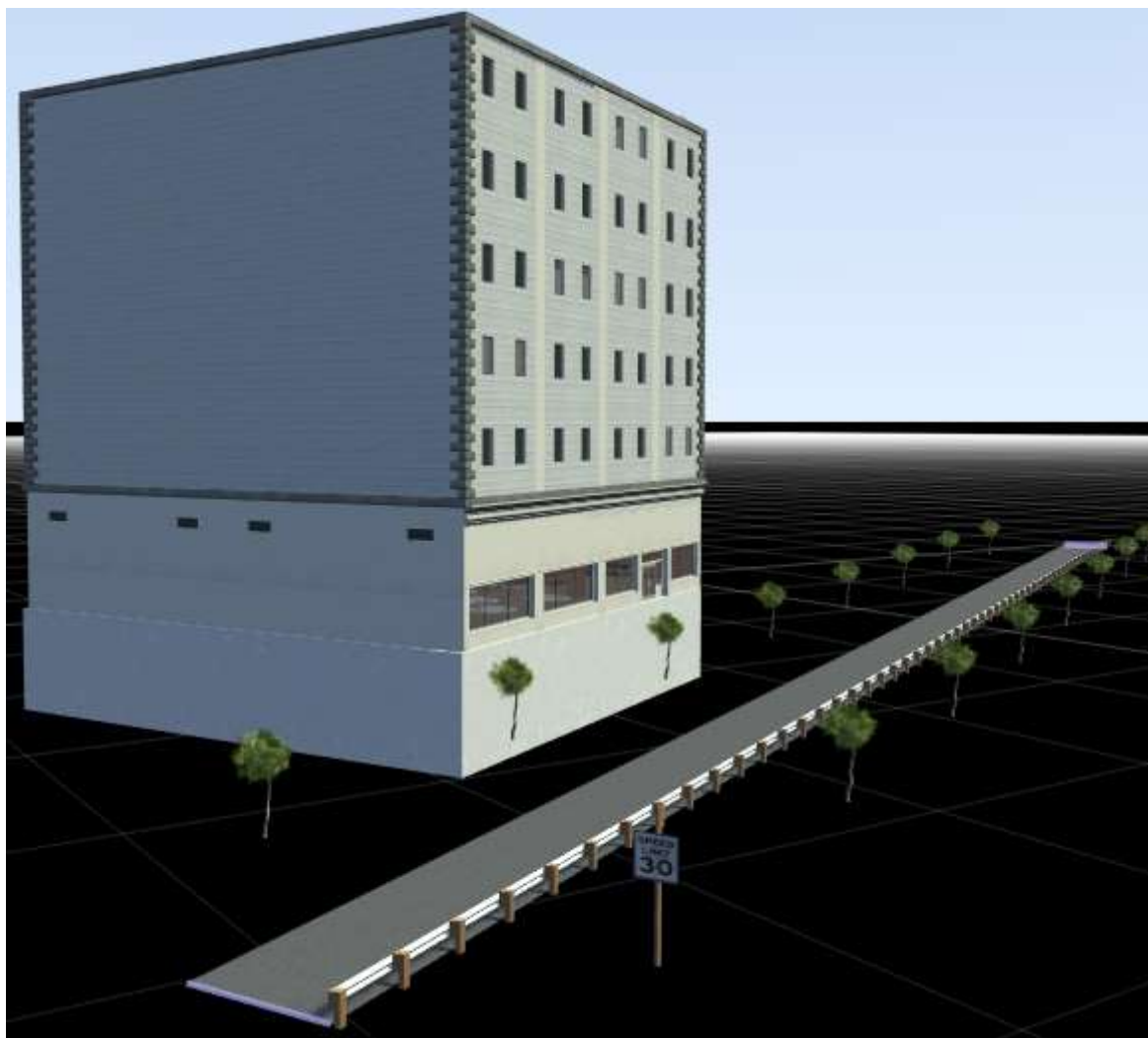
要从导入的数据构建场景，请从场景编辑画布左侧的工具栏中选择 **Build Scene**。



在“Scene Builder”对话框中，通过选择 **Build** 使用默认构建选项构建场景。要了解有关构建选项的更多信息，请参阅 [使用 HERE 高清实时地图数据构建场景](#)。

场景构建工具根据 RoadRunner 高清地图数据构建场景，并在对话框中显示有关结果的信息。点击 **Close** 关闭对话框。

要探索构建的场景，请选择其他工具来禁用 RoadRunner 高清地图数据的可见性。



i 注意

当您从导入的数据构建 RoadRunner 场景时，RoadRunner 会调整车道的速度限制值以匹配 **Tools > Scene Settings**下的 **Speed Units** 选择。

另请参阅

[场景构建工具](#) | [hd_map_header.proto](#) | [hd_map.proto](#) | [hd_lanes.proto](#) | [hd_lane_markings.proto](#) | [hd_junctions.proto](#) | [hd_barriers.proto](#) | [hd_signs.proto](#) | [hd_static_objects.proto](#) | [common_attributes.proto](#) | [geometry.proto](#) | [roadrunnerHDMMap](#)

主题

[为 RoadRunner gRPC API 编译协议缓冲区](#)

[使用 HERE 高清实时地图数据构建场景](#)

[使用 RoadRunner 高清地图以编程方式构建简单的道路](#)

[使用 RoadRunner 高清地图构建具有交叉路口和静态目标的 RoadRunner 场景](#)

[Build Pikes Peak RoadRunner 3D Scene Using RoadRunner HD Map \(Automated Driving Toolbox\)](#)

外部网站

[协议缓冲区](#)