UAV Toolbox Release Notes

MATLAB® & SIMULINK®

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

UAV Toolbox Release Notes

Contents

R2022b

R2022a

R2021a

R2020b

R2022b

Spatial Math: Use SE(2), SE(3), SO(2), and SO(3) transformations

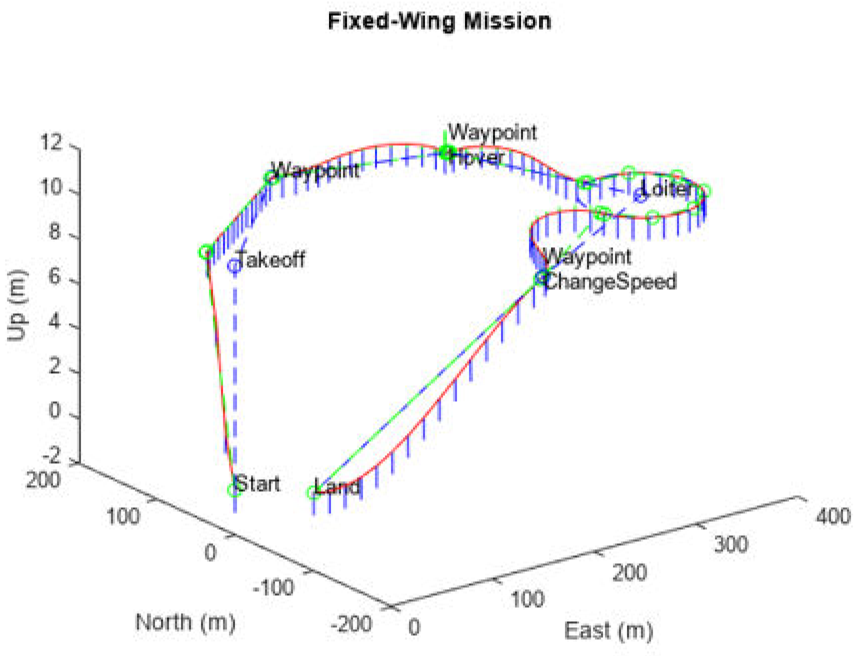
Use the `se3` and `so3` objects to represent 3-D transformation matrices and 3-D rotation matrices, respectively. These objects make it easier to perform common operations, such as transforming 3-D points, normalizing the rotation, or interpolating between positions and orientations. To represent multiple transformations and rotations, you can arrange these objects into arbitrarily shaped arrays.

The `se2` and `so2` objects represent 2-D transformation matrices and 2-D rotation matrices, respectively.

Use `plotTransforms` to visualize the transformations and rotations in a figure. This can be helpful to see the underlying positions and orientations and to see interpolation results.

UAV Mission: Design and import UAV missions

Use the `uavMission` object to design or import UAV missions for multirotor and fixed-wing UAVs. Import UAV missions from plan files (`.plan`) or waypoint files (`.waypoints`) and add mission actions using object functions such as `addTakeoff` and `addWaypoint`. Use the `fixedwingMissionParser` and `multirotorMissionParser` mission parser objects to generate flight trajectories as `fixedwingFlightTrajectory` and `multirotorFlightTrajectory` objects, respectively. For more information, see “Simulate UAV Mission in Urban Environment”.



Obstacle Avoidance: Avoid obstacles using 3D vector field histogram

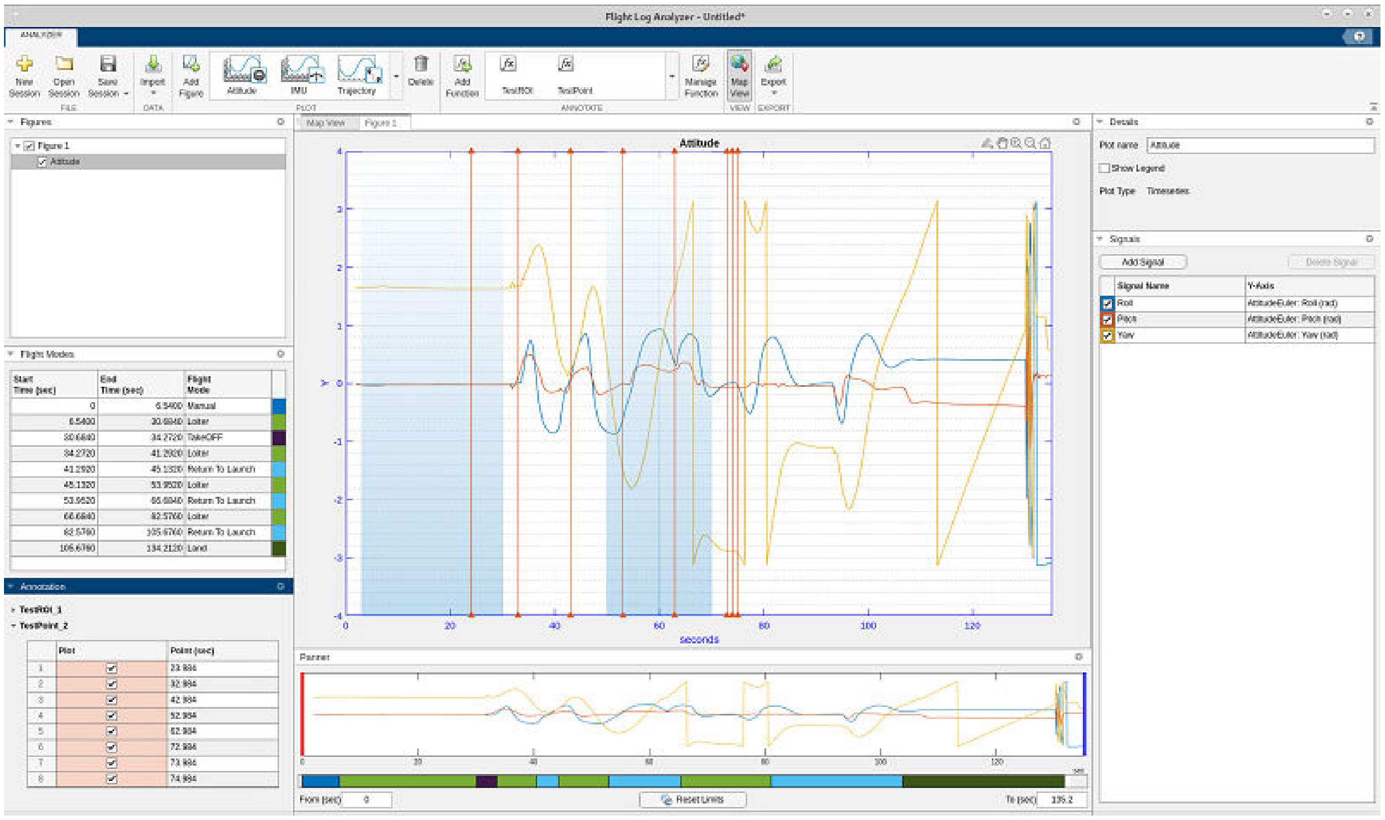
Use the `controllerVFH3D` System object to compute an obstacle-free direction for a UAV using the sensor-data-based positions of obstacles, the UAV position, the UAV orientation, and a target position. Use the `show` function to visualize the sensor readings and cost matrix, which can help you tune the parameters of your algorithm. For more information, see “Tune 3D Vector Field Histogram Controller for Obstacle Avoidance in 3D Scene”.

Flight Log Analyzer: Flight Log Analyzer app enhancements

The **Flight Log Analyzer** app has been updated with these enhancements:

- The **Flight Log Analyzer** app now supports custom annotations. Custom annotation enables you to write custom functions to label plots with region of interest (ROI) and point labels. The **Function Gallery** shows all of the custom functions you add. You can use the **Annotation** pane to interact with the labels and display them on the plots. You can export the annotations and signals to workspace as `labeledSignalSet` objects.
- Improved filtering of the `ulogreader` object, the `mavlinktlog` object, and a `flightLogSignalMapping` object with custom log data from other variables when importing them from the workspace.
- The **Signal Browser** pane now opens as a window when you double-click the data field of the desired signal in the **Signals** pane.
- The **Left** and **Right** boxes in the **Panner** pane have been renamed to **From (sec)** and **To (sec)**, respectively. The boxes are now enabled only when a log file has been loaded into the app.
- The **Name** box in the **Details** pane is now either **Plot name** or **Figure name** depending on the item selected in the **Figures** pane.

  - **Plot name** — Select a plot item in the **Figures** pane.
  - **Figure name** — Select a figure item in the **Figures** pane.



Flight Modes

Start Time (sec)	End Time (sec)	Flight Mode	
0	6.5400	Manual	🟦
6.5400	30.6840	Loiter	🟩
30.6840	34.2720	TakeOFF	🟪
34.2720	41.2920	Loiter	🟩
41.2920	45.1320	Return To Launch	🟦
45.1320	53.9520	Loiter	🟩
53.9520	66.6840	Return To Launch	🟦
66.6840	82.5760	Loiter	🟩
82.5760	105.6760	Return To Launch	🟦
105.6760	134.2120	Land	🟩

Annotation

- TestROI_1
- TestPoint_2

	Plot	Point (sec)
1	✓	23.984
2	✓	32.984
3	✓	42.984
4	✓	52.984
5	✓	62.984
6	✓	72.984
7	✓	73.984
8	✓	74.984

Plot name: Attitude

Show Legend

Plot Type: Timeseries

	Signal Name	Y-Axis
✓	Roll	AttitudeEuler: Roll (rad)
✓	Pitch	AttitudeEuler: Pitch (rad)
✓	Yaw	AttitudeEuler: Yaw (rad)

From (sec): 0

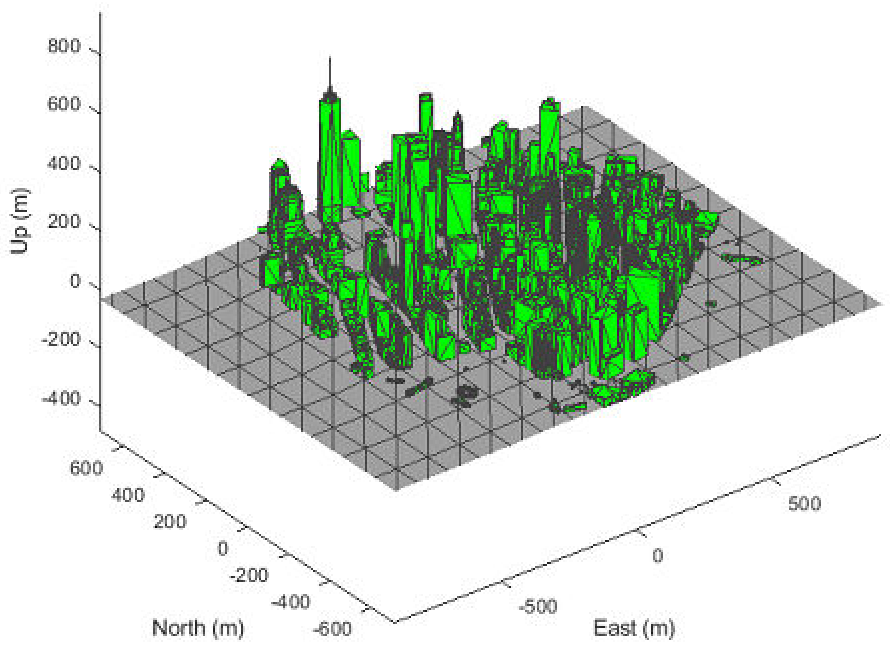
To (sec): 135.2

OSM Import Enhancement: UAV scenario OSM import verbose mode

The `addMesh` function of the `uavScenario` object now supports verbose OSM import using a name-value argument. Specify `Verbose` as `true` to print or store import information, such as the total number of imported buildings.

```
scene = uavScenario(ReferenceLocation=[40.707088 -74.012146 0]);
addMesh(scene,"terrain",{"gmted2010",[-1000 1000],[-1000 1000]},[0.6 0.6 0.6])
addMesh(scene,"buildings",{"manhattan.osm",[-800 800],[-800 800],"auto"},[0 1 0],Verbose=1)
show3D(scene);
```

```
Importing C:\...\matlab\toolbox\shared\openstreetmapdata\manhattan.osm
Latitude range (degree): [40.701 40.7182]
Longitude range (degree): [-74.0188 -74.0003]
Imported 657 out of 836 buildings
UAV Scenario Reference Latitude (degree): 40.7071
UAV Scenario Reference Longitude (degree): -74.0121
```



Unreal Engine Scene: Added new suburban scene

The **Suburban scene** is an Unreal Engine® environment of a suburban area that contains houses with backyards, trees, power poles, and vehicles on the road.

Simulation 3D Scene Configuration: Use new MATLAB API to download maps locally from the server

Starting in R2022b, you can use the `sim3d.maps` class and its object functions to download prebuilt Unreal Engine® scenes and access them directly from the Simulation 3D Scene Configuration block.

- The `sim3d.maps.Map.download` function downloads a map from the server and makes the map locally available.
- The `sim3d.maps.Map.delete` function deletes the map that you download from the server.
- The `sim3d.maps.Map.server` function lists all the maps on the server that are available for download.
- The `sim3d.maps.Map.local` function lists the locally available maps downloaded from the server.

Fisheye Camera Block Update: Fisheye camera block requires Computer Vision Toolbox and Image Processing Toolbox

Starting in R2022b, the Simulation 3D Fisheye Camera block requires Computer Vision Toolbox™ and Image Processing Toolbox™.

waypointTrajectory Reverse Motion: Specify wait and reverse motion for waypoint trajectory

You can now specify wait and reverse motion using the `waypointTrajectory` System object.

- To let the trajectory wait at a specific waypoint, simply repeat the waypoint coordinate in two consecutive rows when specifying the Waypoints property.

- To render reverse motion, separate positive (forward) and negative (backward) groundspeed values by a zero value in the GroundSpeed property.

PX4 Support Package Updates

See *Release Notes for UAV Toolbox Support Package for PX4 Autopilots* (UAV Toolbox Support Package for PX4 Autopilots) for more details.

New Examples

This release contains these new examples:

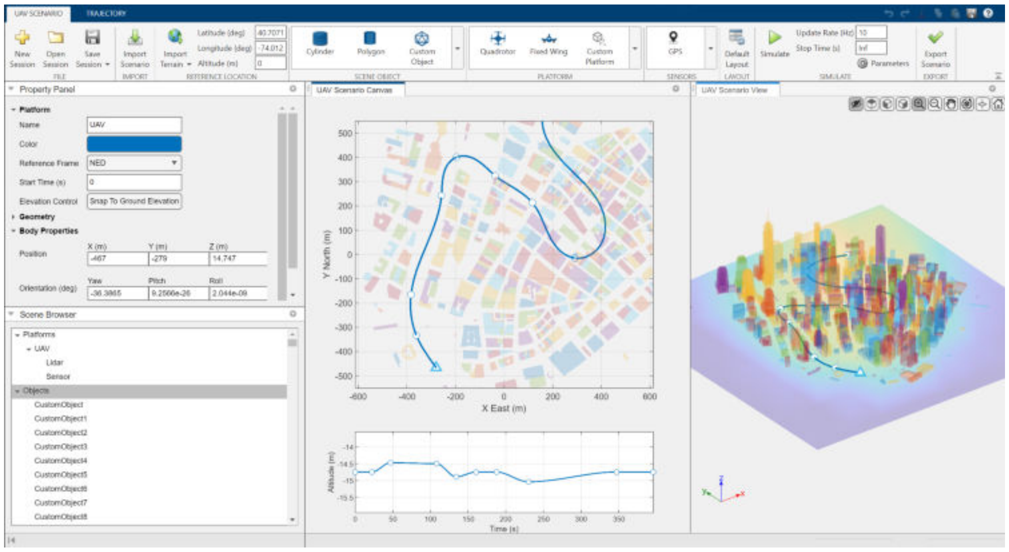
- “Add Custom Functions in Flight Log Analyzer App to Detect Region of Interest for Analysis”
- “Tune 3D Vector Field Histogram Controller for Obstacle Avoidance in 3D Scene”
- “Aerodynamic Parameter Estimation Using Flight Log Data”
- “Simulate UAV Mission in Urban Environment”
- “Map and Classify Urban Environment Using UAV Camera and Deep Learning”

    1. “Survey Urban Environment Using UAV”
    2. “Obtain Orthophotos from Central Perspective Images”
    3. “Semantic Segmentation of Orthophotos”
    4. “Create and Smooth Orthomosaic from Orthophotos”

R2022a

UAV Scenario Designer: Design and visualize UAV simulation scenarios interactively

Interactively design and visualize UAV Scenarios for path planning and control algorithms using the **UAV Scenario Designer**. Import terrain, pick and place simple and complex geometries to create a complex scene. Add multiple drones to the scenario and assign trajectories. Visualize drone moving through scenario. Place sensors on your UAV and simulate the scenario before exporting to MATLAB or Simulink.



Simulation 3D Lidar Reflectivity: Model surface reflections in Unreal Engine environment

In the Simulation 3D Lidar block, use the **Reflectivity** output port to output the reflectivity of surface materials in the Unreal Engine environment.

OpenCV Radial Distortion in Simulation 3D Camera Block: Simulate cameras with OpenCV supported radial distortion model in Unreal Engine Environment

In the Simulation 3D Camera block, you can now use the OpenCV six-coefficient formula for modeling radial distortion. Specify the formula to the **Radial distortion coefficients** parameter. This is in addition to the two-coefficient and three-coefficient models already supported by camera calibration tools in Computer Vision Toolbox. For more information on calibrating a camera using the six-coefficient formula, see Camera Calibration and 3D Reconstruction in the OpenCV documentation.

Simulation 3D Camera Performance Improvements: Run cameras at improved speeds during Unreal Engine simulation

The Simulation 3D Camera block now has improved simulation performance and runs at higher frame rates. This table shows the increase in frames per second (FPS) for each camera in an Unreal Engine simulation.

Number of Cameras in Simulation	Frame Rate per Camera (R2021b)	Frame Rate per Camera (R2022a)	Percent Improvement per Camera
1	75.85 FPS	88.45 FPS	16.6%
4	30.51 FPS	32.80 FPS	7.5%

These simulations were timed on a Windows 10, Intel® Xeon® W-2133 CPU @ 3.60 GHz, with 64 GB of RAM and a GPU with 8 GB of on-board RAM.

These improvements enable you to run cameras at real-time speeds, provided that your system meets the requirements specified by the Unreal Engine Simulation Environment Requirements and Limitations.

Simulation 3D Environment Upgrade: Run 3D simulations using Unreal Engine 4.26

The 3D visualization engine that comes installed with UAV Toolbox has been updated to Unreal Engine 4.26. Previously, the toolbox used Unreal Engine 4.25.

For information about using Unreal Engine to create custom scenes, see Customize Unreal Engine Scenes for UAVs and Unreal Engine Simulation Environment Requirements and Limitations.

Compatibility Considerations

If your Simulink® model uses an Unreal Engine executable or project developed using a prior release of the UAV Toolbox Interface for Unreal Engine Projects support package, the simulation may return an error. To migrate the project so that it is compatible with the R2022a version of the support package, see Migrate Projects Developed Using Prior Support Packages.

Trajectory Generation: Create minimum jerk and minimum snap polynomial trajectories with Simulink

Use the Minimum Jerk Polynomial Trajectory block to generate multi-axis, minimum jerk, polynomial trajectories.

Use the Minimum Snap Polynomial Trajectory block to generate multi-axis, minimum snap, polynomial trajectories.

Import buildings from OSM files

The `addMesh` function now accepts `"buildings"` as an argument to import buildings from an OSM file as a mesh.

MAVLink 3p upgrade in UAV Toolbox

The MAVLink message definitions (dialects) and MAVLink serialization helper functions in UAV Toolbox now obtain their message definitions and serialization functions from the mavlink/message_definitions/v1.0/ and mavlink/c_library_v2 versions of the repositories, respectively.

PX4 Support Package Updates

See *Release Notes for UAV Toolbox Support Package for PX4 Autopilots* (UAV Toolbox Support Package for PX4 Autopilots) for more details.

New Examples

This release contains several new examples:

- Design Obstacle Avoidance Package Delivery Scenario Using UAV Scenario Designer
- Control a Simulated UAV Using ROS 2 and PX4 Bridge
- PID Autotuning for UAV Quadcopter
- UAV Inflight Failure Recovery

R2021b

Simulate UAV Scenarios Using Simulink

You can now use UAV scenarios to simulate systems and advance the simulation using Simulink blocks:

- UAV Scenario Configuration — Import a `uavScenario` object and simulate the scenario.
- UAV Scenario Get Transform — Get a transform matrix that maps points in the source frame to the target frame.
- UAV Scenario Lidar — Simulate lidar measurements based on meshes in the scenario.
- UAV Scenario Motion Read — Read platform and sensor motions.
- UAV Scenario Motion Write — Update platform motion in the simulation.
- UAV Scenario Scope — Visualize scenario and lidar point clouds.

GPS Sensor Block

Use the GPS block to simulate a GPS sensor with noise-corrupted measurements based on the input position and velocity. The block uses the WGS84 earth model to convert local coordinates to latitude-longitude-altitude (LLA) coordinates. You can vary the horizontal position, vertical position, and velocity accuracies to change noise levels in the output signals. Also, you can specify a decay factor, which models global position noise.

INS Sensor Block

Use the INS block to model and simulate an INS sensor and generate INS measurements in Simulink.

Video Streaming to UAV Hardware

The Video Send block sends a UDP video stream to a network address. Send images as a video feed to a hardware board, such as the NVIDIA® Jetson®, or other compatible GStreamer clients. You can send images captured by sensors in an Unreal® scenario for hardware-in-the-loop (HIL) simulation.

Fixed-Wing UAV Point Mass Block

The Fixed-Wing UAV Point Mass block integrates the fourth- or sixth-order fixed-wing point mass equations of motion in coordinated flight.

Obstacle Avoidance Block

Use the Obstacle Avoidance block to compute an obstacle-free direction using range sensor data and a target position.

Minimum Jerk and Snap Polynomial Trajectories

Use the `minjerkpolytraj` function to generate a minimum jerk polynomial trajectory. Specify waypoints, time points, and number of samples to get a series of jerks, accelerations, velocities, and positions.

Use the `minsnappolytraj` function to generate a minimum snap polynomial trajectory. Specify waypoints, time points, and number of samples to get a series of snaps, jerks, accelerations, velocities, and positions.

Flight Log Analyzer App Enhancements

The **Flight Log Analyzer** app has been updated with these enhancements:

- The flight modes in the **Flight Modes** pane are now interactive. You can select one or more consecutive flight modes. The **Panner** limits values and adjusts handles accordingly.
- The **Signals** pane now displays the units of the signals.
- The axes labels of the plots are now customizable.
- You can now export multiple plots to a single figure or separate figures.

Simulation 3D Environment Upgrade: Run 3D simulations using Unreal Engine, Version 4.25

The 3D visualization engine that comes installed with UAV Toolbox has been updated to Unreal Engine, Version 4.25. Previously, the toolbox used Unreal Engine, Version 4.23.

For information about using Unreal Engine to create custom scenes, see Customize Unreal Engine Scenes for UAVs.

Compatibility Considerations

If your Simulink model uses an Unreal Engine executable or project developed using a prior release of the UAV Toolbox Interface for Unreal Engine Projects support package, the simulation might produce an error. To migrate the project so that it is compatible with the R2021b version of the support package, see Migrate Projects Developed Using Prior Support Packages.

Simulation 3D Environment Performance Improvements: Run 3D simulations faster than real-time

Simulink co-simulations with Unreal Engine can run faster than real-time. Previously, the Unreal Engine frames-per-second (FPS) was limited by the inverse of the *simulation* sample rate. If you want to slow down a 3D simulation to investigate the system behavior, you can still use simulation pacing.

To control simulation time, use the Simulation 3D Scene Configuration block parameter **Sample time**. For example, if **Sample time** is `1/30`, then the visualization engine solver tries to achieve a *minimum* frame rate of 30 fps. However, the real-time graphics frame rate is often lower due to factors such as graphics card performance and model complexity. With reasonable graphics card performance and less model complexity, the fps could be greater than 30, not limited to 30 as it was in previous releases.

Position Adjustments of Unreal Engine Cameras: Update relative translation and rotation of camera sensors during simulation

In the Simulation 3D Camera and Simulation 3D Fisheye Camera blocks, use the **Translation** and **Rotation** input ports to update the position of the cameras relative to their mounting positions during simulation. You can use these position adjustments to better model actuator dynamics, isolation mounting, and calibration workflows.

Previously, you could set only constant relative positions by using the **Relative translation [X, Y, Z] (m)** and **Relative rotation [Roll, Pitch, Yaw] (deg)** parameters. Now, by selecting **Input** to enable the corresponding ports, the parameters specify the initial position and the ports specify the position during simulation.

Unreal Engine Vehicle Enhancements: Import custom meshes

You can configure the Simulation 3D UAV Vehicle block to import custom meshes using this process:

1. Install the UAV Toolbox Interface for Unreal Engine Projects support package. See Install Support Package for Customizing Scenes.
2. On the block **Parameters** tab, set **Type** to `Custom`.
3. In the **Path to custom mesh** field, enter the path to the vehicle mesh in the Unreal Engine project. To create a custom vehicle mesh, see Prepare Custom UAV Vehicle Mesh for the Unreal Editor.
4. Use the vehicle dimensions in the custom mesh to enter the dimensions in the corresponding block parameter fields.

R2021a

Version: 1.1

New Features

Terrain and Mesh Support in UAV Scenarios: Add DTED terrain data or meshes to simulated UAV scenarios

Load digital terrain elevation data (DTED) using the `addCustomTerrain` function, and add the terrain to the UAV scenario using the `addMesh` function.

To specify complex mesh objects as triangulated vertices and faces, use the `extendedObjectMesh` object. Add the mesh to UAV scenarios using the `addMesh` function.

Custom Sensor Support in UAV Scenarios: Add custom sensor models to UAVs in simulated scenarios

Specify custom sensor models and define their behavior in simulation using the `uav.SensorAdapter` class. To generate a template for implementing the class, use the `createCustomSensorTemplate` function.

Flight Log Analyzer Update on Signal Export: Export signals as timetables to the MATLAB workspace or a MAT-file

The **Flight Log Analyzer** app now enables you to export signals as timetables to the MATLAB workspace or a MAT-file. To export signals, select **Export** > **Export Signal**. In the **Export Signal** dialog box, select the listed signals you want to export, select the export format, and click **Export**.

Flight Log Signal Mapping Object Update: Check mapped signals using flight log data

Use the `checkSignal` object function of the `flightLogSignalMapping` object to verify mapped signals using the flight log data.

Unreal Engine Scene Environment: Control weather and sun position

Use the Simulation 3D Scene Configuration block to control scene weather and sun position. Options allow you to create realistic environments when you run maneuvers and test control algorithms in the Unreal Engine 3D simulation environment. The Simulation 3D Camera and Simulation 3D Fisheye Camera blocks receive the image from the 3D simulation environment.

To control scene weather and sun position, on the Simulation 3D Scene Configuration block **Weather** tab, select **Override scene weather**. Use the enabled parameters to change the sun position, clouds, fog, and rain during the simulation.

R2020b

Version: 1.0

New Features

UAV Scenarios and Sensor Models: Construct cuboid scenario and simulate sensor readings for UAV applications

The uavSensor object generates a simulation scenario consisting of static meshes, UAV platforms, and sensors in a 3D environment. Add uavPlatform objects to the scenario and attach different sensor models with the uavSensor object.

The uavPlatform object represents an unmanned aerial vehicle (UAV) platform in a given UAV scenario. Use the platform to define and track the trajectory of an object in the scenario. To simulate sensor readings for the platform, mount sensors like the gpsSensor, insSensor , or uavLidarPointCloudGenerator objects with a set of uavSensor objects. Add a body mesh for visualization using updateMesh. Set geofencing limitations using addGeoFence and check those limits using checkPermission.

The uavSensor object creates a sensor that is rigidly attached to a UAV platform, specified as a uavPlatform object. You can specify different mounting positions and orientations. Configure this object to automatically generate readings from a sensor specified as an insSensor, gpsSensor, oruavLidarPointCloudGenerator object.

For an introduction to scenarios, see UAV Scenario Tutorial.

3D Simulation: Develop, test, and verify UAV algorithms in a 3D simulation environment rendered using the Unreal Engine from Epic Games

UAV Toolbox provides a co-simulation framework that models driving algorithms in Simulink and visualizes their performance in a virtual simulation environment. This environment uses the Unreal Engine from Epic Games®.

For an introduction to 3D simulation, see Unreal Engine Simulation for Unmanned Aerial Vehicles.

Flight Log Analyzer App: Import and visualize autopilot flight log with pre-defined and customizable plots

To analyze a log file and create a customized series of plots, use the **Flight Log Analyzer** app.

Flight logs for UAVs contain large amounts of data with many varying formats. Use the flight log analysis functions to load different telemetry log files including TLOG, ULOG, and custom file types. To extract and map signals from a telemetry log for generating plots, use the flightLogSignalMapping object.

For more information, see Flight Log Analysis.

UAV Motion Planning and Control: Design and simulate autonomous missions for fixed-wing and multirotor UAV with waypoint following, orbiting, and path management

Plan and execute UAV flights using guidance motion models for fixed-wing and multirotor UAVs. Fly predefined missions using waypoint and trajectory following algorithms.

For an example using an RRT path planner that plans and simulates a flight in a city setting, see **Motion Planning with RRT for Fixed-Wing UAV** .

For more information, see Planning and Control.

MAVLink Connectivity and Deployment: Communicate with UAV hardware using the MAVLink protocol and deploy to target hardware

The Micro Air Vehicle Link (MAVLink) communication protocol is a message protocol for sending and receiving messages between UAVs. The protocol uses a publish-subscribe pattern for data streams with specified topics and message types. There are different sub-protocols for missions and parameters. Use the MAVLink supported functions or blocks to specify predefined or custom dialects, setup clients, and send or receive messages.

UAV Reference Application: Use a reference model to simulate and test UAV package delivery with obstacle avoidance

For this reference application, see the UAV Package Delivery example.

UAV Toolbox Support Package for PX4 Autopilots: Build and deploy flight control algorithms to the Pixhawk Autopilot

For more information, see UAV Toolbox Support Package for PX4 Autopilots.