MVIS Auto Annotation

MOSAIK Suite™ 2.0

User Manual
19-Oct-2023

MicroVision GmbH
Neuer Höltigbaum 6
22143 Hamburg
Germany
Phone: +49 40 298 676-0
Fax: +49 40 298 676-10
E-mail: info@microvision.eu

Table of Contents

1. About MVIS Auto Annotation

MVIS Auto Annotation is a cloud-compatible software that automatically post-processes recorded sensor output to create reference data.

You can post-process the data of MVIS lidar sensors and also of selected third-party lidar sensors.

MVIS Auto Annotation automatically detects, classifies, and tracks objects. The software also detects the ground surface, lane markings, road boundaries, traffic signs, traffic lights, free space, and scenarios on highways.

MVIS Auto Annotation has a GUI version and a terminal version. Both versions are installed together but must be licensed separately.

1.1. Version history

Document date	Software version	Changes
19-Oct-2023	2.0	General: the following features are now documented: • coordinate systems • ego state estimation • point cloud mapping • object tracking New feature: camera perception New feature: ground detection New feature: free space detection New feature: direct output visualization in MVIS Laser View 2 Updated feature: processing options Updated feature: road detection Updated feature: scenario detection
28-Mar-2023	1.5	Rebranding: ibeo Auto Annotation is now MVIS Auto Annotation. Company name, product names and contact information were adjusted accordingly. Updated feature: scenario detection Updated feature: road detection Updated feature: user settings Updated feature: licensing files
15-Jun-2022	1.4	Updated feature: scenario detection
21-Mar-2022	1.3	Updated feature: road detection Updated feature: OpenDRIVE appendix
31-Jan-2022	1.1	Updated feature: system requirements

Document date	Software version	Changes
15-Dec-2021	1.1	General: new document layout New feature: proxy settings
25-Nov-2021	1.0	Updated feature: road detection Updated feature: scenario detection Updated feature: trip processing settings in the `jobs.xml` file Updated feature: OpenDRIVE elements in road detection
07-Jun-2021	1.0	First document version

1.2. Related documents

The following documents and files are referred to in this user manual.

You can download these documents and files on MyMVIS, MicroVision's customer platform: https://my.microvision.eu → **Downloads** tab.

Document	Location on MyMVIS
MVIS Laser View User Manual	**Software Products** → **MOSAIK Suite™** → **Documentation** tab **Software Products** → **MVIS Laser View** → **Documentation** tab
MVIS Laser View 2 User Manual	**Software Products** → **MOSAIK Suite™** → **Documentation** tab **Software Products** → **MVIS Laser View 2** → **Documentation** tab
MVIS License Manager User Manual	**Software Products** → **MOSAIK Suite™** → **Documentation** tab **Licensing** → **MVIS License Manager** → **Documentation** tab
MVIS SDK documentation	**Software Products** → **MVIS SDK** → **Software** tab
MVIS Data Interface Specification	**General** → **MVIS Interface** → **Documentation** tab

1.3. Audience

This user manual is written for trained and qualified staff who will integrate, configure, and operate MVIS Auto Annotation in the operating environment.

1.4. Abbreviations

Abbreviation	Meaning	More information
GPS	Global Positioning System	https://en.wikipedia.org/wiki/Global_Positioning_System
IDC	MVIS Data Container	List of supported file formats (page 8)
IMU	Inertial measurement unit	https://en.wikipedia.org/wiki/Inertial_measurement_unit
INS	Inertial navigation system	https://en.wikipedia.org/wiki/Inertial_navigation_system

Abbreviation	Meaning	More information
ISMC	Idealized sensor measurement coordinate system	Coordinate systems (page 9)
MSL	Mean sea level	https://en.wikipedia.org/wiki/Sea_level
UTM	Universal Transverse Mercator	https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
VBC	Vehicle body coordinate system	Coordinate systems (page 9)
VRC	Vehicle road coordinate system	Coordinate systems (page 9)
WGS84	World Geodetic System 1984	https://en.wikipedia.org/wiki/World_Geodetic_System#WGS84
WRC	World reference coordinate system	Coordinate systems (page 9)

1.5. List of supported file formats

Type	File formats	More information
Input	IDC	MVIS Data Container
	MDF4 PCAP PCAPNG VPCAP ASC ROS1.BAG DBC	Formats that contain raw data being received directly from loggers.
	PCD LAS ROS1.BAG CSV	Formats that contain interpreted data like point clouds.
Output	IDC	MVIS Data Container
	CORR	Correction files can be used for the Smart Editing feature of MVIS Laser View. For more information, refer to the *MVIS Laser View User Manual*, see Related documents (page 7).
	CSV	CSV files contain the following: • Scenario detection labels, see Output description for scenario detection (page 96) • Free space detection labels, see Output description for free space detection (page 80).
	XODR	OpenDRIVE files contain road detection information such as lane markings, road boundaries, traffic signs, and traffic lights. For more information, see Output description for road detection (page 71).

1.6. Trips and trip processings

Trips

A trip is a group of files that you recorded while test driving. You can process these trips and automatically evaluate the trips based on defined processing options.

File formats such as IDC and VPCAP are supported, see List of supported file formats (page 8).

The following applies:

- You cannot include different file formats in one trip. Select only one file type for a trip.
- You can include several trips with different file formats in one trip processing. So, you can, for example, analyze trips with IDC files and trips with VPCAP files in the same trip processing.

Trip processings

When you process trips, the input files that are included in the trips are evaluated based on the processing options that you set.

You can process the same input files several times with different processing options.

You can include several trips in one processing. The processing time depends on the size of data that you process.

Before processing, perform a validity check to see if your input files are valid, see Validity check for input files (page 25).

1.7. Coordinate systems

The following coordinate systems are relevant to understand and interpret the results of MVIS Auto Annotation. All of these coordinate systems are right-handed Cartesian coordinate systems.

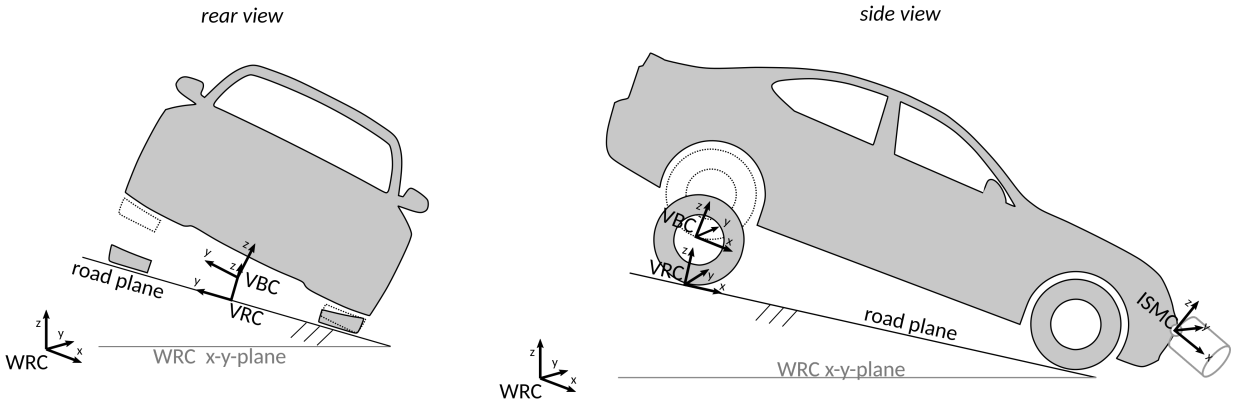
Abbreviation	Description	More information
WRC	World reference coordinate system	See World reference coordinate (WRC) system (page 10).
VRC	Vehicle road coordinate system	See Vehicle road coordinate (VRC) system (page 11).
VBC	Vehicle body coordinate system	See Vehicle body coordinate (VBC) system (page 11).
ISMC	Idealized sensor measurement coordinate system	See Idealized sensor measurement coordinate (ISMC) system (page 11).

NOTE

The OpenDRIVE standard defines additional coordinate systems that are relevant to understand the results of road detection. For more information, see asam.net.

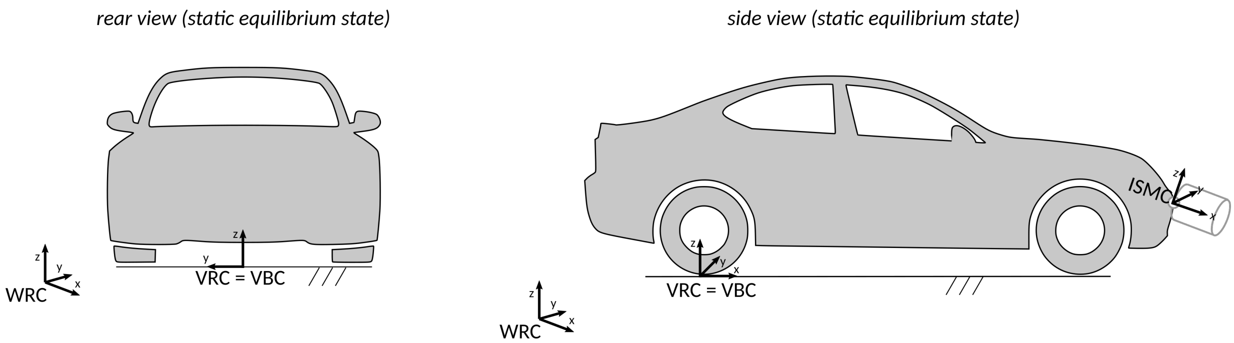
Coordinate systems overview

The image below provides an overview of the different coordinate systems. The VRC is always aligned to the road plane while the VBC is fixed to the vehicle body. That means that the VBC moves with the car body, for example, when breaking.



The road plane is the actual road surface under the ego vehicle idealized to a plane. The road plane changes as the vehicle drives and is not aligned to the static WRC system.

The image below depicts the different coordinate systems at static equilibrium state, that is, during stand still. VRC and VBC are identical at static equilibrium state, while in general, for example, when driving, this is not the case.



1.7.1. World reference coordinate (WRC) system

The world reference coordinate (WRC) system is fixed to the world.

The WRC origin is given as a UTM coordinate consisting of a UTM zone, x- and y-coordinates in UTM projection and z-coordinates as height above MSL.

The x-axis is parallel to the UTM northing-axis of the given zone pointing in the same direction. Note, that the x-direction is not exactly north direction depending on the position due to the UTM projection (see meridian convergence, Wikipedia).

The y-axis is parallel to the UTM easting-axis but pointing in the opposite direction (UTM easting-axis pointing east, WRC y-axis pointing west).

The z-axis is normal to the XY-plane pointing up.

All coordinates in WRC are relative to the origin x-, y-, z-coordinates and are UTM projected using the zone of the origin.

Example: The position $pos = \begin{pmatrix} x_{pos} & y_{pos} & z_{pos} \end{pmatrix}$ is given in the WRC system. The WRC origin is given as $\begin{pmatrix} x_{ref} & y_{ref} & z_{ref} \end{pmatrix}$ with the UTM zone $zone_{ref}$. To transform pos into WGS84 coordinates, first calculate the actual UTM coordinate as $\begin{pmatrix} x_{pos} + x_{ref} & -(y_{pos} + y_{ref}) & z_{pos} + z_{ref} \end{pmatrix}$ and then apply the UTM to WGS84 conversion on this UTM coordinate with the UTM zone $zone_{ref}$.

If no valid GPS data is available in the trip recording, the trip is not georeferenced. In that case, the origin is fixed at the position and orientation of the ego vehicle at system initialization. The XY-plane is parallel to the road plane, the x-axis is pointing in forward direction, the y-axis is pointing left, and the z-axis is pointing up.

1.7.2. Vehicle road coordinate (VRC) system

The vehicle road coordinate (VRC) system is fixed to the ego vehicle.

The x-axis is the projection of the ego vehicle thrust line onto the road plane, the z-axis is normal to the road plane pointing up, and the y-axis is normal to the XZ-plane pointing left.

The VRC pitch changes, for example, when driving uphill, but is not affected by vehicle dynamics due to braking or accelerating.

The origin of the VRC system is the center of the ego vehicle rear axle projected onto the road plane.

1.7.3. Vehicle body coordinate (VBC) system

The vehicle body coordinate (VBC) system is fixed to the ego vehicle's body position and orientation.

The origin of the VBC system is defined to be identical to the VRC system in static equilibrium state, that is the steady state with the vehicle at rest on a road plane parallel to the WRC XY-plane.

In contrast to the VRC system, the VBC system is affected by vehicle dynamics such as pitch and roll due to driving maneuvers.

1.7.4. Idealized sensor measurement coordinate (ISMC) system

The idealized sensor measurement coordinate (ISMC) system is fixed to a sensor's body.

Each sensor in the ego vehicle has its own ISMC system.

The origin and orientation of these ISMC systems are defined per sensor and given in VBC.

1.7.5. Coordinate systems of MVIS Auto Annotation results

The MVIS Auto Annotation components utilize different coordinate systems and output their results with respect to one of them. The following list gives an overview.

Component	Coordinate systems	Description
Measurement data	ISMC	Measurement data is typically recorded in ISMC.
MVIS CaliGraph	ISMC VBC	MVIS CaliGraph estimates the static transformation between ISMC and VBC for lidar sensors.
Ego state estimation	WRC VRC VBC	Ego state estimation defines the origin of WRC and estimates the dynamic transformations between WRC, VRC, and VBC throughout the trip.
Point cloud mapping detection	WRC	Point cloud mapping detection generates a static point cloud map of the environment in WRC.
Road detection	WRC	Road detection estimates lane markings, road boundaries, traffic signs, and traffic lights as OpenDRIVE map in WRC. The OpenDRIVE standard defines additional coordinate systems that are relevant to understand the results of road detection. For more information, see asam.net.
Object tracking	VRC	Objects are tracked and output in VRC.
Free space detection	VRC	Free space is output in VRC.
Ground detection	WRC	Ground detection is an estimation of the static ground surface and the output is a static map in WRC.

1.8. Camera perception

MVIS Auto Annotation can make use of camera images to detect traffic signs and traffic lights.

The following properties are recommended:

- Front facing 80° horizontal field of view or higher
- 2 MP or higher (for example, 1920 x 1080)
- 30 FPS or higher
- Auto white balance
- Auto gain

2. Installation

MVIS Auto Annotation has a GUI version and a terminal version. The GUI installer will automatically install both versions.

> **NOTE**
> You must start the GUI version of MVIS Auto Annotation once to make the terminal version available. Starting the GUI version will import the MVIS Auto Annotation Docker image into your Docker installation. After that, you can execute MVIS Auto Annotation from the terminal as well. For information on how to run the Docker image as a Docker container from the terminal, see The terminal version of MVIS Auto Annotation (page 46).

After installation, activate your user license for MVIS Auto Annotation, see Licensing (page 21).

2.1. System requirements

Hardware requirements

Component	Requirements
CPU	64-bit with the following features: • Second Level Address Translation (SLAT) which is supported by most Intel CPUs since 2008 and most AMD CPUs since 2007. • On Windows, Intel Virtualization Technology (VTx) must be enabled in the BIOS. For more information, see https://docs.docker.com/docker-for-windows/troubleshoot/#virtualization.
RAM	32 GB
HDD	Depends on the data being processed. Recommendation: three times more space available than the size of the data you want to process.

Software requirements

Component	Requirements
Operating system	Windows 10, 64-bit (Pro, Enterprise, or Education) with build 18362 The latest updates must be installed.
	Linux Ubuntu 18, x86, 64-bit, Kernel version 3.10 or higher The latest updates must be installed.

Component	Requirements
Docker	Docker Engine 20.10.0 (API version 1.41) or higher For information on how to install Docker, see Install Docker on Windows (page 14) or Install Docker on Linux (page 16).

2.2. Overview of installation steps

Step		More information
1	Ensure that your system meets the system requirements.	See System requirements (page 13).
2	Download and install Docker.	See Install Docker on Windows (page 14). See Install Docker on Linux (page 16).
3	Create a user account on MyMVIS.	See Create a user account on MyMVIS (page 17).
4	Download and install MVIS Auto Annotation.	See Download and install MVIS Auto Annotation (page 18).
5	Optional: extend MVIS Auto Annotation for camera perception.	See Optional: extend MVIS Auto Annotation for camera perception (page 20).

2.3. Install Docker on Windows

Context

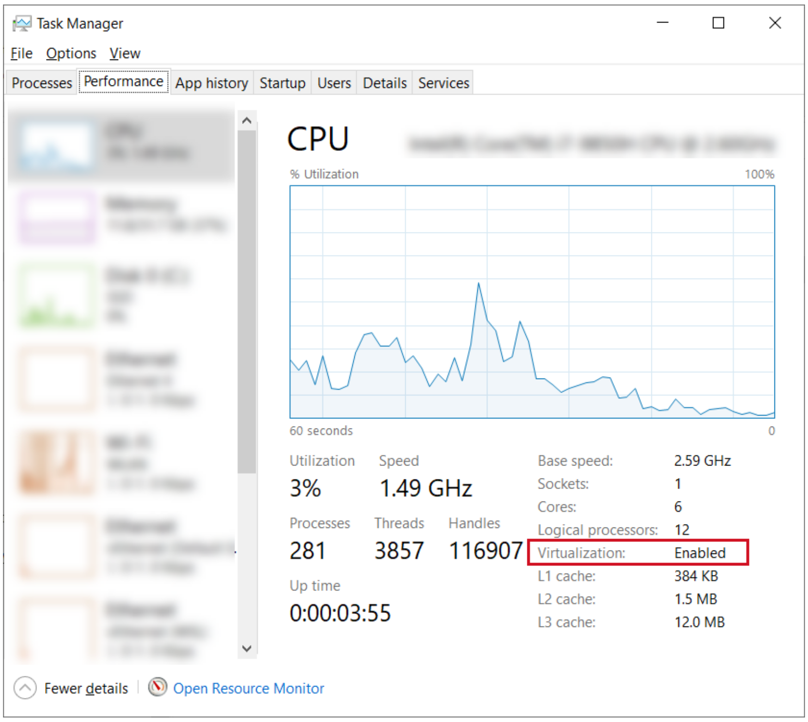
Before you can install Docker on Windows, you must install Windows subsystem for Linux (WSL 2) and set Windows features for Docker.

Step 1: install Windows subsystem for Linux (WSL 2)

1. Right-click on the Windows icon and select **Windows PowerShell (Admin)**.
2. Execute the following commands:
    - `dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart`
    - `dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart`
3. Restart your computer.
4. Download and install the Linux kernel update package from Microsoft: https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi
5. Right-click on the Windows icon and select **Windows PowerShell (Admin)**.
6. Execute the following command:
   `wsl --set-default-version 2`

Step 2: set Windows features for Docker

1. On Windows, run a search for the **Turn Windows features on or off** dialog.
2. In this dialog, activate the following checkboxes:
    - **Virtual Machine Platform**
    - **Hyper-V**
    - **Windows subsystem for Linux**
3. Restart your computer.
4. Start the Task Manager.
5. On the **Performance** tab, check if the **Virtualization** field displays **Enabled**.



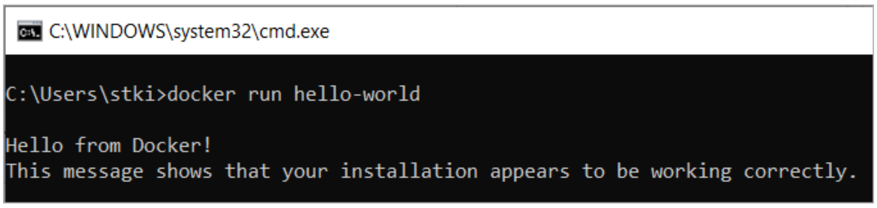
Step 3: download and install Docker for Windows

1. Download Docker for Windows:
   https://docs.docker.com/docker-for-windows/install/
2. Install Docker for Windows with the default settings.
3. Log out and log back in.
4. In the Docker Desktop settings, under **General**, deselect the **Use the WSL 2 based engine** checkbox.
5. Click **Apply & Restart**.

Step 4: run a test

1. Start the terminal.
2. Enter `docker run hello-world` and press **Enter**.

A success message displays `Hello from Docker!`



3. Close the terminal.

Step 5: deactivate "Resource saver mode"

1. In the Docker Desktop settings, under **Features in development**, select **Experimental features**.
2. Deselect the **Access experimental features** checkbox.
3. Click **Apply & restart**.

Step 6: increase usable RAM for Docker

> **NOTE**
> This step is only necessary if you deselected the **Use the WSL2 based engine** checkbox as described in Step 3: download and install Docker for Windows (page 15).

1. In the Docker Desktop settings, under **Resources → Advanced**, increase the memory to 20 GB.
2. Click **Apply & restart**.

2.4. Install Docker on Linux

1. Update the `apt` package index:

```
sudo apt-get update
```

2. Install packages to allow apt to use a repository over HTTPS:

```
sudo apt-get install \
     apt-transport-https \
     ca-certificates \
     curl \
     gnupg \
     lsb-release
```

3. Add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
-o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. Set up the stable repository:

```
echo \
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/
null
```

NOTE
The `lsb_release -cs` sub-command returns the name of your Ubuntu distribution, such as xenial. In a distribution like Linux Mint, you might need to change `$(lsb_release -cs)` to your parent Ubuntu distribution. For example, if you are using Linux Mint Tessa, you could use bionic. Docker does not offer any guarantees on untested and unsupported Ubuntu distributions.

5. Install Docker Engine:

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. Optional but recommended: if you do not want to run Docker with `sudo`:

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

7. Verify that Docker Engine is installed correctly by running the `hello-world` image:

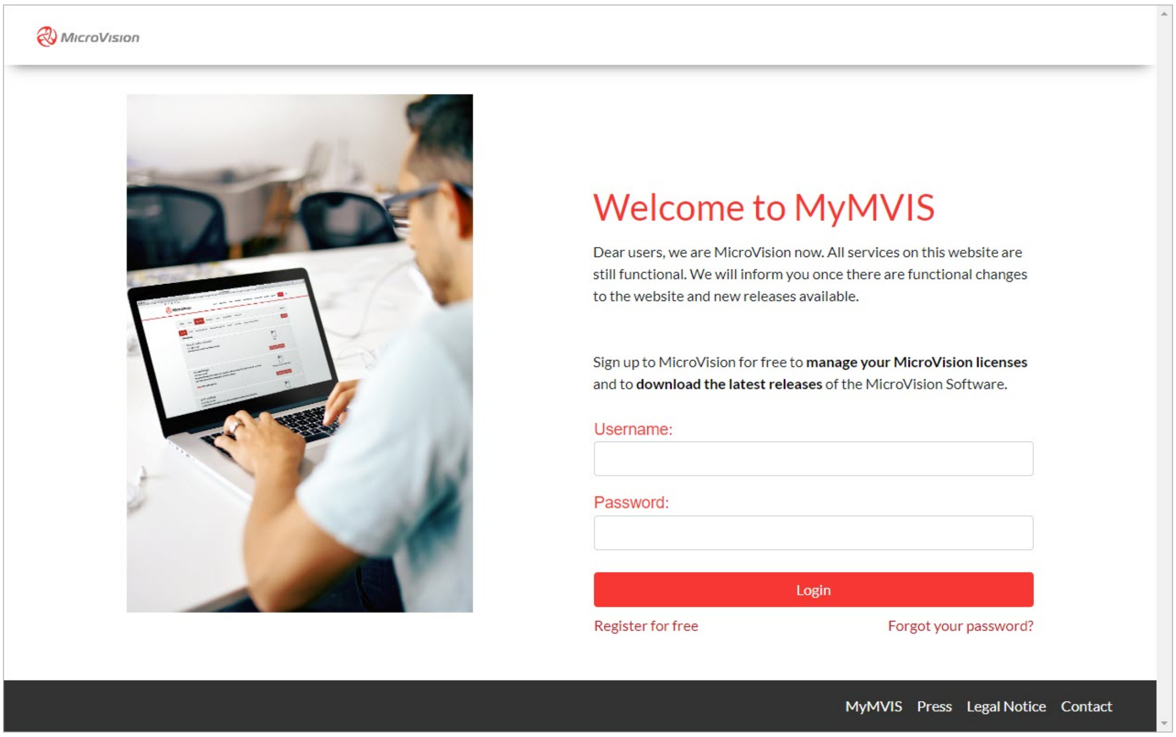
```
docker run hello-world
```

2.5. Create a user account on MyMVIS

Context
On MyMVIS, MicroVision's customer platform, you can download MicroVision software products and the related documentation, manage your licenses and get in touch with the MicroVision Support Team.

Procedure

1. Go to the website https://my.microvision.eu.



2. Click **Register for free**.
3. Follow the instructions.
   A confirmation e-mail is sent to your e-mail address.
4. In the confirmation e-mail, click the link.
   The MyMVIS user account is activated.

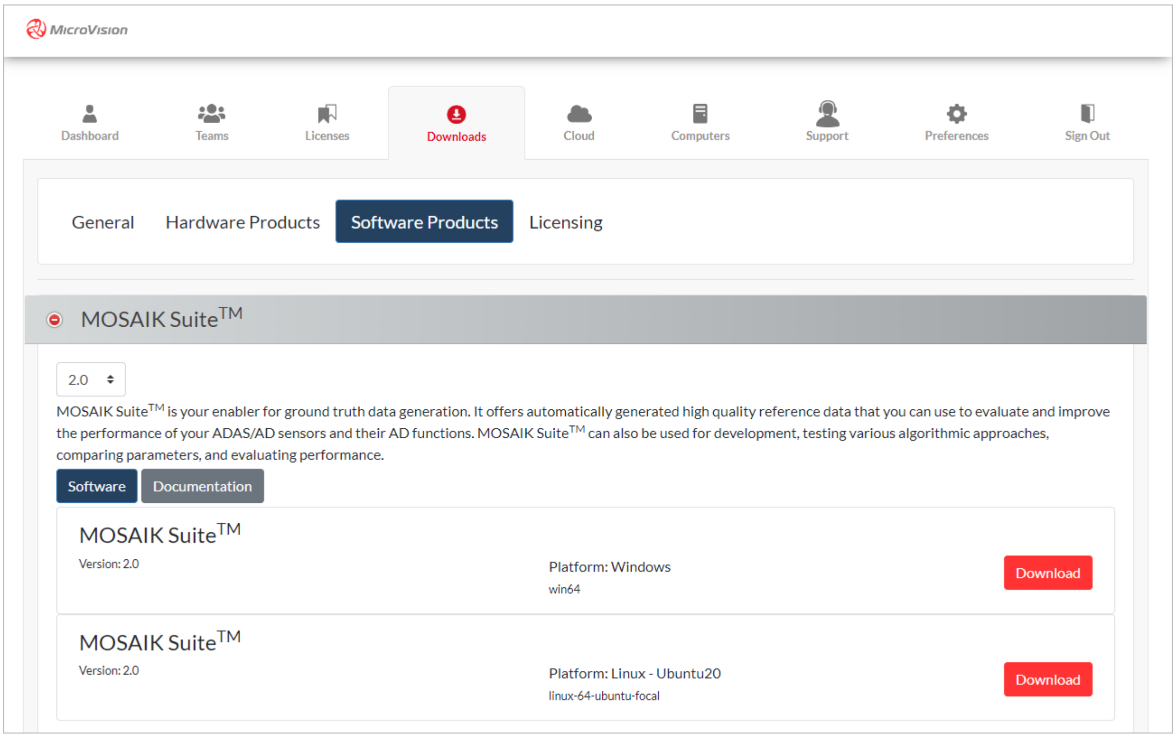
2.6. Download and install MVIS Auto Annotation

Prerequisites

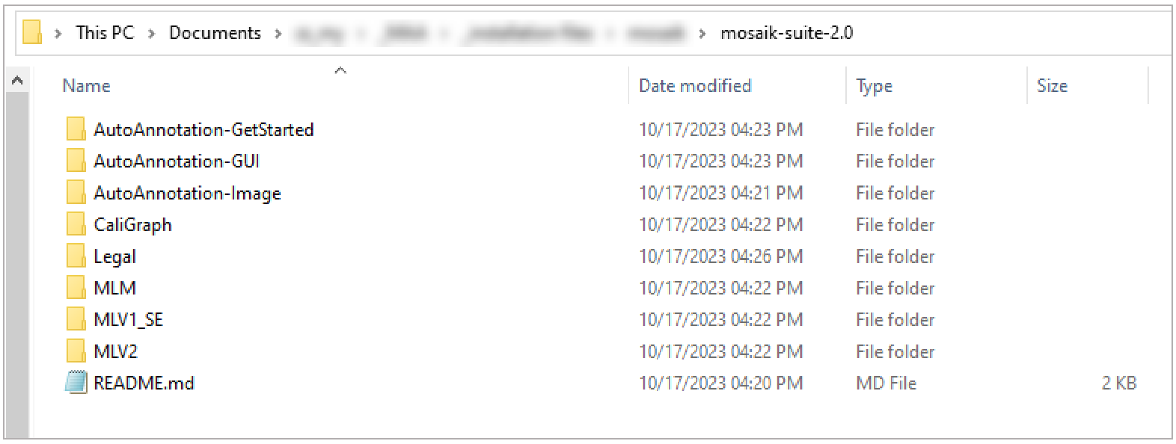
You have a user account on MyMVIS, see Create a user account on MyMVIS (page 17).

Procedure

1. Log in to MyMVIS: https://my.microvision.eu.
2. Click **Downloads → Software Products**.
3. Expand the **MOSAIK Suite™** section and select your version.



4. Download the MOSAIK Suite™ for your operating system.
5. Unzip the downloaded file. The file contains the following folders:



NOTE

The `AutoAnnotation-GetStarted` folder can be used to get started with the terminal version of MVIS Auto Annotation, see The terminal version of MVIS Auto Annotation (page 46).

6. Navigate to the MVIS Auto Annotation installation file, either in the `AutoAnnotation-GUI` folder or in the `AutoAnnotation-Image` folder.
7. Start the MVIS Auto Annotation installation file and follow the instructions.

2.7. Optional: extend MVIS Auto Annotation for camera perception

Context

MVIS Auto Annotation includes all means for lidar perception. If you want to additionally use camera perception for traffic signs and traffic lights recognition, MVIS Auto Annotation can be easily extended. The following steps will download all required tooling, extend your already installed container image, and optionally store the extended image to disk.

Procedure

1. Open the terminal.
2. Execute the `install.bat` or `install.sh` script with option `-e`.
   The scripts are located in the `AutoAnnotation-Image` folder, see Download and install MVIS Auto Annotation (page 18).
3. Optionally, type `-s` to store a copy of the extended image to disk. Example:
   `./install.sh -e -s`

> **NOTE**
> If MVIS Auto Annotation has not been installed on your computer yet, it will be automatically installed now.

3. Licensing

To work with MVIS Auto Annotation, you need an activated and valid user license.

A user license is connected to a user account on MyMVIS. Before you can activate your user license, you must register your user license key on MyMVIS. Your user license key is provided by MicroVision.

For more information on user licenses, refer to the *MVIS License Manager User Manual*, see Related documents (page 7).

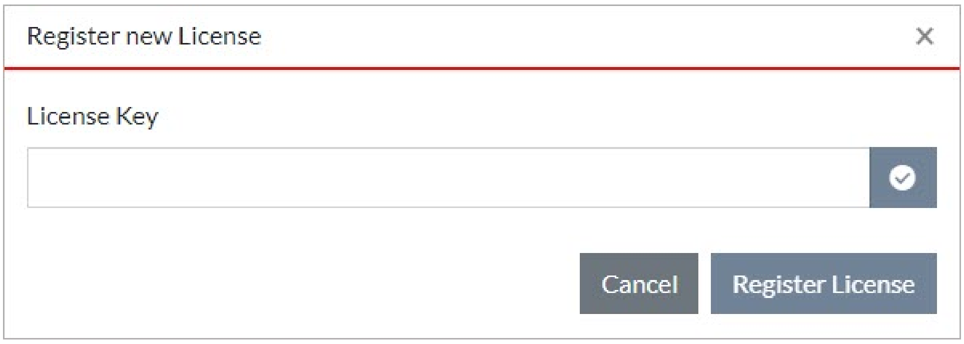
MVIS Auto Annotation has a GUI version and a terminal version. The activation process for these two versions differs:

- Activate your user license for the GUI version (page 21)
- Activate your user license for the terminal version (page 24)

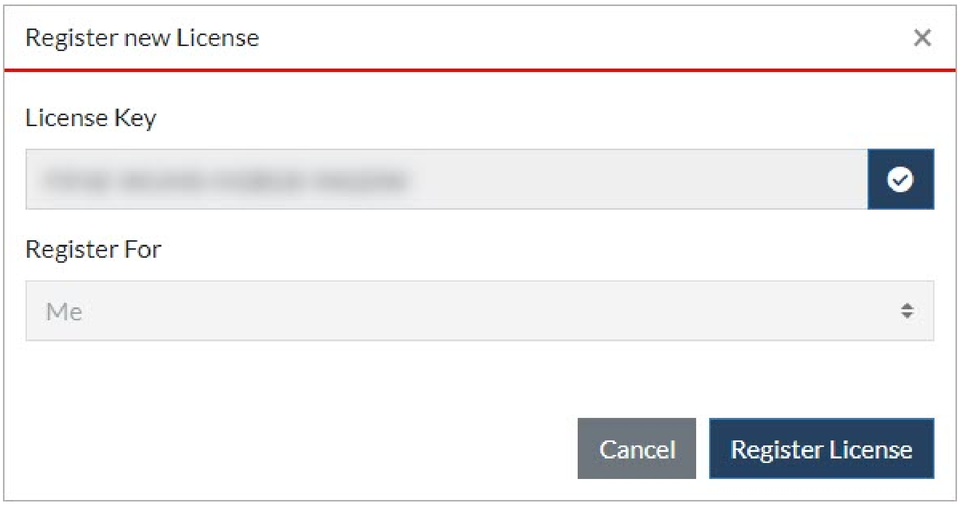
> **NOTE**
> For information on the license model including your credit balance, contact the MicroVision Support Team.

3.1. Activate your user license for the GUI version

Step 1: register your user license key

1. Log in to MyMVIS: https://my.microvision.eu.
   Use the user account to which the user license will be connected.
2. Click the **Licenses** tab.
3. Click **Register new license**.
   The **Register new license** dialog appears.
   

4. Enter the license key that you received from MicroVision and click **Enter**.



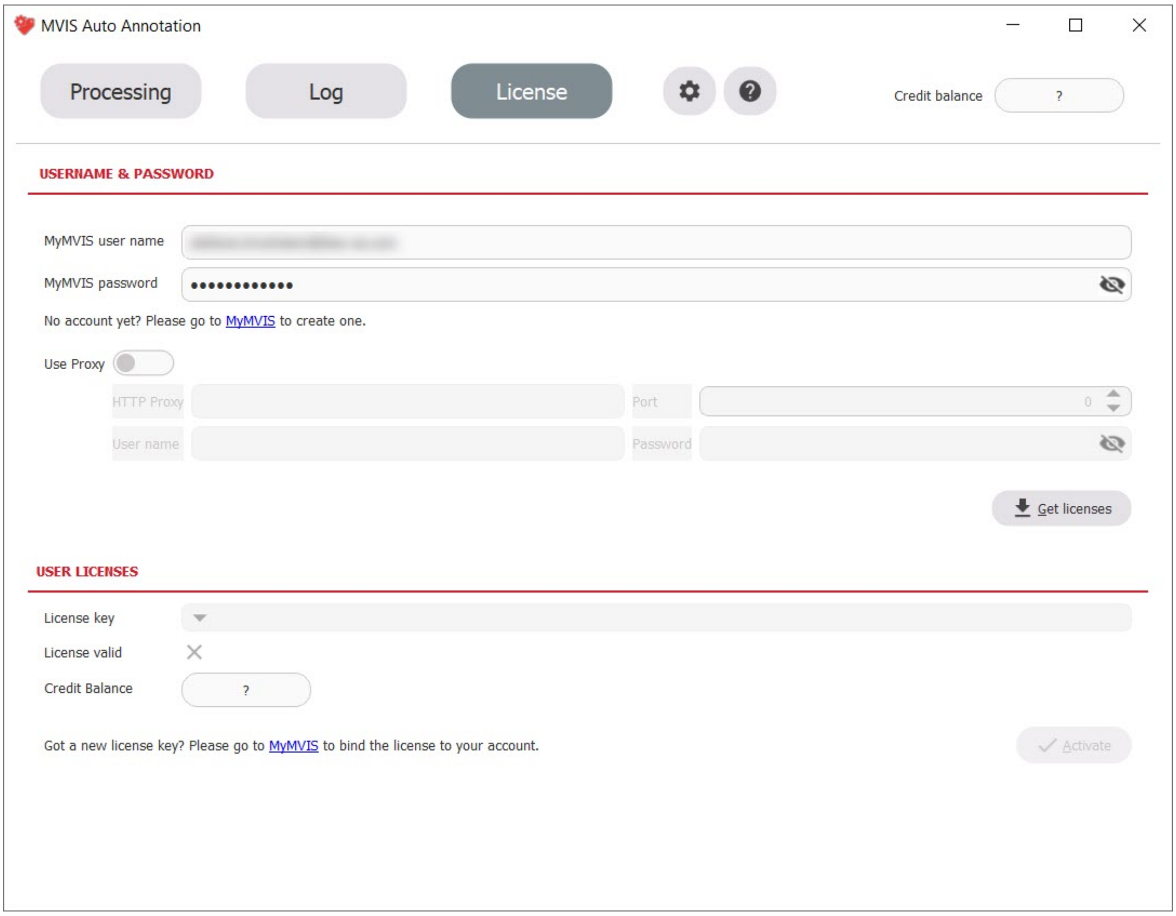
5. Click **Register license**.
   The license key is registered and connected to the current MyMVIS user account.

Step 2: activate your user license

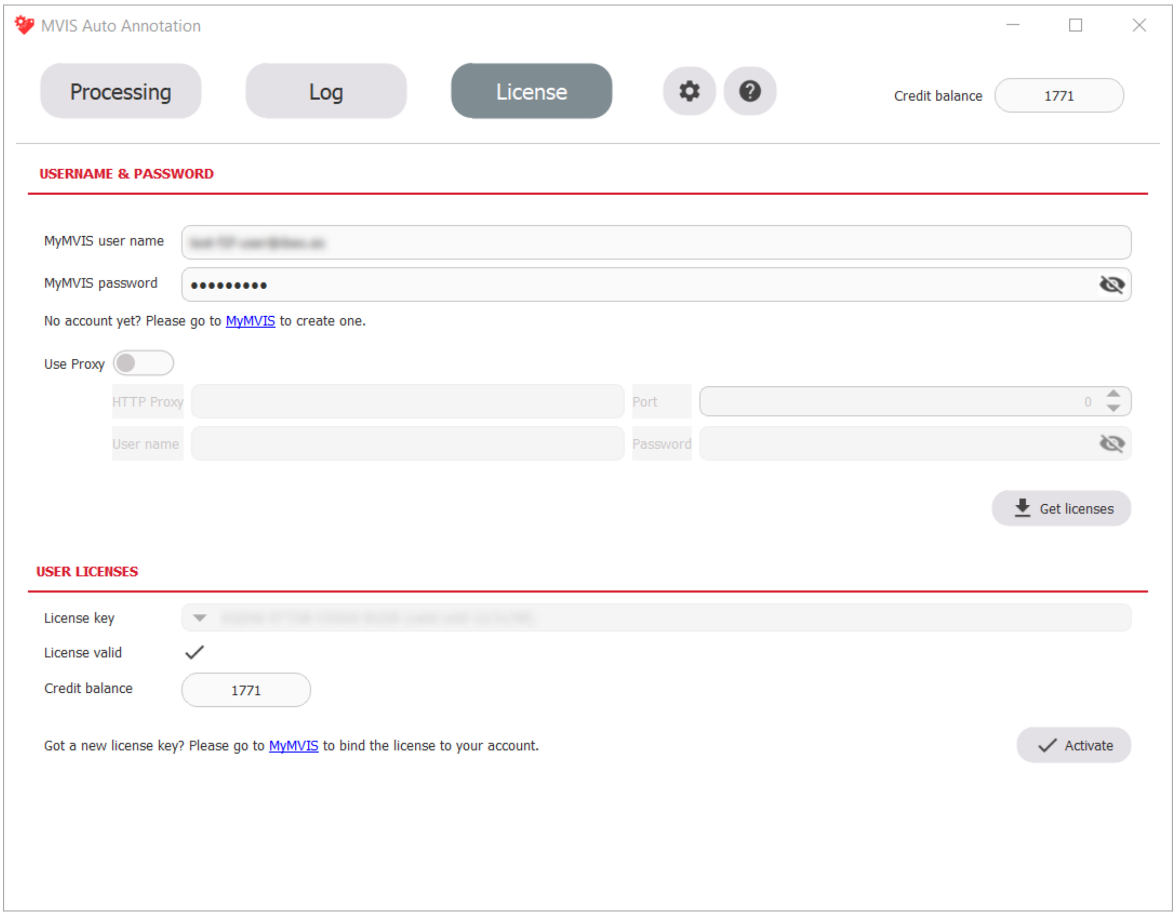
1. Start the GUI version of MVIS Auto Annotation.

   > **NOTE**
   > When you start MVIS Auto Annotation for the first time, a message informs you that a Docker image will be installed.

   The **License** tab appears.



2. Enter the username and the password of the MyMVIS user account to which your user license is connected.
3. If your IT system uses a proxy server for inbound and outbound Internet connections, the proxy settings must be specified.
   In this case, activate the **Use proxy** option and enter the proxy settings of your IT system.
4. Click **Get licenses**.
   The user license is displayed in the **User Licenses** section.
5. Click **Activate**.
   - A check mark is displayed in the **License valid** field.
   - Your credit balance is displayed in the **Credit Balance** field.



3.2. Activate your user license for the terminal version

You activate your user license for the terminal version in two steps.

Step		More information
1	Generate and export the licensing files.	Refer to the *MVIS License Manager User Manual*, see Related documents (page 7).
2	Insert the exported licensing files to the license folder.	Select a license folder that is accessible during trip processing. For more information on the license folder, see Target folders in the terminal (page 46).

4. Validity check for input files

Credits will be booked for all processed files, even if the input files are not valid and will thus produce poor output quality.

To avoid unusable processing results, validate the input files before you process them in MVIS Auto Annotation.

You can use MVIS Laser View to validate IDC input files.

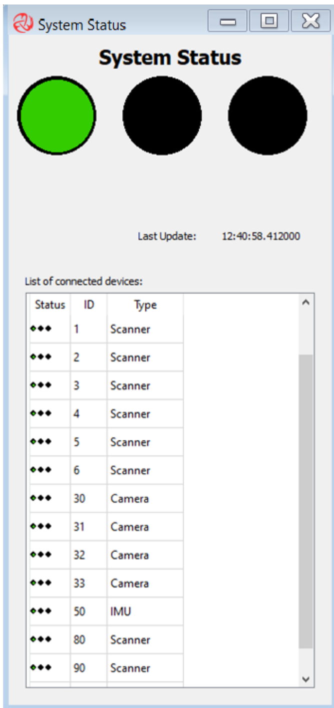
4.1. Validate IDC input files with MVIS Laser View

Context

If the values and statuses that are described in the following procedures do not apply, the input file should not be used for further processing with MVIS Auto Annotation. The input file may produce poor output quality.

Procedure for checking the system status

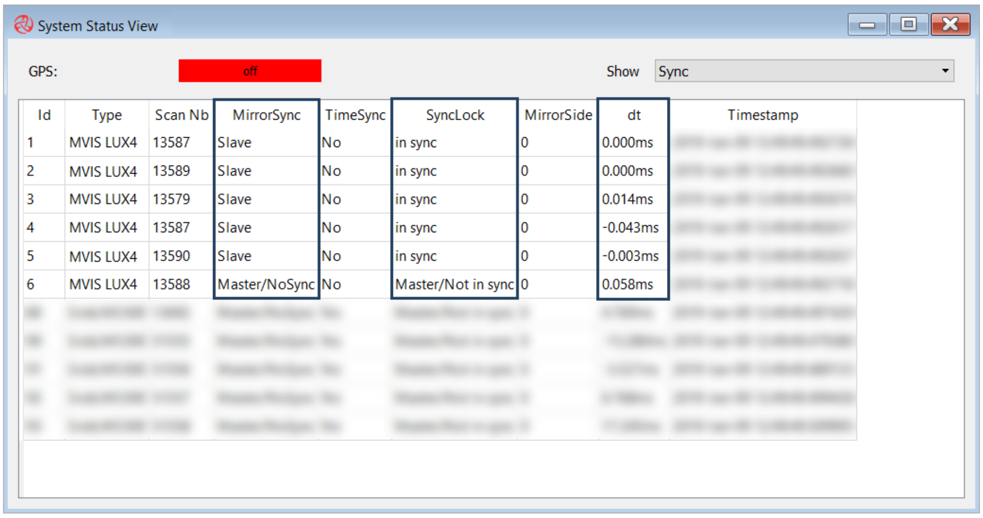
1. In MVIS Laser View, in the Menu bar, click **View → Show System Status**.
   The **System Status** dialog appears.



2. Check if the status of all components that are involved in the recording and the overall status display in green.

Procedure for checking the system status view

1. In MVIS Laser View, in the Menu bar, click **View → Enable System Status View.**
   The **System Status View** dialog appears.



2. Check the following settings:

Devices	Column	Value
One MVIS LUX sensor	**MirrorSync**	`Master/NoSync`
	SyncLock	`Master/Not in sync`
All remaining MVIS LUX sensors	**MirrorSync**	`Slave`
	SyncLock	`in sync`
All MVIS LUX sensors	**dt**	|dt| < 5 ms

Procedure for checking the vehicle status

1. In MVIS Laser View, in the Menu bar, click **View → Enable Vehicle Status View**.
   The **Vehicle Status** dialog appears.
2. Check if the vehicle status values, the vehicle speed, and the yaw rate values are plausible for the recording situation.
   Example: For a parking lot environment, a vehicle speed of 200 km/h is not plausible.

5. The GUI version of MVIS Auto Annotation

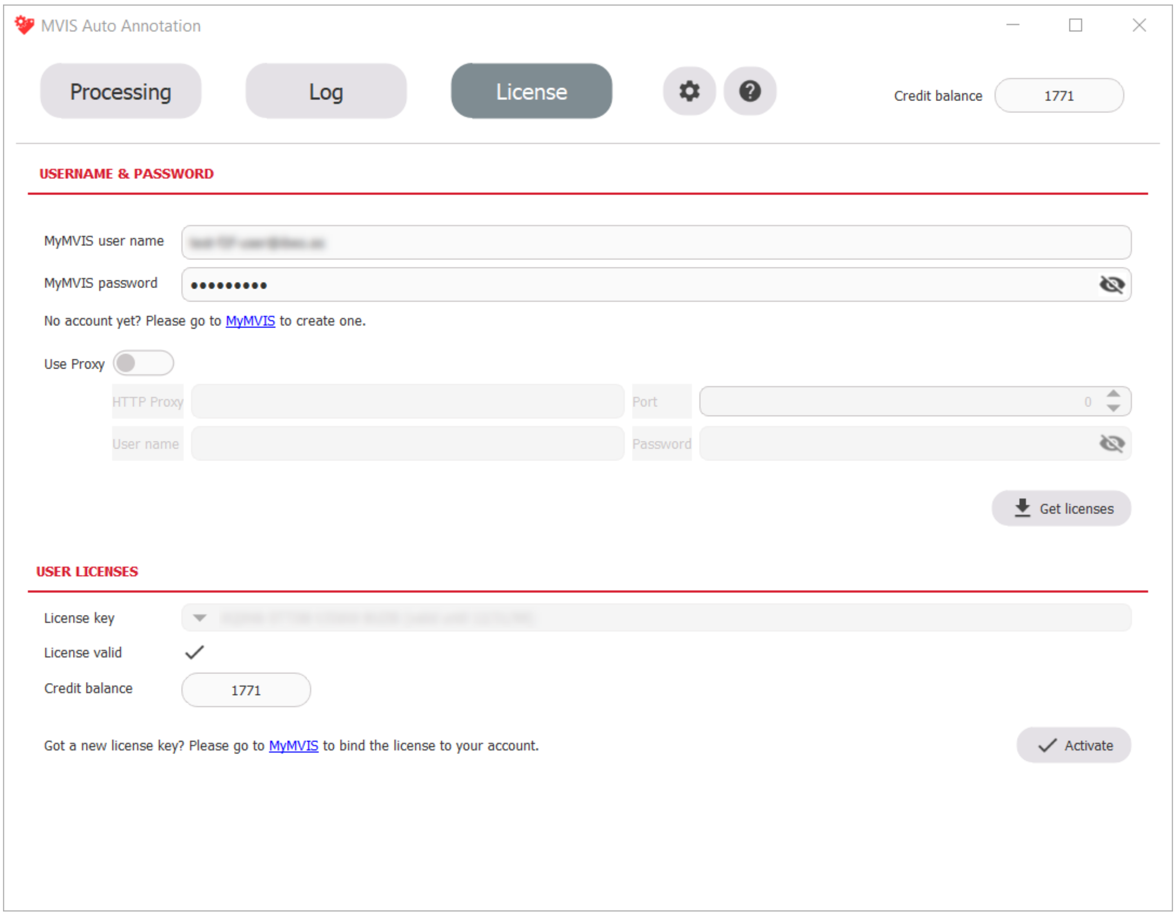
MVIS Auto Annotation has a GUI version and a terminal version.

The GUI version of MVIS Auto Annotation is intended to be used on a system that has a graphical user interface.

For many user interface elements, you can display a tooltip.

5.1. View your user license information

1. Start the GUI version of MVIS Auto Annotation.
2. Select the **License** tab.

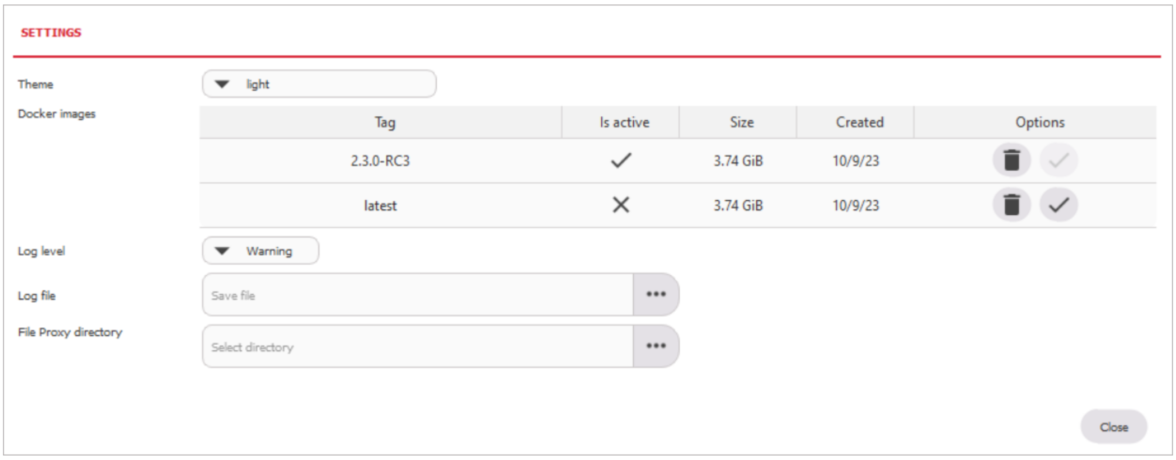


3. In the **Username and password** section, you view the MyMVIS user account to which your user license is connected. If applicable, you also view your proxy settings.
4. In the **User licenses** section, you view your user license key with expiration date, your credit balance and if your user license is valid.

5.2. Edit your user settings

1. Start the GUI version of MVIS Auto Annotation.

2. Click **Settings**.
   The **Settings** view appears.



3. You can edit the following settings:

Setting	Description
Theme	Select a theme: light or dark.
Docker images	• View which Docker images are in use or inactive. • Delete a Docker image.
Log level	Only applicable when contacting the MicroVision Support Team: Select a log level for the GUI version of MVIS Auto Annotation.
Log file	Only applicable when contacting the MicroVision Support Team: Select the location where the log file will be saved.
File proxy directory	For information on this feature, contact the MicroVision Support Team.

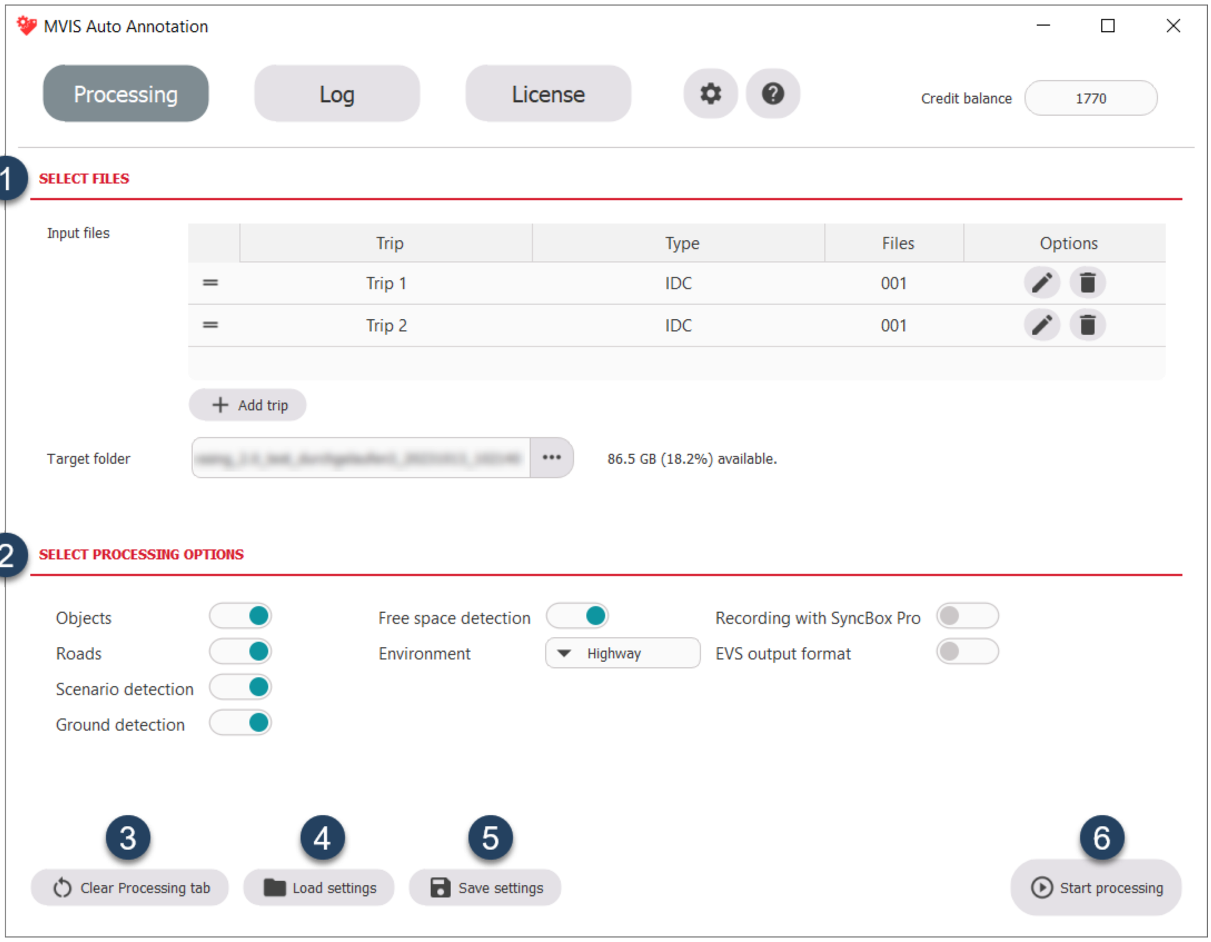
4. Click **Close**.

5.3. Settings for a trip processing

Before you start a trip processing, specify the settings for the trip processing.

These settings include the input files that you want to process, a target folder for the files that will be generated during the trip processing, and processing options.

5.3.1. Elements of the "Processing" tab (settings mode)



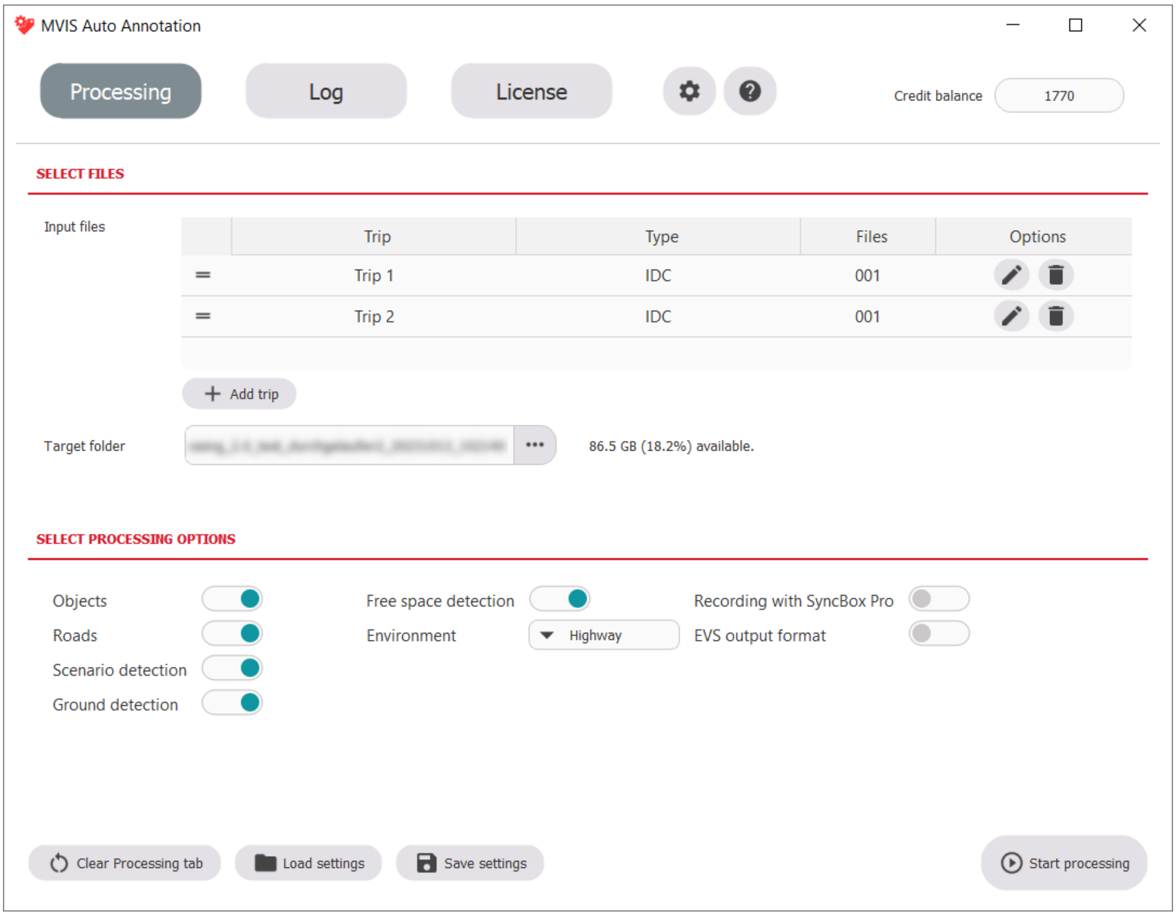
No.	Element	Description
1	**Select files** section	In this section, you select the input files and the target folder for the trip processing. For more information, see Specify the settings for a trip processing (page 30).
2	**Select processing options** section	In this section, you set the processing options for the trip processing. For more information, see List of processing options in the "Processing" tab (page 30).
3	**Clear Processing tab** button	Clears the input files, the target folder, and resets the processing options.
4	**Load settings** button	Loads saved settings. Settings are saved as an INI file.
5	**Save settings** button	Saves the current settings.
6	**Start processing** button	Starts a trip processing with the current settings. For more information, see Start a trip processing (page 37).

5.3.2. List of processing options in the "Processing" tab

Processing option	Included options	Output format	Description
Objects	–	IDC CORR	Creates a file with dynamic objects that were automatically detected, tracked, and classified. In addition, a correction file is created that you can use in MVIS Laser View for Smart Editing. For more information on object tracking, see Object tracking (page 64).
Roads	–	XODR (OpenDRIVE) IDC	Creates a file with lane markings, road boundaries, traffic signs, and traffic lights that were automatically detected. In addition, tile maps are created that you can use in MVIS Laser View 2 for Smart Editing. For more information on road detection, see Road detection (page 68).
Scenario detection	**Objects** **Roads**	CSV	Creates a file with scenarios that were automatically detected. For more information on scenario detection, see Scenario detection (page 82).
Ground detection	–	IDC	Creates a file with the ground model that was automatically detected. FFor more information on ground detection, see Ground detection (page 61).
Free space detection	**Objects**	CSV	Creates a file with the drivable free space that was automatically detected. For more information on free space detection, see Free space detection (page 77).
Environment: • **City** • **Rural** • **Highway**	–	–	To improve the quality of the output select the environment that matches your recording.
Recording with SyncBox Pro	–	–	Select this option if new MVIS SyncBox Pro was used for time synchronization of sensors.
EVS output format	–	IDC	Select this option to convert the new 3D ego state (`0x2809 VehicleState` data type) to 2D ego state (`0x2808 VehicleState` data type) like it was produced by ibeo Evaluation Suite. For more information, refer to the *MVIS Data Interface Specification*, see Related documents (page 7).

5.3.3. Specify the settings for a trip processing

1. Start the GUI version of MVIS Auto Annotation.
   The **Processing** tab appears.



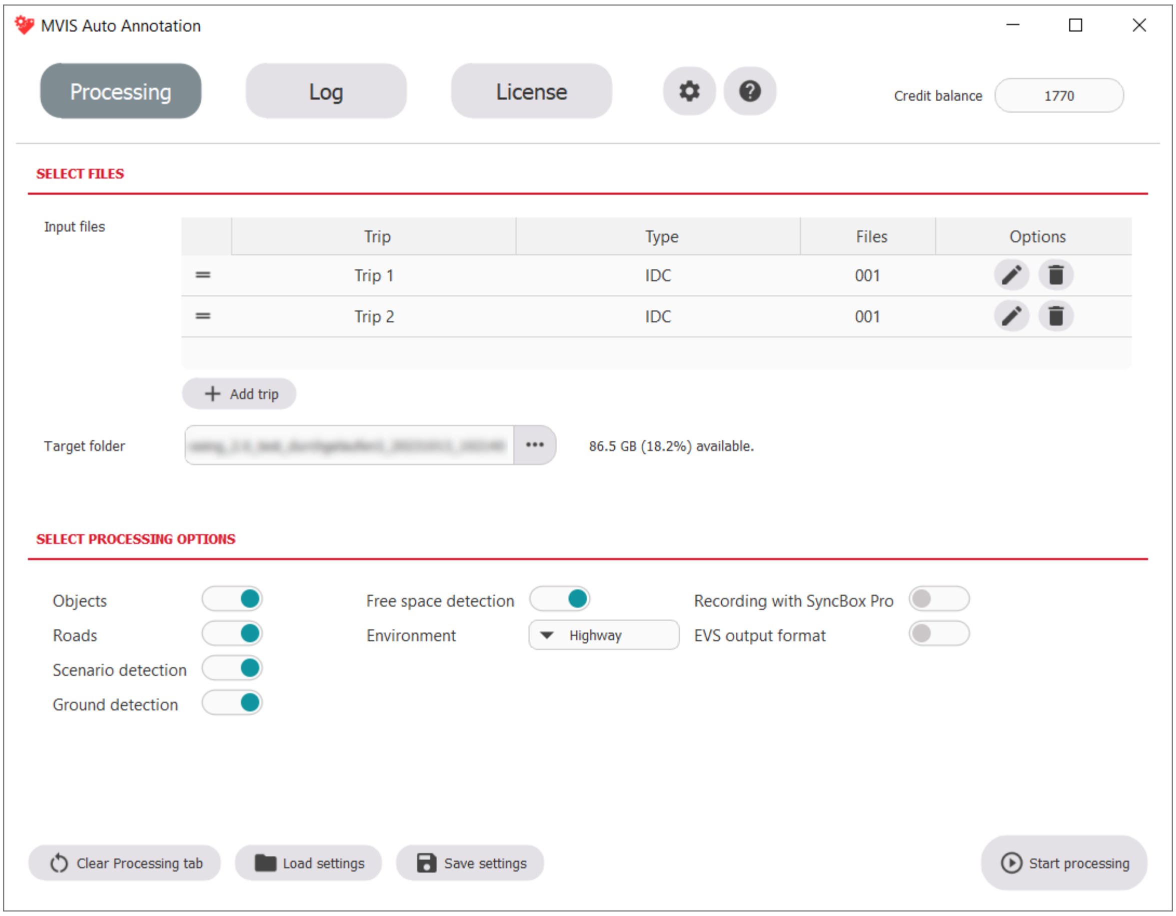
2. The settings of the last session are displayed. You can change the settings as follows:

Option	Steps
Keep the current settings.	—
Keep the input files and select different options.	1. In the **Target folder** field, select a folder for the files that will be generated during the trip processing. The folder must be empty. 2. Select processing options, see List of processing options in the "Processing" tab (page 30).
Load previous settings.	1. Click **Load settings.** 2. Select an INI file with previous settings.
Enter new settings.	1. Click **Clear Processing tab**. 2. Add at least one trip, see Add a trip to a trip processing (page 32). 3. In the **Target folder** field, select a folder for the files that will be generated during the trip processing. The folder must be empty. 4. Select processing options, see List of processing options in the "Processing" tab (page 30).

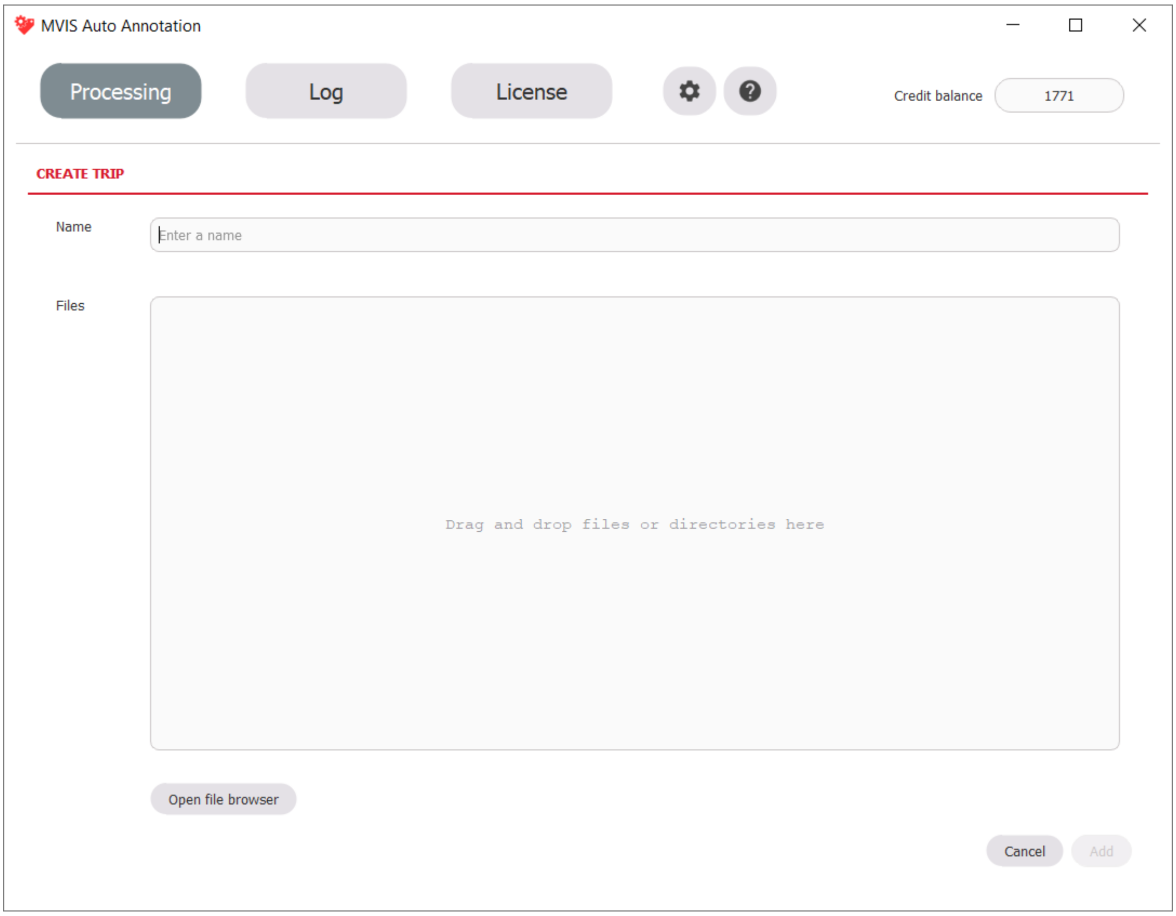
3. You can save your settings as an INI file. To do so, click **Save settings**.

5.3.4. Add a trip to a trip processing

1. Perform a validity check to see if your input files are valid, see Validity check for input files (page 25).
2. Start the GUI version of MVIS Auto Annotation.
   The **Processing** tab appears.



3. Keep the settings or click **Load settings** to load the settings you want to add a trip to.
4. In the **Select files** section, click **Add trip**.
   The **Create trip** view appears.

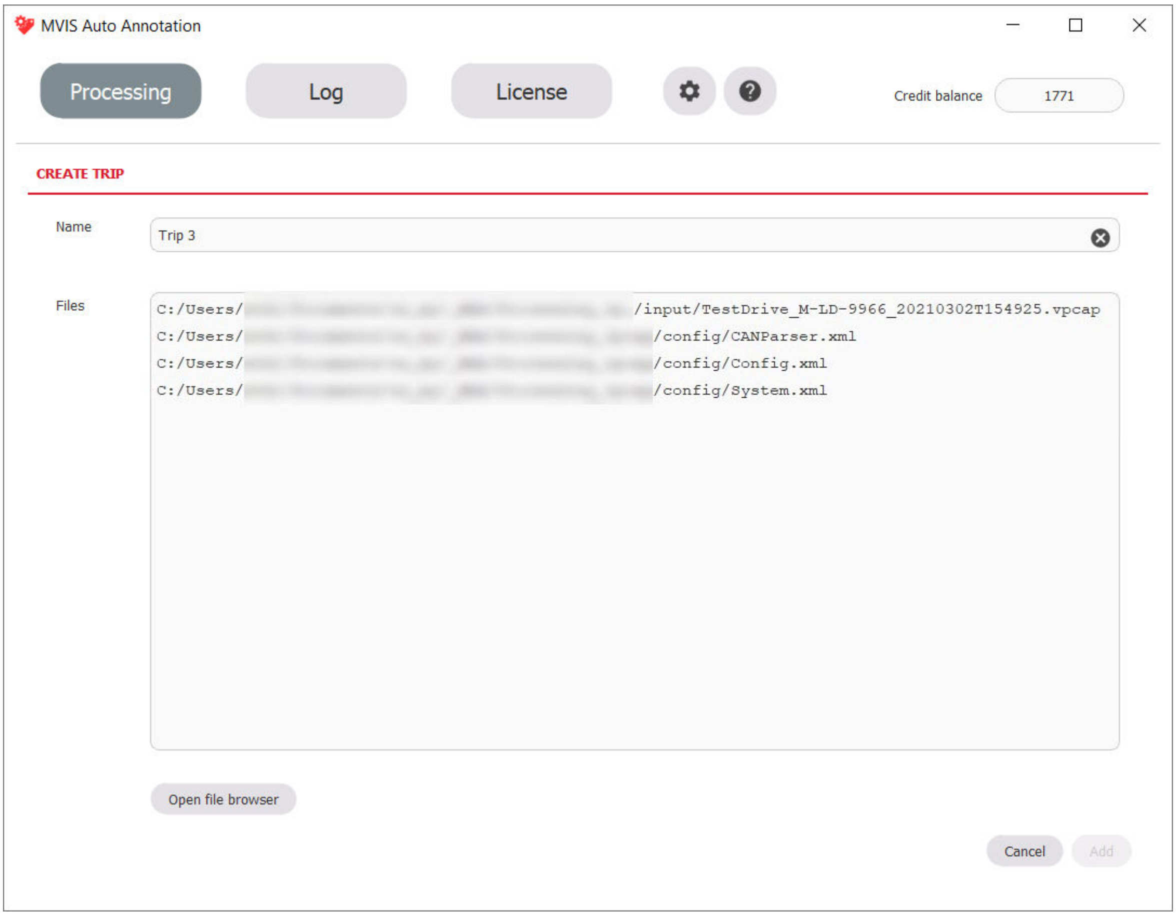


5. In the **Name** field, enter a name for the trip.
6. In the **Files** field, select the files that you want to process, either by drag and drop or by clicking **Open file browser**.
   All selected files must have the same file format, for example, IDC or VPCAP. For information on the supported file formats, see List of supported file formats (page 8).

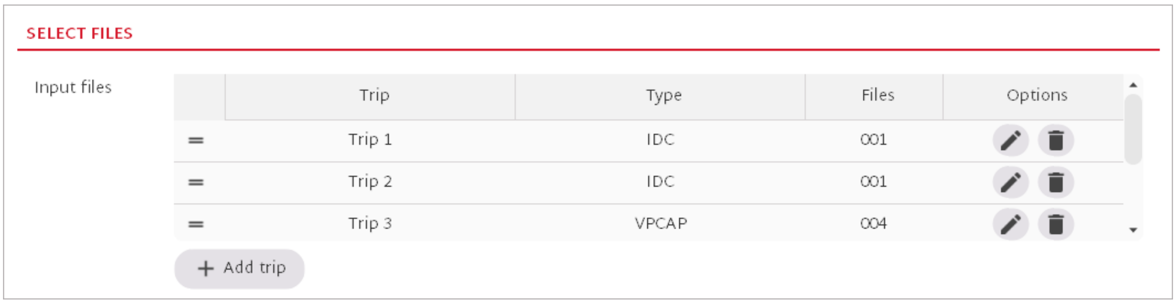
   NOTE

   To remove a file, click the file and press **Del**, or right-click the file and select **Delete**.

7. If the selected files require supporting configuration files, select these files as well, for example, for a VPCAP file:



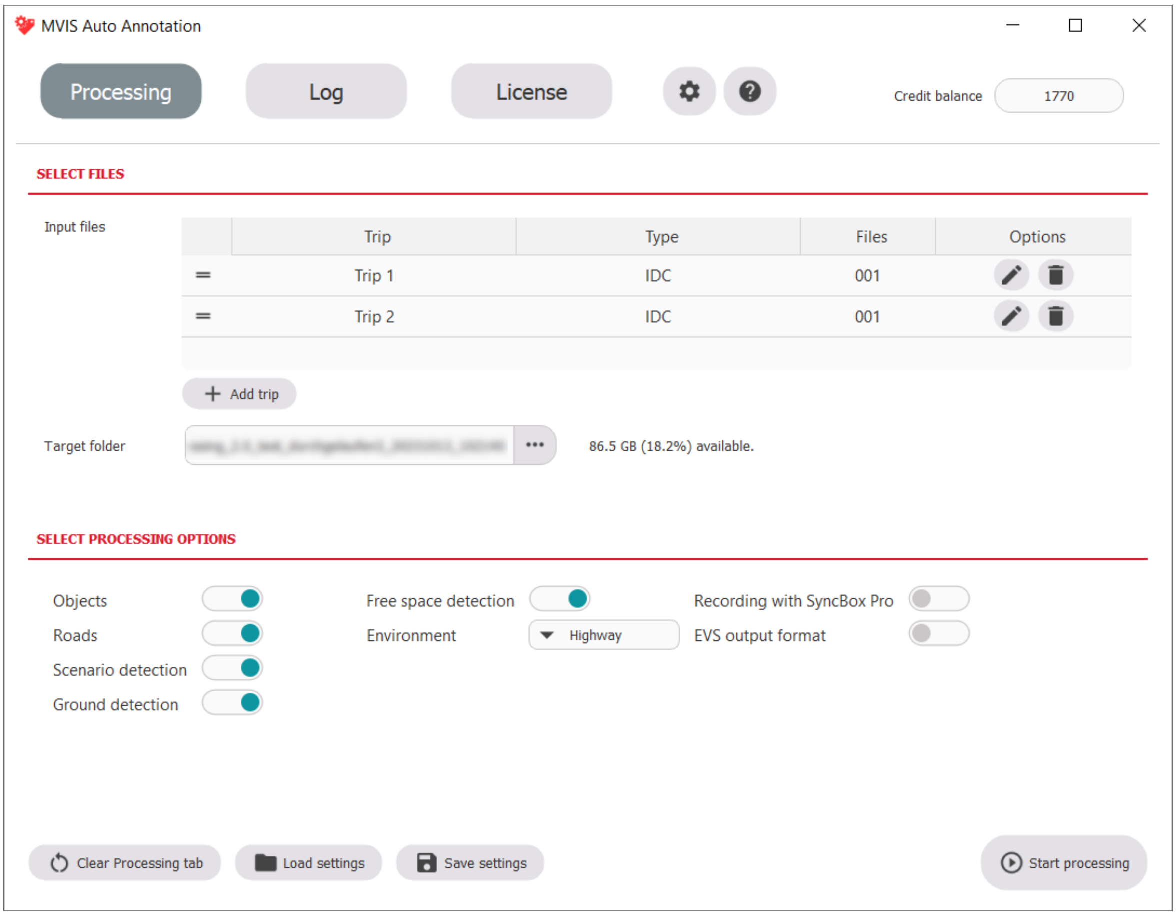
8. Click **Add**.
   The trip is added at the end of the **Input files** list.



9. To change the processing order of the trips, drag and drop the ═ icon next to a trip to the new position.

5.3.5. Remove a trip from a trip processing

1. Start the GUI version of MVIS Auto Annotation.
   The **Processing** tab appears.

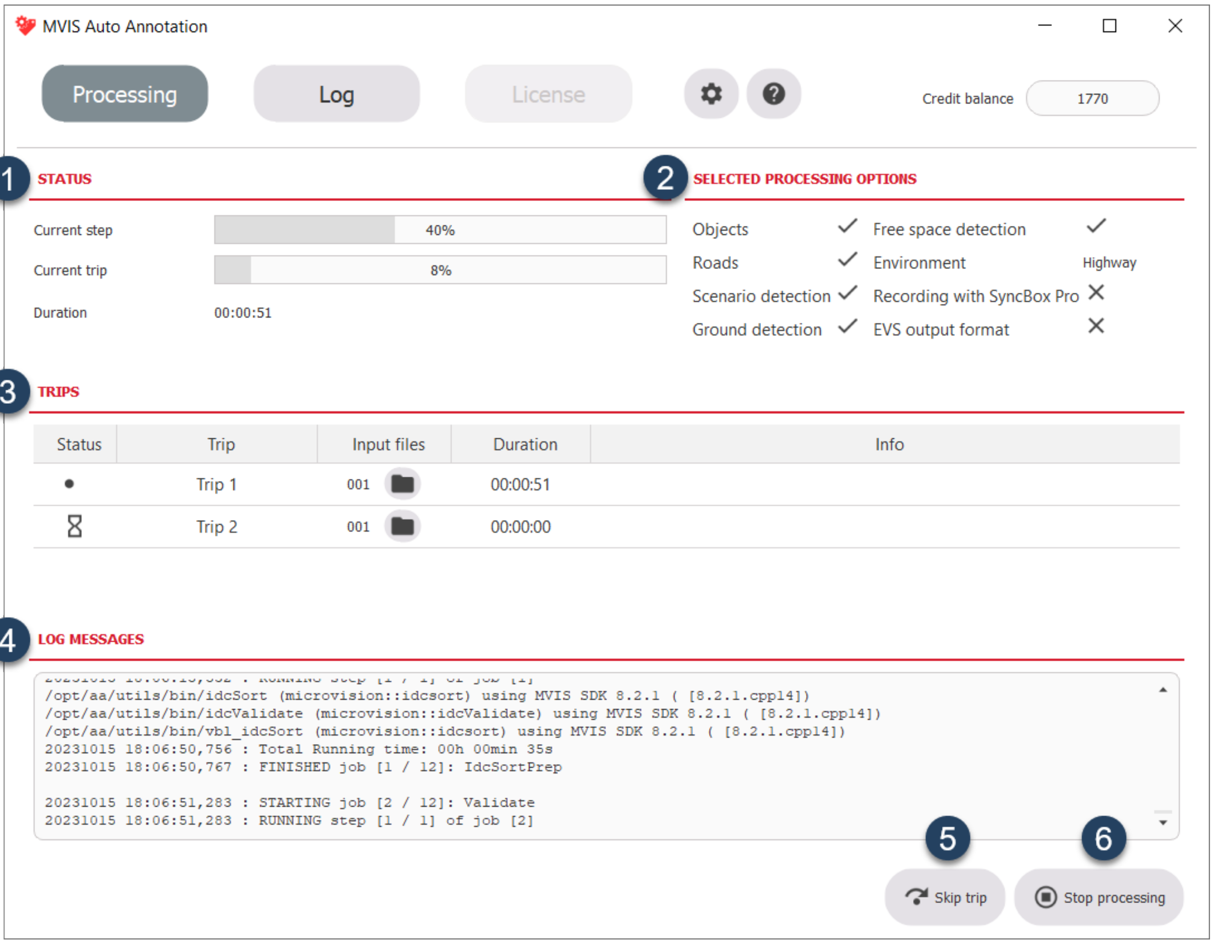


2. Keep the settings or click **Load settings** to load the settings you want to remove a trip from.
3. In the **Select files** section, next to a trip, click **Delete**.
4. Confirm the message that follows.

5.4. Trip processing

After you specified the settings for a trip processing, you can start the trip processing. For each trip, a status is displayed. You can skip a trip or stop the trip processing.

5.4.1. Elements of the "Processing" tab (processing mode)



No.	Element	Description
1	**Status** section	Displays the progress and duration of the trip processing.
2	**Selected processing options** section	Displays the processing options that were selected for the trip processing.
3	**Trips** section	Displays the trips that were selected for the trip processing. The **Status** column displays the current state of a trip, see List of trip statuses (page 37). The **Input files** column displays the number of input files that are included in a trip. Click **Folder** to view the files.
4	**Log messages** section	Displays the log messages for the current trip processing. For more detailed log information, select the **Log** tab, see Logging (page 39).
5	**Skip trip** button	See Stop the currently processed trip (page 38).
6	**Stop processing** button	See Start a trip processing (page 37).

5.4.2. List of trip statuses

Icon	Trip status	Description
	In progress	The trip is being processed.
	Waiting	The trip is waiting to be processed.
	Successful	The trip was successfully processed.
	Failed	The trip processing failed, for example, because there was not enough free space left on the disk. For further analysis, send the `summary.zip` file for this trip to the MicroVision Support Team, see View the "summary.zip" file for a processed trip (page 41).
	Stopped	The trip processing was started and then stopped or skipped by the user.
	Skipped	The trip was skipped before the processing of the trip started.

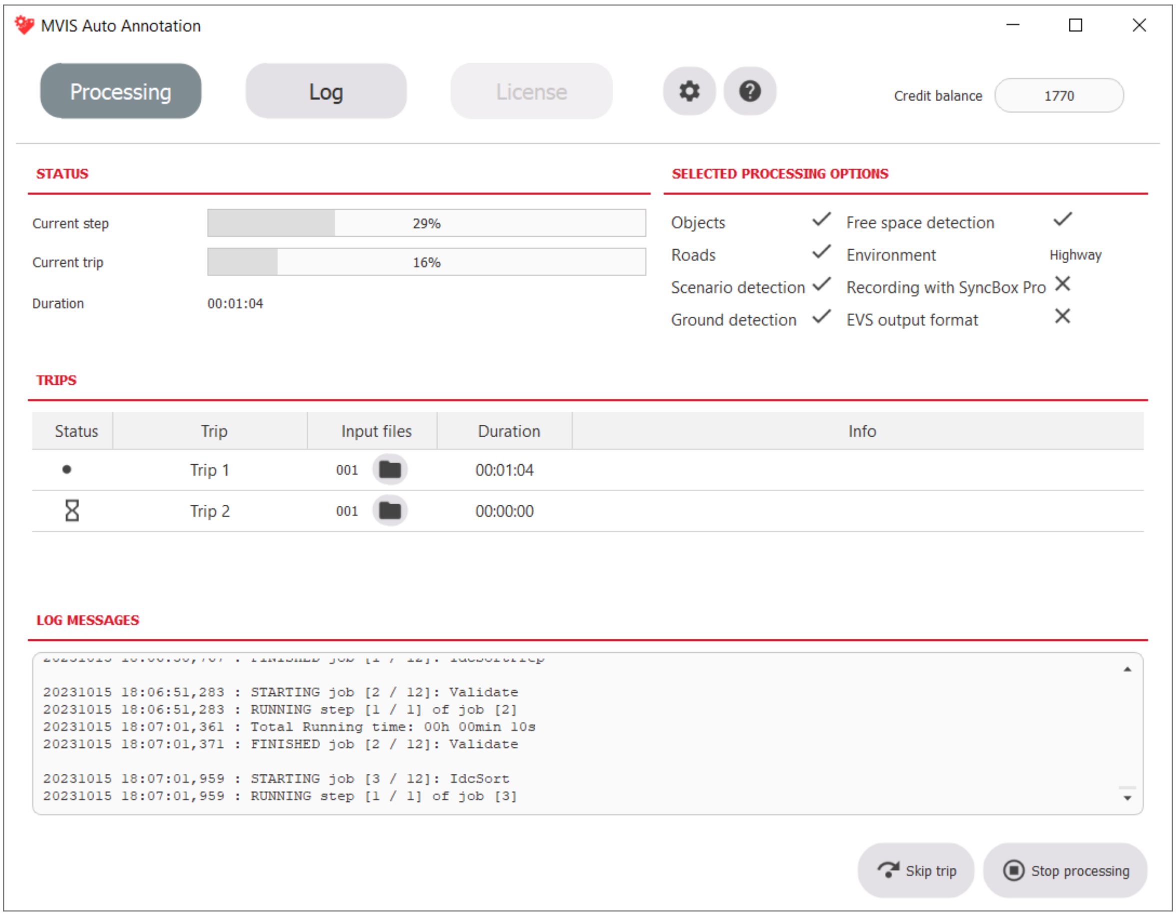
5.4.3. Start a trip processing

Prerequisites

- Your computer has Internet access.
- You have enough credits left (**Credit balance** field).
- All input files are valid, see Validity check for input files (page 25).

Procedure

1. Specify the settings for the trip processing, see Specify the settings for a trip processing (page 30).
2. Click **Start processing**.
3. If the specified target folder is not empty, a new target folder is suggested. Click **Ok** to use the suggested target folder or click **Cancel** and select an empty target folder.
   The trip processing starts.



4. View information on the current state of the trip processing, see Elements of the "Processing" tab (processing mode) (page 36).

5. When all trips were processed or the processing was stopped, click **Back** to go back to the trip processing settings.

5.4.4. Stop the currently processed trip

1. During a trip processing, select the **Processing** tab.

   The trip that is currently being processed has the status ● **In progress**.

2. Click **Skip trip**.

3. Confirm the message that follows.
   The trip gets the status ⊗ **Stopped**.

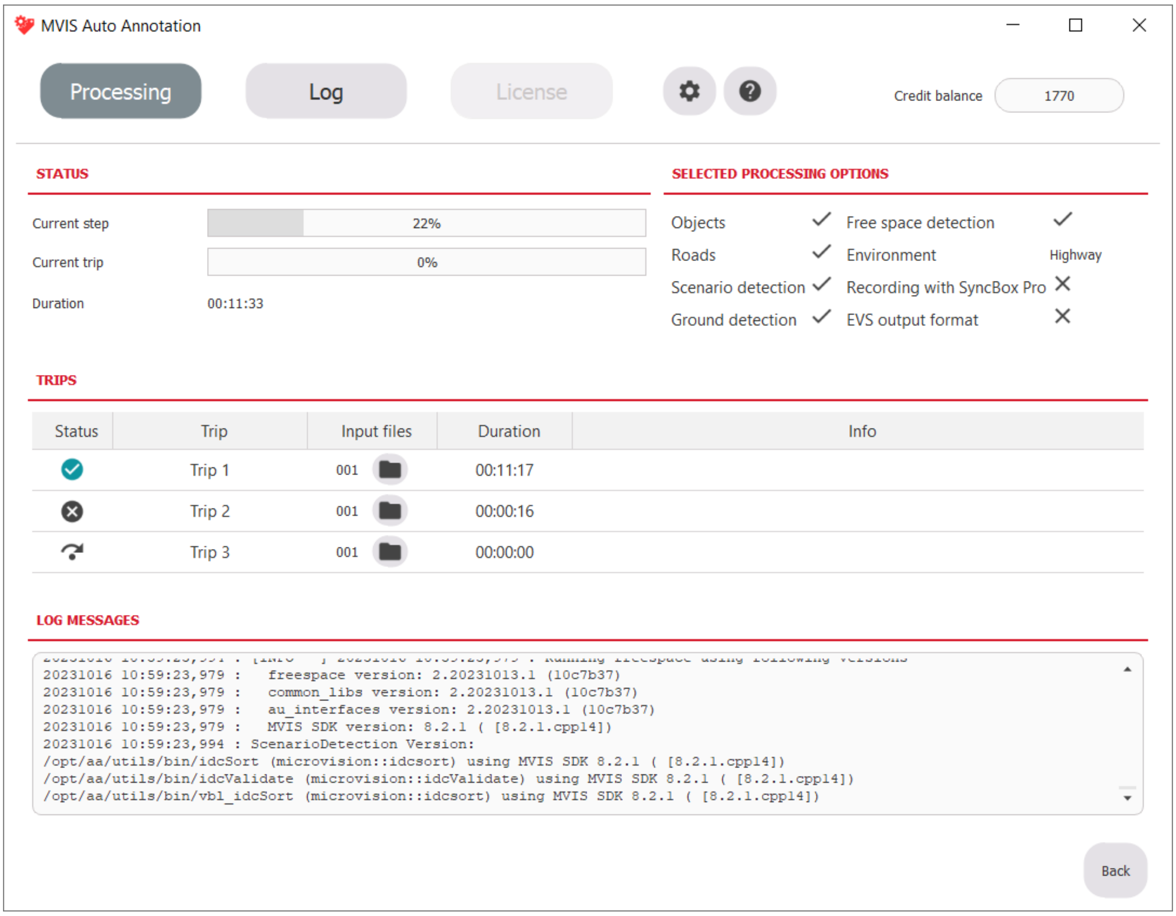
5.4.5. Stop the complete trip processing

1. During a trip processing, select the **Processing** tab.

2. Click **Stop processing**.

3. Confirm the message that follows.
   The complete trip processing is stopped.

- Trips that were already processed keep their status, either **Successful**, **Failed**, or **Stopped**.
- The trip that was being processed gets the status **Stopped**.
- The remaining trips get the status **Skipped**.



4. Click **Back** to go back to the trip processing settings.

5.5. Logging

MVIS Auto Annotation automatically generates and saves log information for each trip processing. You can view and copy the log information for further analysis.

When you start a new trip processing, the previous log message on the **Processing** tab and on the **Log** tab will be deleted.

For log information on a previous trip processing, view the automatically saved log files.

5.5.1. Elements of the "Log" tab

No.	Element	More information
1	**Trip** field	Filters the log message by a trip.
2	**Component** field	Filters the log message by a component.
3	**Wrap** slider	Wraps the log message.
4	**Scroll lock** slider	Prevents scrolling when a new log message is generated.
5	Log message field	Displays the log message for the selected trip and component.
6	**Clipboard** button	Copies the log message to the clipboard.
7	**Open** button	Opens the log message in the default viewer.

5.5.2. View the log messages for the current trip processing

1. During a trip processing, select the **Log** tab.
2. Select the trip and the component for which you want to view the log message.
   For more information, see Elements of the "Log" tab (page 40).

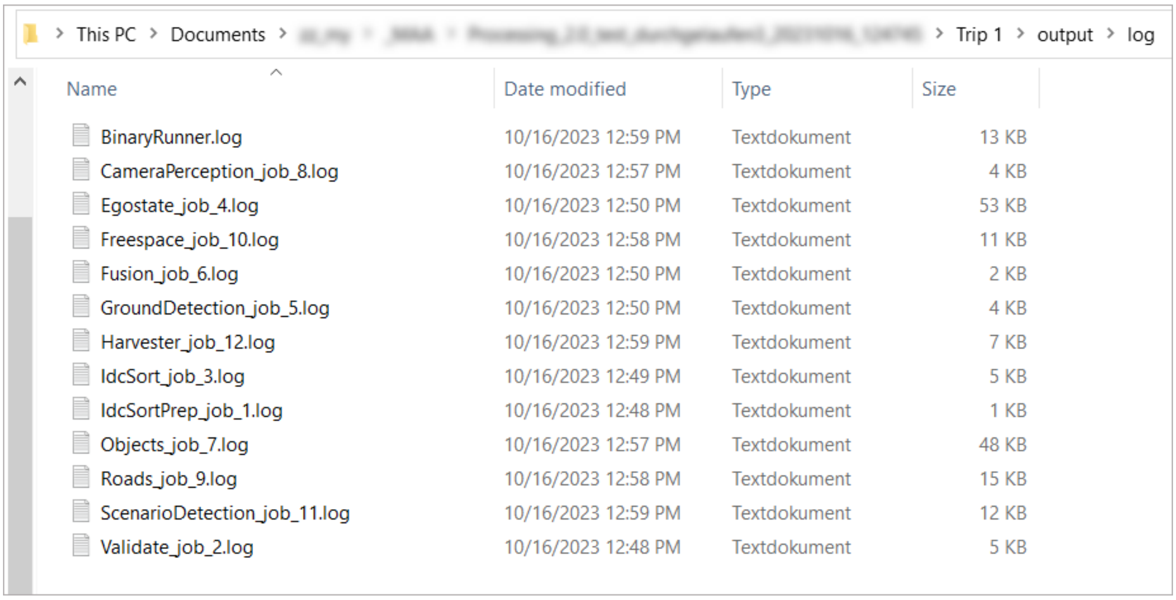
5.5.3. View the log files for a processed trip

Context

For each trip in a trip processing, log files are automatically saved in the `log` folder. This applies to the GUI version and the terminal version.

Procedure

1. Navigate to the target folder that you specified for the trip processing.
2. Select the folder for one of the processed trips, for example `Trip 1`.
3. In the `output` folder, select the `log` folder.
   All log files for the selected trip are displayed.



5.5.4. View the "summary.zip" file for a processed trip

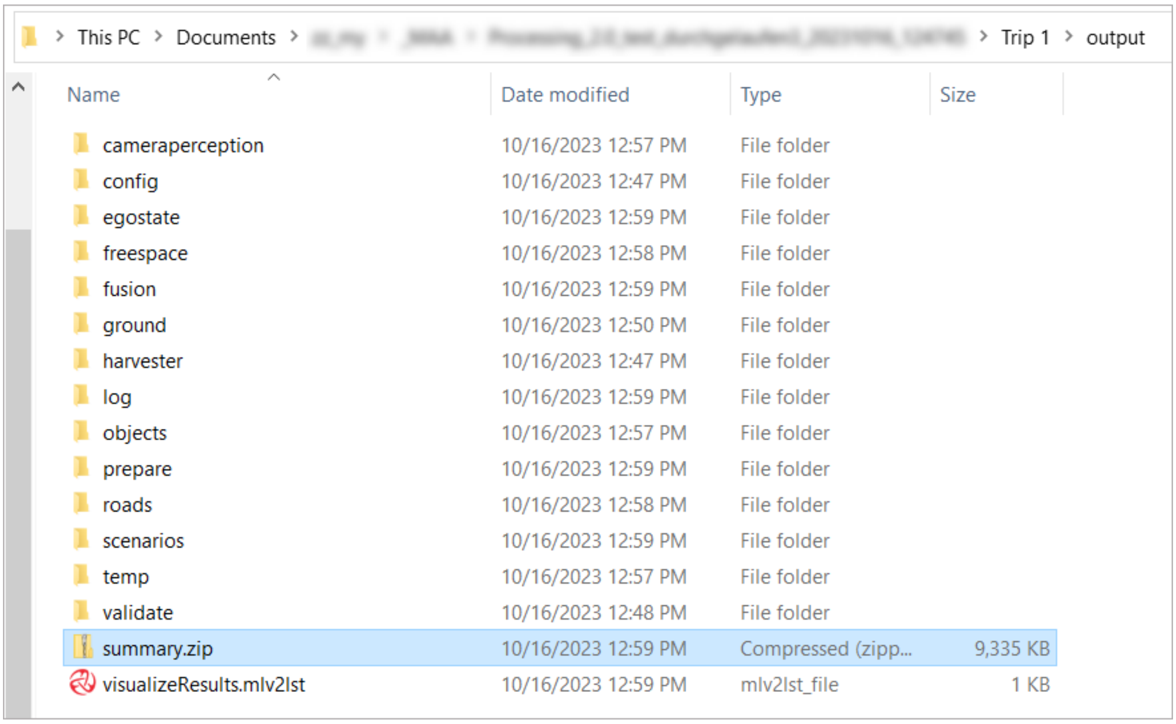
Context

All log files of a processed trip are saved in the `summary.zip` file. This applies to the GUI version and the terminal version.

In case of errors, send the `summary.zip` file to the MicroVision Support Team.

Procedure

1. Navigate to the target folder that you specified for the trip processing.
2. Select the folder for one of the processed trips, for example `Trip 1`.
3. Select the `output` folder.
   The `summary.zip` file for the selected trip is displayed.



5.6. Visualize the output of a trip with MVIS Laser View 2

Context

After a successful processing with MVIS Auto Annotation, multiple files are created that contain the different outputs such as objects and lanes. These results can be visualized with MVIS Laser View 2. Depending on the output type, you have the following options:

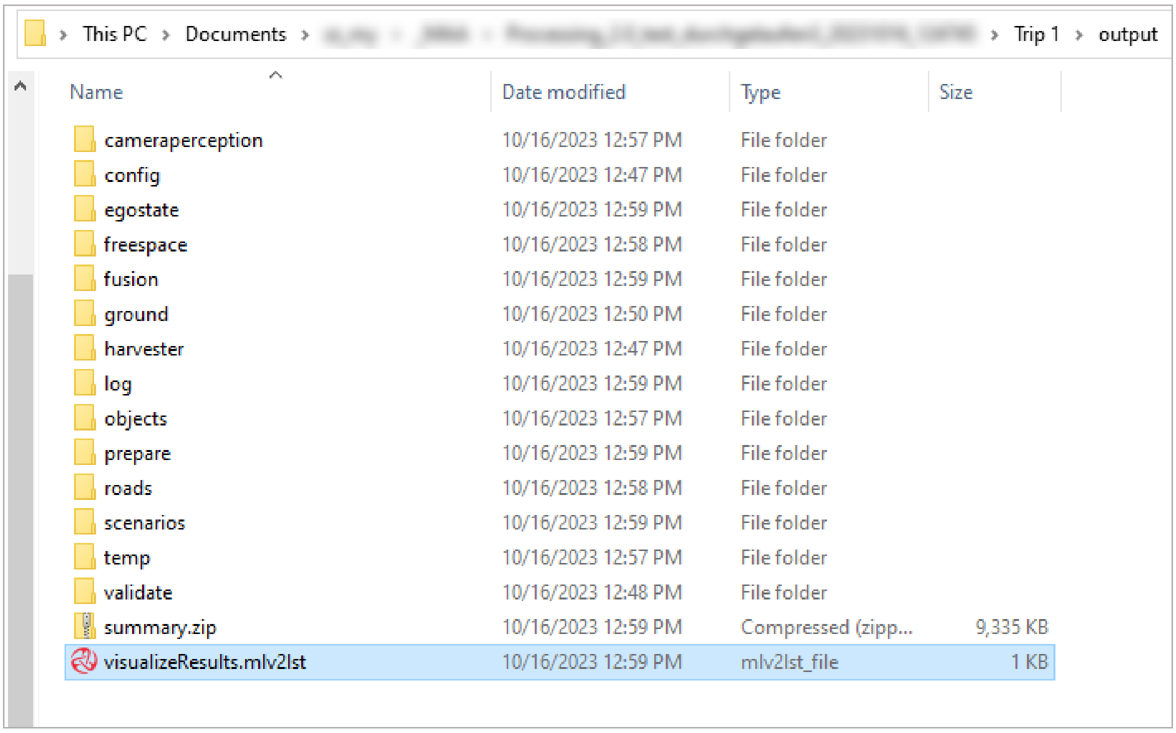
Output type	Visualization option	Procedure
Objects Roads Free space	You can open all of these output types at once with the correct configuration in MVIS Laser View 2. To do this, use the MLV2LST file that is automatically created for each trip.	See Procedure for visualizing objects, roads, and free space (page 42).
Ground	You can open the results of ground detection by using the **Ground Model** default session in MVIS Laser View 2.	See Procedure for visualizing the ground model (page 43).

Prerequisites

- MVIS Laser View 2 version 20.0.0 or higher is installed.
- The MVIS ODR plug-in version 20.0.0 or higher is installed.
- The MVIS Ground plug-in version 20.0.0 or higher is installed.

Procedure for visualizing objects, roads, and free space

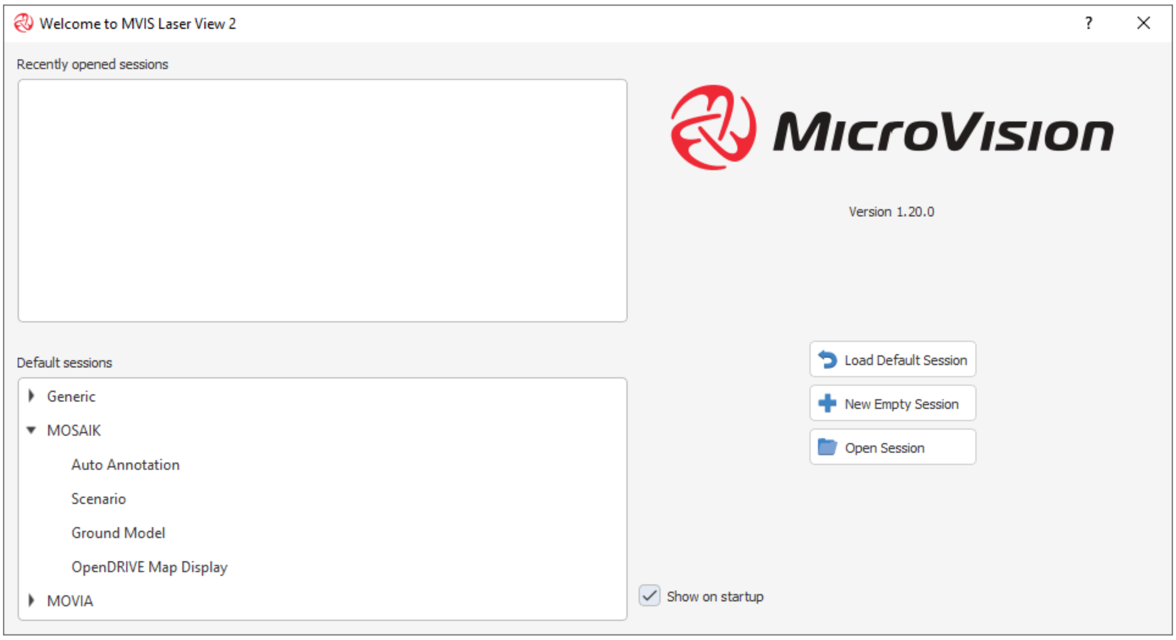
1. Navigate to the target folder that you specified for the trip processing.
2. Select the folder for one of the processed trips, for example `Trip 1`.
3. In the `output` folder, double-click the `visualizeResults.mlv2lst` file.



The trip data is displayed in MVIS Laser View 2.

Procedure for visualizing the ground model

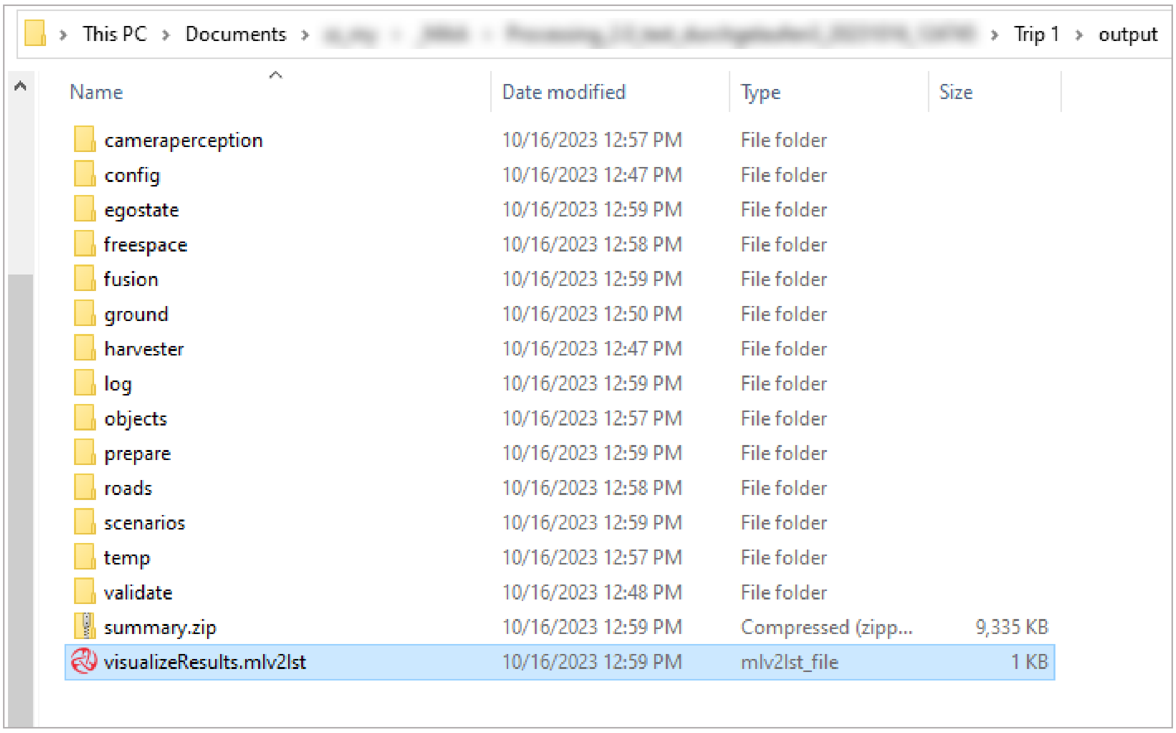
1. Start MVIS Laser View 2.
2. On the Welcome page, in the **Default sessions** section, select **MOSAIK → Ground Model**.



The Ground Model session is opened in MVIS Laser View 2.

3. Open your file browser and navigate to the target folder that you specified for the trip processing.

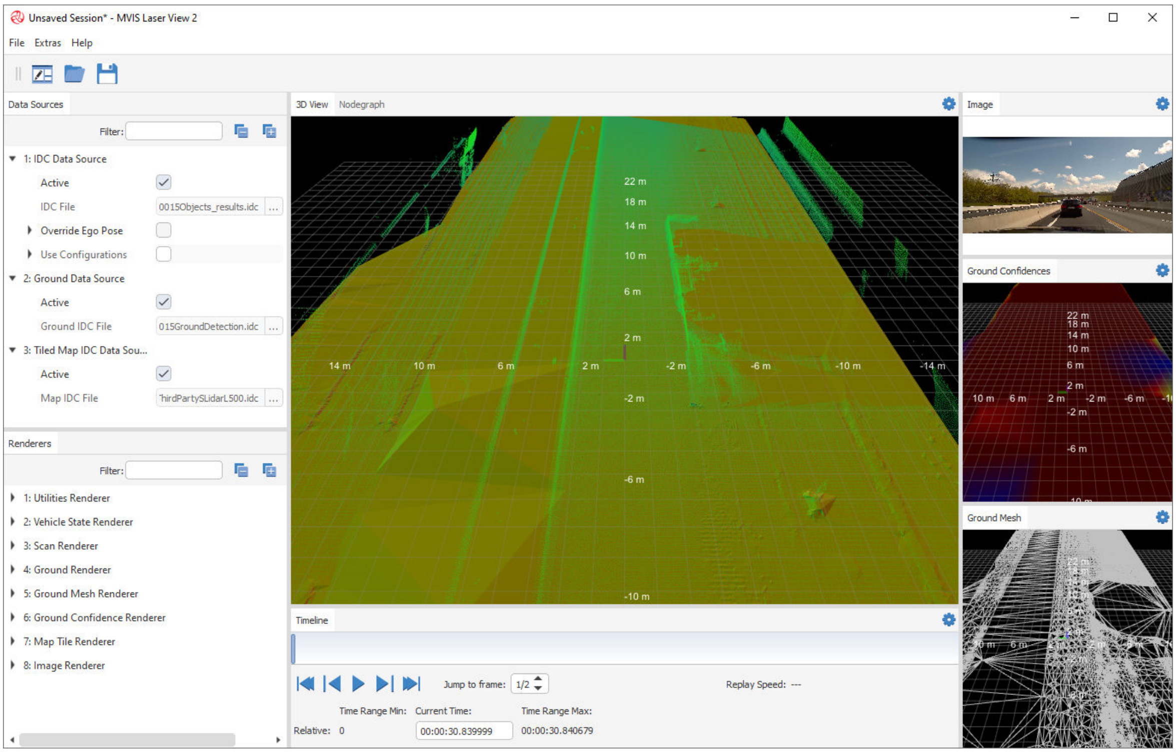
4. Select the folder of the processed trip that you want to visualize, for example `Trip 1`.



5. Drag and drop the following files from the `output` folder into the **3D View** area of MVIS Laser View 2:

   - `\output\ground\[trip]GroundDetecion.idc`
   - `\output\objects\[trip]Object_results.idc`
   - `\output\egostate\pointCloud3D\_tileMap_[sensor].idc`

   The data of the selected trip is visualized.


Unsaved Session* - MVIS Laser View 2
File
Extras
Help
Data Sources
Filter:
1: IDC Data Source
Active
IDC File
0015Objects_results.idc
Override Ego Pose
Use Configurations
2: Ground Data Source
Active
Ground IDC File
015GroundDetection.idc
3: Tiled Map IDC Data Sou...
Active
Map IDC File
hirdPartySLidarL500.idc
Renderers
Filter:
1: Utilities Renderer
2: Vehicle State Renderer
3: Scan Renderer
4: Ground Renderer
5: Ground Mesh Renderer
6: Ground Confidence Renderer
7: Map Tile Renderer
8: Image Renderer
3D View
Nodegraph
Image
Ground Confidences
Ground Mesh
Timeline
Jump to frame:
1/2
Replay Speed: ---
Time Range Min:
Current Time:
Time Range Max:
Relative:
0
00:00:30.839999
00:00:30.840679

6. The terminal version of MVIS Auto Annotation

MVIS Auto Annotation has a GUI version and a terminal version.

The terminal version of MVIS Auto Annotation is intended to be used on a system that has no graphical user interface.

6.1. View help in the terminal

1. Open the terminal.
2. Run the `run.bat` or `run.sh` file that is located in the `AutoAnnotation-GetStarted` folder, see Download and install MVIS Auto Annotation (page 18).
3. Type `-h` to show information on how to activate your user license and how to set the paths to the target folders for the trip processing. Example:
   `"C:\Users\[username]\Documents\AutoAnnotation-GetStarted\run.bat" -h`

6.2. Content of the "AutoAnnotation-GetStarted" folder

The `AutoAnnotation-GetStarted` folder in located in the downloaded installation file, see Download and install MVIS Auto Annotation (page 18).

The folder has the following content.

Content	Description
`input` folder	These folders can be used as target folders for trip processings. For more information, see Target folders in the terminal (page 46).
`output` folder	
`license` folder	
`config` folder	
`run.bat` file	Starts a trip processing on Windows.
`run.sh` file	Starts a trip processing on Linux.
`jobs.xml` file	Contains the settings for a trip processing. The file is located in the `config` folder and can be edited, see Edit the settings for a trip processing in the "jobs.xml" file (page 50).

6.3. Target folders in the terminal

For each trip processing, you need the following target folders: input, output, license, and config. Each target folder has a specific content and path parameter.

Folder	Content	Path parameter
Input	Your trip data (for example, IDC or VPCAP files)	• Terminal (Windows): `#1` • Terminal (Linux): `-i` • Script: `INPUT_PATH`
Output	Processing results and log files for one trip processing To avoid that previously generated outputs will be overwritten, the folder must be empty when a trip processing is started.	• Terminal (Windows): `#2` • Terminal (Linux): `-o` • Script: `OUTPUT_PATH`
License	Licensing files of your user license: • `license.mlf` • `settings.json` For information on how to generate and export the licensing files, refer to the *MVIS License Manager User Manual*, see Related documents (page 7).	• Terminal (Windows): `#3` • Terminal (Linux): `-l` • Script: `LIC_PATH`
Config	`jobs.xml` file and supporting configuration files, for example, for the conversion of VPCAP to IDC files. The `jobs.xml` file is located in the `AutoAnnotation-GetStarted` → `config` folder.	• Terminal (Windows): `#4` • Terminal (Linux): `-c` • Script: `CFG_PATH`

6.4. List of trip processing settings in the "jobs.xml" file

Setting	Corresponding GUI settings	Default	Description
`Ground` job	**Ground detection**	Deactivated	See List of processing options in the "Processing" tab (page 30).
`Objects` job	**Objects**	Activated	
`Roads` job	**Roads**	Activated	
`Freespace` job	**Free space detection** **Objects**	Deactivated	
`Scenarios` job	**Scenario detection** **Objects** **Roads**	Activated	

Setting	Corresponding GUI settings	Default	Description
`PrepareIdcIgnoreValidation` job	–	Deactivated	By default, all input files are validated. Regardless of the result of the validation, all input files will be processed.
`IdcConclude` job	**EVS output format**	Deactivated	Select this option to convert the new 3D ego state (`0x2809 VehicleState` data type) to 2D ego state (`0x2808 VehicleState` data type) like it was produced by ibeo Evaluation Suite. For more information, refer to the *MVIS Data Interface Specification*, see Related documents (page 7).
`Help` job	–	Deactivated	Prints a brief description of all known jobs.
`License` job	–	Deactivated	Prints all used license statements.
`inputFiles` parameter	**Input files**	Empty	All files in the `input` folder will be processed.
`maxFileSystemUsage` parameter	–	99	Specifies how much space can be in use on the file system that contains the `output` folder. If more space is in use, the trip processing will fail.
`environment` parameter	**Environment**	highway	To improve the quality of the output select the environment that matches your recording.

6.5. Set up the target folders for a trip processing in the terminal

Context

Before you start a trip processing in the terminal, set up the target folders.

For more information on target folders, see Target folders in the terminal (page 46).

Procedure

1. Navigate to the `AutoAnnotation-GetStarted` → `config` folder, see Download and install MVIS Auto Annotation (page 18).
2. Open the `jobs.xml` file.
3. Set up the target folders. You can use the folders in the `AutoAnnotation-GetStarted` folder or you can use your own folders.
   The following applies:

- The `output` folder must be empty.
- The `input` folder must contain your input files.
- The `config` folder must contain the `jobs.xml` file and if applicable the supporting configuration files.
- The `license` folder must contain your licensing files.

6.6. Preset the paths to target folders in the script file

Context

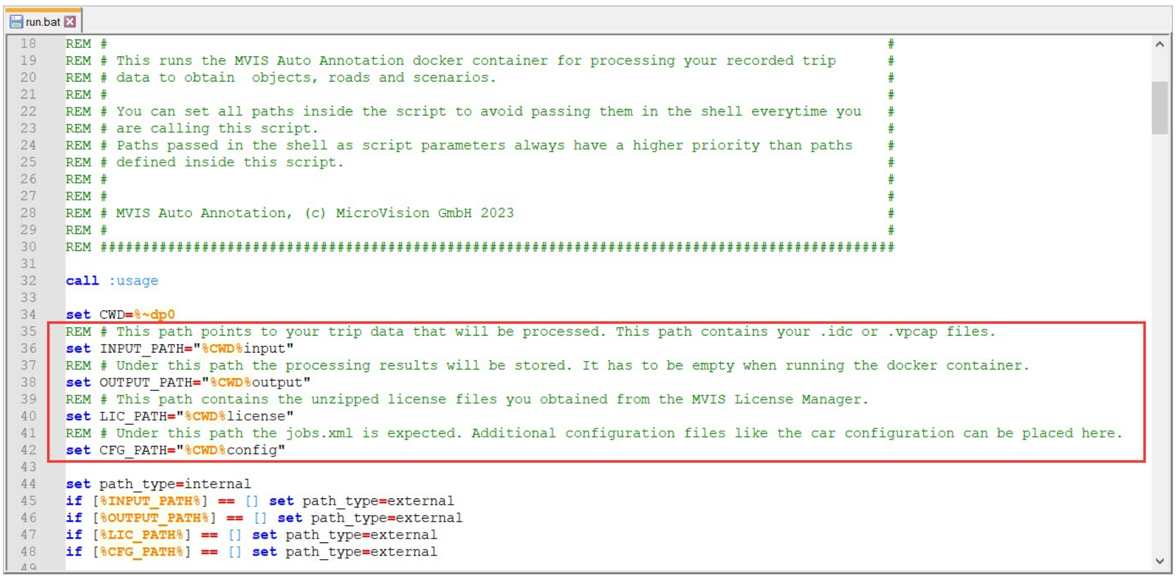
When you start a trip processing, you must set the paths to the following target folders: `input`, `output`, `license`, and `config`.

You can either set these paths each time you start a trip processing in the terminal, or you can preset them in the script file and only type `run` in the terminal. On Linux, you can combine both methods. On Windows, you can only use one method.

Paths that you preset in the script file always have a lower priority than paths that you set in the terminal. That means, you can override the preset paths in the terminal.

Procedure

1. Open the `run.bat` or `run.sh` file that is located in the `AutoAnnotation-GetStarted` folder, see Download and install MVIS Auto Annotation (page 18)..
2. Set the path for one or more target folders with the following parameters:
   - `INPUT_PATH`
   - `OUTPUT_PATH`
   - `LIC_PATH`
   - `CFG_PATH`



```
run.bat
REM #                                                                              #
REM # This runs the MVIS Auto Annotation docker container for processing your recorded trip      #
REM # data to obtain  objects, roads and scenarios.                                #
REM #                                                                              #
REM # You can set all paths inside the script to avoid passing them in the shell everytime you   #
REM # are calling this script.                                                     #
REM # Paths passed in the shell as script parameters always have a higher priority than paths    #
REM # defined inside this script.                                                  #
REM #                                                                              #
REM #                                                                              #
REM # MVIS Auto Annotation, (c) MicroVision GmbH 2023                              #
REM #                                                                              #
REM ################################################################################################

call :usage

set CWD=%~dp0
REM # This path points to your trip data that will be processed. This path contains your .idc or .vpcap files.
set INPUT_PATH="%CWD%input"
REM # Under this path the processing results will be stored. It has to be empty when running the docker container.
set OUTPUT_PATH="%CWD%output"
REM # This path contains the unzipped license files you obtained from the MVIS License Manager.
set LIC_PATH="%CWD%license"
REM # Under this path the jobs.xml is expected. Additional configuration files like the car configuration can be placed here.
set CFG_PATH="%CWD%config"

set path_type=internal
if [%INPUT_PATH%] == [] set path_type=external
if [%OUTPUT_PATH%] == [] set path_type=external
if [%LIC_PATH%] == [] set path_type=external
if [%CFG_PATH%] == [] set path_type=external
```

3. Save the script file.

6.7. Edit the settings for a trip processing in the "jobs.xml" file

Context

The settings for a trip processing are stored in the `jobs.xml` file.

When you use the GUI version, the `jobs.xml` file will automatically be generated when you start a trip processing.

When you use the terminal version, you can use the `jobs.xml` file that is located in the `AutoAnnotation-GetStarted` folder, see Download and install MVIS Auto Annotation (page 18). You can keep the default settings or edit the `jobs.xml` file.

You can also specify the settings in the GUI, generate the `jobs.xml` file and use it in the terminal version.

Procedure in the "jobs.xml"

1. Navigate to the `jobs.xml` file.
   By default, it is located in the `AutoAnnotation-GetStarted` → `config` folder.
2. Edit the settings for the trip processing.
   For information on trip processing settings in the `jobs.xml` file, see List of trip processing settings in the "jobs.xml" file (page 47).
3. Save the `jobs.xml` file.

Procedure using the GUI

1. Specify the settings for a trip processing in the GUI, see Specify the settings for a trip processing (page 30).
2. On the **Processing** tab, in the **Select processing options** section, select the processing options you want to use in the terminal.
3. Click **Start processing**.
   The `jobs.xml` file is automatically generated and located here:
   `[selected target folder]` → `[trip folder]` → `config`

6.8. Start a trip processing in the terminal

Context

You start a trip processing in the terminal with the `run.bat` or `run.sh` script file. These files are located in the `AutoAnnotation-GetStarted` folder, see Download and install MVIS Auto Annotation (page 18).

Prerequisites

You have enough credits left (MyMVIS → **Licenses** tab → **User licenses** → **Credits** column).

Procedure on Windows

1. Set up the target folders for the trip processing, see Set up the target folders for a trip processing in the terminal (page 48).
2. Do one of the following:
   - Preset the paths to all target folders in the script file, see Preset the paths to target folders in the script file (page 49).
     Open the terminal and run the `run.bat` file. Example:

     ```
     "C:\Users\[username]\Documents\AutoAnnotation-GetStarted\run.bat"
     ```

   - Open the terminal, run the `run.bat` file, and set the paths to all target folders. Example:

     ```
     "C:\Users\[username]\Documents\AutoAnnotation-GetStarted\run.bat"
     "D:\myUsbDrive\trip1input"
     "D:\myUsbDrive\trip1output"
     "D:\myUsbStick\license"
     "D:\myUsbDrive\vehicle1\config"
     ```

     The following applies:

     – To use network shares, start the network volume wizard. Example:

     ```
     "C:\Users\[username]\Documents\AutoAnnotation-GetStarted\run.bat" -n
     ```

     Alternatively, you can enter the IP addresses manually. For more information, see View help in the terminal (page 46).

     – To use USB sticks, ensure that in your Docker Desktop settings, the **Use the WSL 2 based engine** checkbox is deselected.

   The trip processing starts.
3. For log information, see Logging (page 39).

Procedure on Linux

1. Set up the target folders for the trip processing, see Set up the target folders for a trip processing in the terminal (page 48).
2. If applicable, preset one or more paths to the target folders in the script file, see Preset the paths to target folders in the script file (page 49).
3. Open the terminal.
4. Run the `run.sh` file and se the paths to the target folders that are not preset in the script file.
   - Example 1: no target folder path is preset in the script file:

     ```
     /home/[username]/Documents/run.sh -i
     "/media/[username]/myUsbDrive/trip1input" -o
     "/media/[username]/myBigNetworkStoreage/trip1output" -l
     "/media/[username]/myUsbStick/license" -c
     "/media/[username]/myUsbDrive/vehicle1/config"
     ```

- Example 2: only the output folder path is not preset:

  ```
  /home/[username]/Documents/run.sh -o
  "/media/[username]/myBigNetworkStoreage/trip1output"
  ```

  The trip processing starts.

5. For log information, see Logging (page 39).

6.9. Stop a trip processing in the terminal

Context

You can stop a trip processing either by pressing **Ctrl** **C** or by stopping a container.

Procedure for stopping a container

1. During a trip processing, enter `docker container ls`.
   A list of all running containers is displayed.
2. Copy the ID of the container that you want to stop.
3. Enter `docker stop [copied ID]`.

7. Switching from ibeo Evaluation Suite to MVIS Auto Annotation

If you have been working with ibeo Evaluation Suite so far, the following comparison will support you when switching to MVIS Auto Annotation.

> **NOTE**
> If you have any questions regarding the processing parameters or outputs of MVIS Auto Annotation, contact the MicroVision Support Team.

Comparison of how to process a trip

Phase	Steps in ibeo Evaluation Suite	Steps in MVIS Auto Annotation
Preparation	1. Start MongoDB. 2. Start ibeo Evaluation Suite (client-server version or standalone version).	1. Start container host (for example, Docker daemon or Docker Desktop). 2. Optional: start the GUI version of MVIS Auto Annotation. **Note**: Processing is possible without the GUI version, for example, to allow more options, for automation, or to run the processing in a cloud. In this case, the MVIS Auto Annotation Docker can be started directly via a script and the rest is done automatically.
Trip processing	1. Import trip. 2. Wait for import to be completed. 3. Create session with trip. 4. Start the processing. 5. Wait for processing to be completed. 6. Export results. 7. Wait for export to finish.	1. Select trip data and start the processing. 2. Wait for processing results.

Comparison of processing parameters

ibeo Evaluation Suite	MVIS Auto Annotation
GPS Device selected manually	Detected automatically
IMU Device selected manually	Detected automatically
Lanes options	New detection model used, no parameter setting required
Blind spot monitoring	Embedded in scenario detection

Comparison of outputs

Output type	ibeo Evaluation Suite	MVIS Auto Annotation
Dynamic objects	`output` folder → IDC file	ibeo Evaluation Suite output format: `conclude` folder → IDC file New data output format: `objects` folder → IDC file
Lanes	`output` folder → IDC file (as `CarriageWay`)	`roads` folder → XODR file (as OpenDRIVE)
Ground detection	—	`ground` folder → IDC file
Free space	—	`freespace` folder → CSV file
Scenarios	—	`scenarios` folder → CSV files

8. Ego state estimation

The ego vehicle is the vehicle that is equipped with the reference sensors and that records the measurement data.

Ego state estimation estimates the vehicle's position, orientation, and motion based on the recorded GPS, IMU, odometry. and lidar data. Subsequently, the ego state results are used by other components like road detection or object tracking.

8.1. Limitations of ego state estimation

The following limitations apply for ego state estimation:

- In general, the performance of ego state estimation is heavily affected by the quality of the input signals like GPS, IMU, and odometry.
- Currently, loop closure is not supported, that is, driving the same road twice or more in a single trip.
- The ego vehicle accelerations are not estimated or filtered. Therefore, the acceleration values can be subject to noise.
- The ego vehicle turn rates are not estimated or filtered. Therefore, the turn rates can be subject to noise.
- The availability of ego state features depend on the sensor setup, see Prerequisites for ego state estimation (page 56).

8.2. Features for ego state estimation

Ego state estimation samples timestamps throughout the trip according to the measurement data. The ego vehicle state is estimated for each of these timestamps.

An ego vehicle state consists of:

Component	Description
WRC reference point	The reference point of the fixed world reference coordinate system. This reference point is constant throughout the whole trip. For more information on WRC, see World reference coordinate (WRC) system (page 10).
VRC	The vehicle road coordinate system position and orientation relative to the WRC system. For more information on VRC, see Vehicle road coordinate (VRC) system (page 11).
VBC	The vehicle body coordinate system relative to the VRC system. For more information on VBC, see Vehicle body coordinate (VBC) system (page 11).
3D vehicle motion	The 3D vehicle motion consists of 3D velocity and 3D turn rates.

8.3. Prerequisites for ego state estimation

Ego state estimation supports several sensor setups. The sensor setup determines which ego state estimation features are available.

Sensor setup	Core ego state features (WRC/VRC/motion)	IMU pitch offset correction	VBC-VRC estimation
• INS (for example, OxTS RT, NovAtel PwrPak7) • Vehicle odometry	✓	✗	✗
• Reference grade INS • Vehicle odometry • 360° lidar, mounted horizontally on roof	✓	✓	✗
• Reference grade INS • Vehicle odometry • VBC/VRC setup: – MIVS LUX on Roof, facing ground in front of ego vehicle – MVIS LUX or third-party S Lidar on roof, facing ground behind ego vehicle	✓	✓	✓

8.4. Output description for ego state estimation

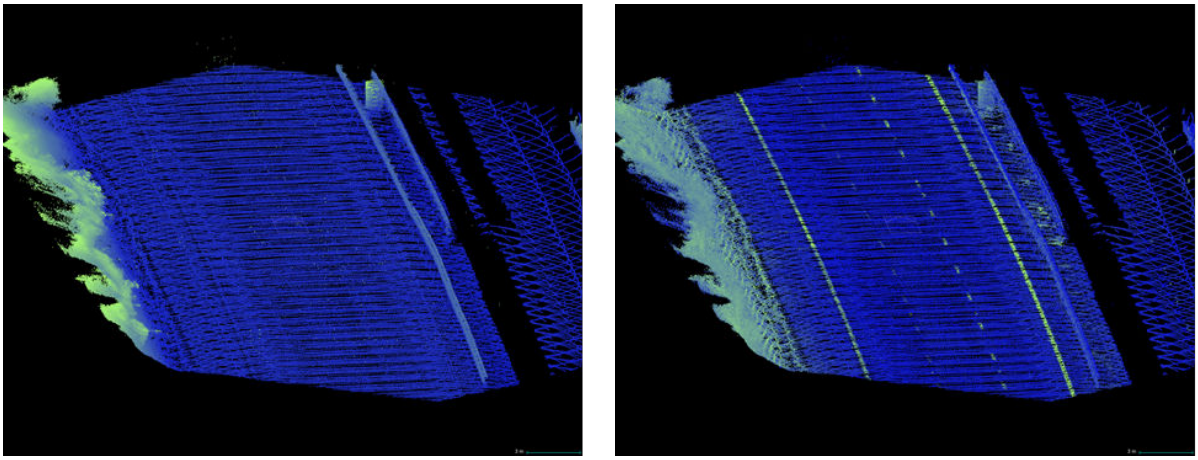
The output of ego state estimation is located in the output path in the `egostate` folder.

The results of ego state estimation are saved for each of the sampled timestamps in the MVIS Auto Annotation output IDC file.

The results are saved as `VehicleState` datatype (SDK datatype ID `DataType_VehicleStateBasic2809`).

9. Point cloud mapping

Point cloud mapping generates a static map of the environment in point cloud representation. The following example point cloud map shows a highway that is color-coded by height (left) and by intensity (right).



The lidar scans that are created throughout the trip are transformed to WRC coordinates by using the ego state results. The transformed lidar scans are then accumulated into a map. For each scanner type, a separate map is created. Example: If a vehicle is equipped with 4 MOVIA sensors and 2 third-party lidar sensors, 2 maps will be created: one map that contains the accumulated recordings of the MOVIA sensors and one that contains the accumulated recordings of the third-party lidar sensors.

The output of point cloud mapping is saved in form of map files in IDC format. For each scanner type, a separate map file is saved. A map file consists of tiled point cloud map data as described in Output description for point cloud mapping (page 58). The map file name reflects the scanner type, for example, `myMap_tileMap_ThirdPartySLidarL500.idc`.

9.1. Limitations for point cloud mapping

The following limitations apply for point cloud mapping:

- In general, the point cloud quality depends on a precise ego state estimation and thus is affected by the quality of the ego state input signals like GPS, IMU, and odometry.
- Currently, loop closure is not supported, that is, driving the same road twice or more in a single trip.
- No dynamic classification of scan points is applied, so all scan points are considered in the mapping process. That means, scan points that originate from dynamic objects (for example, other cars or pedestrians) are not removed from the map.

9.2. Features for point cloud mapping

Point cloud mapping supports the following features:

- Maps are tiled and point coordinates are stored relative to the tile corner, see Output description for point cloud mapping (page 58).
- Thinning is applied to reduce point density to a reasonable granularity.
- Typically, two maps are generated for each scanner type: a 3D map and a 2.5D map. For the 2.5D map, the ego-state results are projected onto the XY-plane during mapping such that ground points are close to z = 0. This is required by road detection that outputs 2D OpenDRIVE maps which need to be aligned with point cloud maps.
- Points beyond a virtual view distance are neglected. First, these points are outside the region of interest, and second, this minimizes the effects of orientation errors of the ego state. The default virtual view distance is 20 m from the ego vehicle. This value can be configured.
- In case parts of the ego vehicle are in the field of view of the lidars, points measured on the ego vehicle are rejected.
  The ego vehicle dimensions can be configured.
  - It is recommended to configure the ego vehicle dimensions in the `raw2idc` converter configuration.
  - Alternatively, configuration is also possible in the `egostate` configuration file. Note that ego vehicle dimensions within the `egostate` configuration file are dismissed if the input data (IDC files) contain ego vehicle dimensions as well.

9.3. Prerequisites for point cloud mapping

This feature depends on:

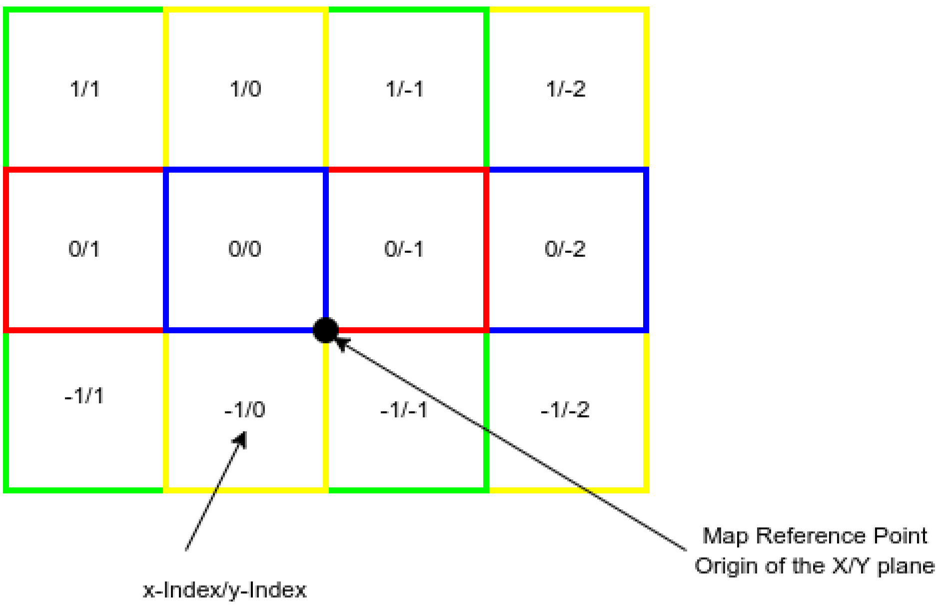
- Ego state estimation
- Precise calibration of all lidar sensors

9.4. Output description for point cloud mapping

The output of point cloud mapping is located in the output path in the `egostate\pointCloud` and `egostate\pointCloud3D` folders.

The point cloud mapping output consists of map files in IDC format. Within a map file, all points are assigned to their respective map tile.

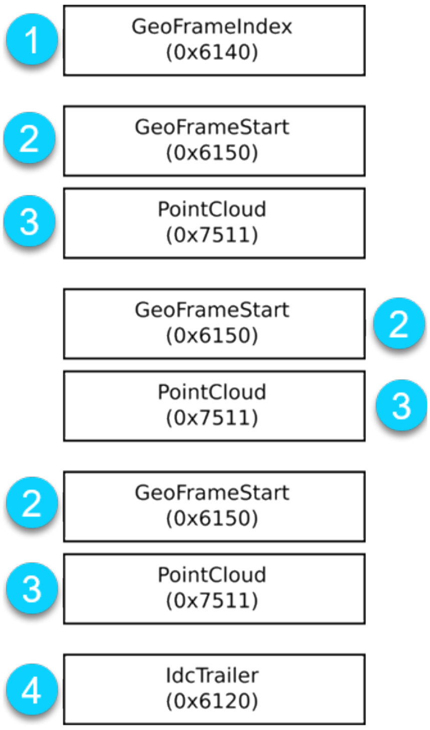
Structure of a point cloud map
A point cloud map consists of a map reference point and map tiles.



Element	Description
Map reference point	The map reference point is identical to the ego state WRC origin. All map content is given in WRC coordinates, so that it aligns with the results of the other components.
Map tiles	Map tiles contain the actual map content, that is a segment of the point cloud consisting of map points. The point cloud of a map tile is stored as `DataType_PointCloud7511` datatype. Map tiles are 2D squares. All map tiles of a point cloud map have the same size. The size is constant. Map tiles are indexed by an $(x \quad y)$index. The x- and y-directions are aligned with the WRC x-axis and y-axis. All point coordinates are relative to the corresponding tile corner. In order to retrieve the WRC coordinates of a point, the tile offset (tile size * tile index) has to be added. Note that 32-bit floating point precision is insufficient for WRC coordinates of large maps.

Structure of a map file

The tiles and point cloud data within a single map file are stored sequentially as depicted in the image below.

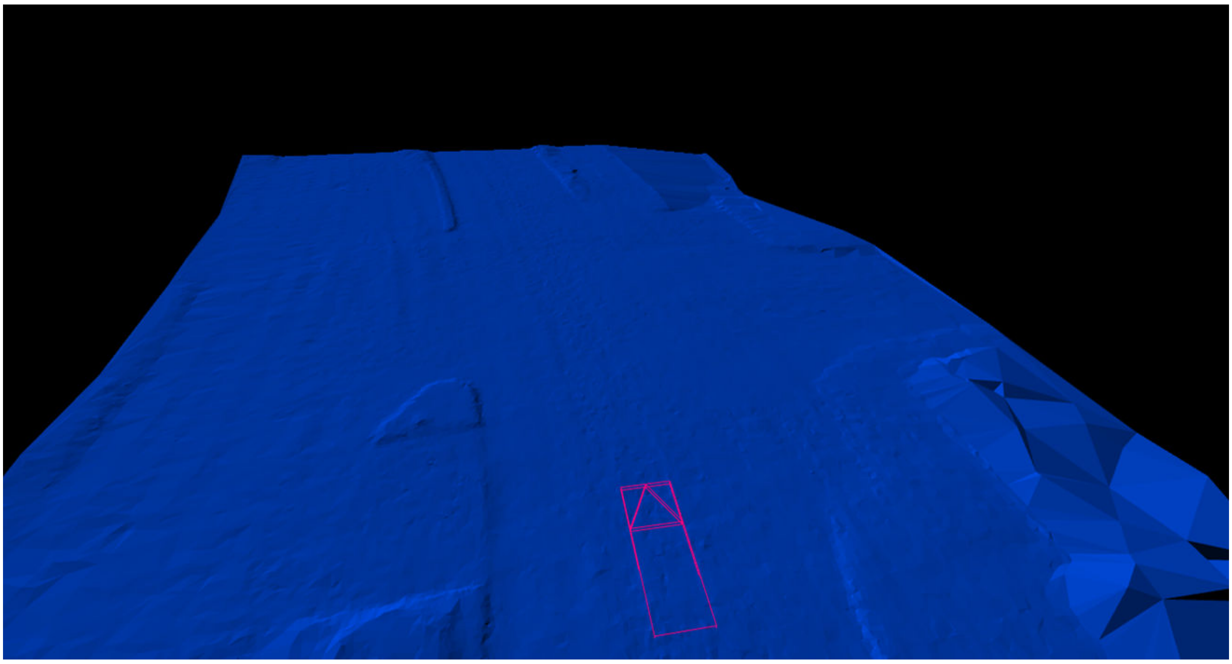


No.	Element	Description
1	`GeoFrameIndex`	This datatype is an index of all map tiles that are available within the point cloud map. This datatype provides the exact position (byte offset) of each map tile within the map file.
2	`GeoFrameStart`	This datatype marks the start of a single map tile. All data that lies after this datatype and before the next `GeoFrameStart` belongs to the this tile.
3	`PointCloud`	This is the actual point cloud of the tile.
4	`IdcTrailer`	This datatype marks the end of the file.

10. Ground detection

Ground detection offers automatically extracted and highly accurate information of the ground surface within the recorded trip data.

The resulting static georeferenced map is expressed in IDC format. You can visualize the output with the MVIS Laser View 2 Ground plug-in and access it through the provided C++ library, see Visualize the output of a trip with MVIS Laser View 2 (page 42).



10.1. Limitations for ground detection

- Ground detection is currently developed for a dual third-party S Lidar L500 setup. Other third-party sensors are supported but the quality of the output may decrease.
- Ground detection is developed for dry weather conditions. Deviation from these conditions may decrease the quality of the output.
- Loop closures are not handled. This means that visiting the same location twice during a trip is likely going to result in multiple ground meshes on top of each other.
- Occlusion: in case of complete or partial sensor occlusion, there will be no output or output of decreased quality in affected areas. This includes occlusion by cars, trucks, other objects, and vegetation.
- Onset time: at the beginning of the recording, at the end of the recording, before occlusions and after occlusions, the output quality may be reduced.

NOTE
The limitations are subject to change. New features are developed constantly.

10.2. Prerequisites for ground detection

Ground detection is based on high-level data and relies on the following previous processing step:

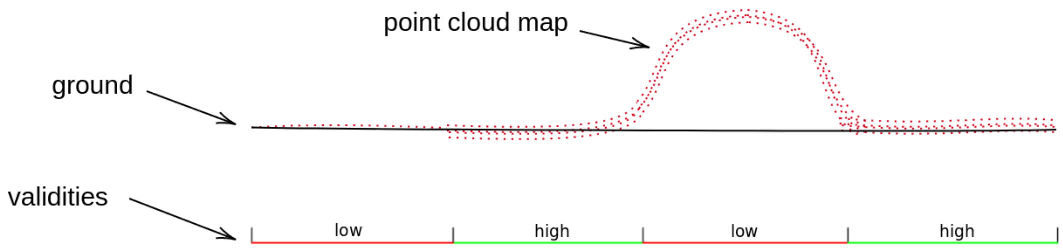
Ego state estimation including point cloud mapping

10.3. Features of ground detection

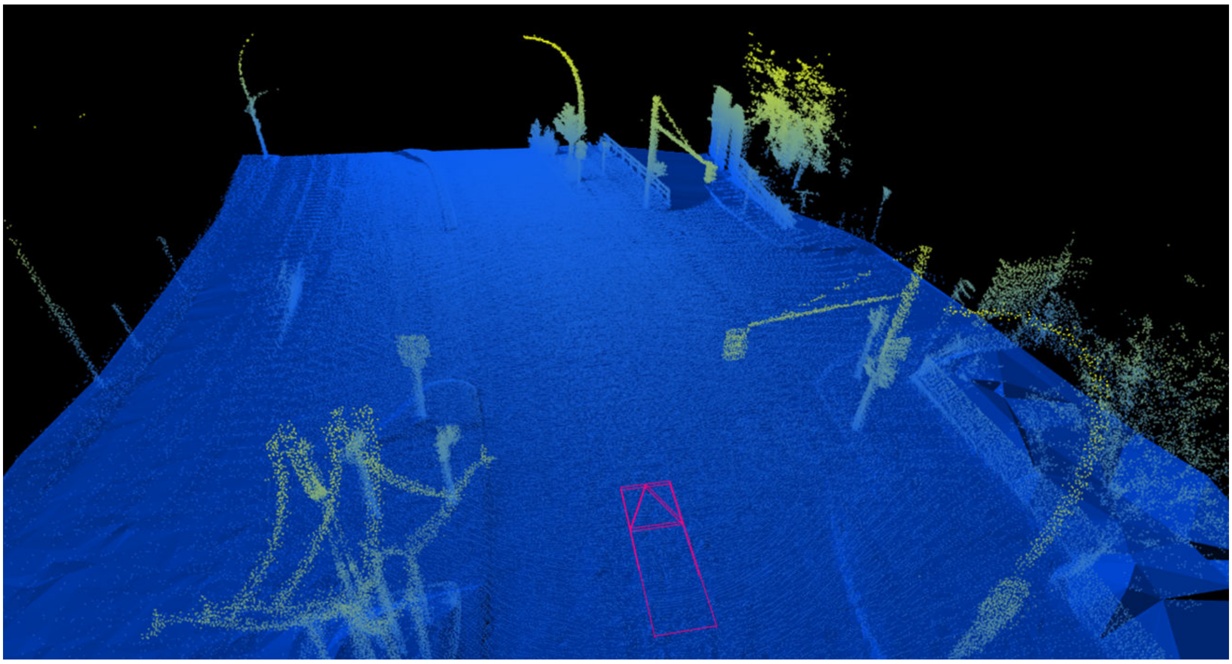
Ground detection automatically extracts the surface geometry in proximity to the recording vehicle. The surface is represented by a triangle mesh which is serialized as a single georeferenced ground model.

Validities

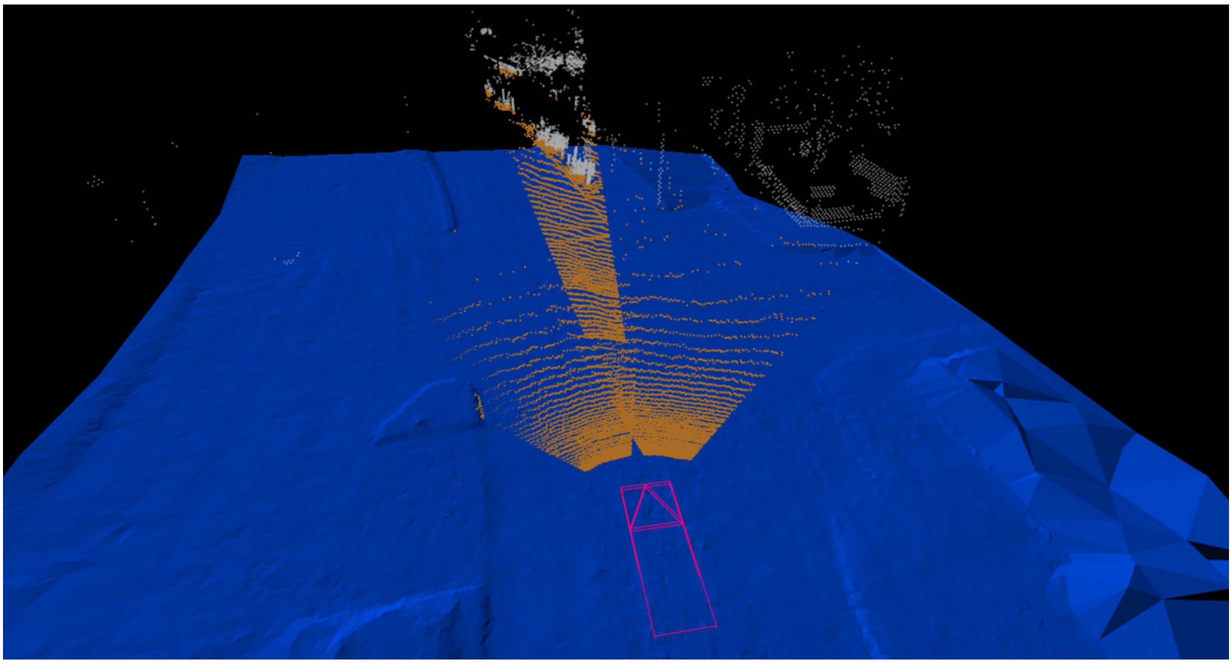
The ground model also features validities $v \in [0,1]$. The validity is based on the number of scan points that are in close proximity to the calculated ground model at the time of calculation.



Overlay of ground model and point cloud map (colorization according to point height)



Example use case: binary classification of scans into ground (brown) and non-ground (gray)



10.4. Output description for ground detection

The output of ground detection is located in the output path in the `ground` folder.

The ground detection output is an IDC file. The pre-compiled library `ground-lib` is provided and serves as the interface to the ground model.

11. Object tracking

With object tracking, you can automatically detect dynamic objects, such as cars, bikes, and pedestrians, as well as potentially dynamic objects such as parking cars.

You can visualize and manually edit the output with MVIS Laser View 2 and the Smart Editing plug-in.

The quality of object tracking depends on the sensor setup that is used to record trips.

11.1. Limitations for object tracking

The following limitations apply for object tracking:

- Object tracking is developed for a highway environment. Object tracking can also be applied to other scenarios, for example, a city, but the quality is not guaranteed.
- Object tracking is developed for tracking dynamic objects, that means traffic participants.
- Non-moving objects, such as traffic signs, trees, and buildings, will not be detected.
- All object properties, such as dimensions and velocity, are measurement-based. That means, object tracking does not derive the object size from other properties such as the object class; no property is the best guess.
  For example, if a car is 250 m away in front of the ego vehicle for the whole trip, the object size will be rather small in length, because the point cloud from that car only comes from the rear part.
- Objects that are not visible for a certain period of time, for example, due to occlusion, may be detected with a different ID.

11.2. Prerequisites for object tracking

- Ego state estimation
- Point cloud from scans

11.3. Features for object tracking

Detected objects are gathered in object lists. An object list contains information on the objects that were detected in a scan.

A scan is always followed by an object list. If no objects are detected in a scan, the object list is empty.

Attributes of an object

Attribute	Description	Format/Unit
`objectId`	A unique positive integer assigned to the object The object ID is also unique within a trip, that means, within an IDC file. Note that the object IDs within a trip do not have to be consecutive. There can be objects, for example, with ID 1, 2, and 5 within a trip while ID 3 and 4 are missing.	unsigned integer
`objectTimestamp`	Timestamp of the object The timestamp is estimated based on timestamps of scan points associated to the object. Scan points in the same scan are usually measured at different timestamp, for example, scan points near scan start angle have earlier timestamps compared to those near the end of this scan. Therefore, objects detected in the same scan can have different timestamps.	NTP format
`objectAge`	Lifetime of the object The lifetime starts from the timestamp when the object is first detected and ends at the timestamp when the object is last tracked.	millisecond
`classification`	Class of the object For more information, see Notes on the "classification" attribute (page 66).	unsigned integer
`classificationQuality`	Classification confidence The value is between 0 and 1, with 0 being not confident at all and 1 being of very high confidence.	float between 0 and 1
`objectBoxSize`	Size of the object bounding box in x, y, and z direction	float array of length 3 ([x, y, z])
`objectBoxSizeSigma`	Standard deviation of the object bounding box size in x, y, and z direction	float array of length 3 ([x, y, z])
`centerOfGravity`	Position of the object's center in x, y, and z direction, relative to the origin The object center is defined as the respective center of the object bounding box in each dimension.	float array of length 3 ([x, y, z])
`courseAngle`	Direction of current motion	radians
`courseAngleSigma`	Standard deviation of object course angle	radians

Attribute	Description	Format/Unit
`relativeVelocity`	2D velocity relative to the ego vehicle in x and y direction	meter/second
`relativeVelocitySigma`	Standard deviation of the object's relative velocity	meter/second
`absoluteVelocity`	Absolute velocity of the object	meter/second
`absoluteVelocitySigma`	Standard deviation of the object's absolute velocity	meter/second
`longitudinalAcceleration`	Longitudinal acceleration of the object in direction of the velocity vector	meter/second 2
`longitudinalAccelerationSigma`	Standard deviation of longitudinal acceleration	meter/second 2
`yawRate`	Yaw rate of the object	radian/second
`yawRateSigma`	Standard deviation of yaw rate	radian/second
`timeToCollision`	Predicted time until collision between the ego vehicle and the object It is assumed that the motion of the ego vehicle and the object do not change, that means constant velocity is assumed. A value of -1 means that no collision has been estimated.	second

Notes on the "classification" attribute

The following table shows the numbers and classes of the `classification` attribute and how they are associated.

Number	Class	Notes
0	unclassified	—
1	unknown small	—
2	unknown big	—
3	pedestrian	—
4	bike	Only used when the algorithm cannot distinguish between a motorbike and a bicycle.
5	car	—
6	truck	—
15	motorbike	Only used when the algorithm can distinguish between a motorbike and a bicycle.
16	infrastructure	—
17	bicycle	Only used when the algorithm can distinguish between a motorbike and a bicycle.

Number	Class	Notes
18	small obstacle	—

11.4. Output description for object tracking

The output of object tracking is located in the output path in the `objects` folder.

The object tracking output is an IDC file. Detected objects can also be exported in JSON format.

12. Road detection

With road detection, you can automatically detect lane markings, road boundaries, traffic signs, and traffic lights in different environments.

The resulting static georeferenced map has the open file format OpenDRIVE. For more information, see https://www.asam.net/standards/detail/opendrive

You can visualize and manually edit the output with MVIS Laser View 2 and the OpenDRIVE plug-in.

Detection of semantic roads and lanes is based on a combination of lidar intensity features as measured on lane markings, geometric features like road boundaries and curbstones, and trajectory information of traffic participants.

12.1. Limitations for road detection

The following limitations apply for road detection:

- Lane marking detection requires highly reflective lane markings in comparison to their surroundings. Worn out or faded lane markings may decrease the quality of the output.
- Road detection is developed only for dry weather conditions. Any other weather condition may decrease the quality of the output.
- The quality of the road detection depends on the sensor setup that is used to record trips.
- Environment processing option:
  - Road detection is developed for highway environments. Application to off-highway environments (for example, rural or city) may decrease the quality of the output.
  - Currently, road detection relies on separate configurations for highway and off-highway environments. Since there is no automatic detection in place, ensure that trips are cut in advance so they only feature one environment exclusively. Then process these trips with the matching **Environment** processing option.
- Loop closures are not handled. This means that visiting the same location twice during a trip is likely going to result in multiple lanes on top of each other.
- Not yet supported and marked as such are:
  - Construction sites
  - Highway entries and exits
  - Junctions
- Lane marking color is not classified.

- Lane marking width detection quality decreases with increasing lateral distance of lane markings to the ego vehicle. Lane marking width is underestimated as a consequence of reduced density of lidar measurement points in increased lateral distance.
- Supplemental road marking features like arrows, text, crosswalk markings are not supported. Botts dots are not supported.
- Guardrails are reported as generic guardrails. No classification in concrete, metal, double metal, etc.
- Traffic signs and traffic lights can only be detected if camera perception is used.
- For traffic signs and traffic lights, only the height above ground and the height of the actual signs and lights are detected. The width is not detected.
- Occlusion: A sensor can be partially or completely occluded by cars, trucks, other objects, and vegetation. If object tracking does not detect the objects that occlude the sensor, there will be no output or output of decreased quality in affected areas.
- Onset time: At the beginning, end of the recording and before, after occlusions, the output quality may be reduced.

12.2. Prerequisites for road detection

Road detection relies on the following previous processing steps:

- Ego state estimation including point cloud mapping
- Object tracking
- Optional: camera perception

12.3. Features for road detection

Road detection automatically extracts information on lane markings, road boundaries, traffic signs, and traffic lights from the recorded trip data.

12.3.1. Extraction of semantic lanes

The extraction of semantic lanes is the central feature of road detection. This high-level association of more basic features as lane marking geometries enables the output of a global map that accurately reflects the road topology.

Lanes that are found to be drivable are marked as such. Additionally, stop lanes are detected.

Based on internal probabilistic quantities, an `existenceConfidence` measure is derived and appended to each semantic lane.

12.3.2. Extraction of lane geometries

In general, lane geometries are provided by means of a polynomial representation with respect to a local reference coordinate system. However, there are two different common approaches to accommodate lane geometries:

- `LaneBorder`: based on the actual geometry of each lane border. This approach is the natural choice when extracting geometries from measurement data.
- `LaneWidth`: based on the lane width between two neighboring lane borders, respectively. This approach is very common in a simulation context.

Road detection supports both of these definitions. By default, separate output files for `LaneBorder` and `LaneWidth` variants are provided.

In principle, both definitions can be used almost interchangeably. Note, though, that the supplemental "accuracy estimation" output is available for `LaneBorder` output only.

Refer to the OpenDRIVE standard documentation for more information on the definition of the different geometries and coordinate systems.

12.3.3. Classification of lane marking types

The most common types of lane markings are detected automatically. Detected types include solid and dashed markings as well as pairs of combinations thereof.

In case the type could not be determined, a low `typeConfidence` value is output.

12.3.4. Estimation of lane marking widths

The width of detected lane markings is measured at multiple positions in the input lidar point cloud and provided as a single width estimation output value per geometry. Due to the inherent sparsity of lidar scanning, the width estimation typically is most accurate in close proximity to the ego vehicle.

12.3.5. Detection of road boundaries

Another fundamental feature of road detection is the extraction of road boundaries like metal guardrails and concrete barriers. Road boundaries are detected on the left-hand side and the right-hand side of the ego vehicle.

Note that there is no constraint to the number of road boundaries when counting laterally, even if only the closest road boundaries could actually be reached by the ego vehicle.

Static road boundaries are detected and output irrespective of their making. So different types of man-made structures are detected as well as natural boundaries like walls or vegetation.

12.3.6. Detection of traffic signs and traffic lights

If the optional camera perception processing is used, road detection is able to automatically provide traffic signs and traffic lights as part of its OpenDRIVE output.

The fusion of detections found in the camera image and the high resolution 3D information of lidar sensors ensures both a highly accurate position estimation and an extensive list of supported classes.

12.4. Output description for road detection

The output of road detection is located in the output path in the `roads` folder.

The road detection output is an OpenDRIVE map database that is saved as an XODR file. The `LaneWidth`-based output file is marked with the postfix `_width`.

12.4.1. OpenDRIVE standard conformity

The output generated by road detection conforms to OpenDRIVE revision 1.5.

Notes on LaneWidth

The `LaneWidth` XODR files pass the XSD schema validation for OpenDRIVE 1.5.

The schema validation checks for OpenDRIVE 1.6 and OpenDRIVE 1.7 are passed, too, once the version number in the file header is increased accordingly.

Known issues for `LaneWidth` files: none.

Notes on LaneBorder

There are some caveats for the `LaneBorder`-based output files.

There are some inconsistencies to be aware of between the standard documentation and the respective XSD schema files for OpenDRIVE 1.5, OpenDRIVE 1.6.0, OpenDRIVE 1.6.1, and OpenDRIVE 1.7.0.

Known issues for `LaneBorder` files:

- In OpenDRIVE 1.5, the XSD schema does not permit `LaneBorder` entries in the center lane. This is essential to provide usable output and was rectified in later versions.
- In OpenDRIVE 1.6.0, OpenDRIVE 1.6.1, and OpenDRIVE 1.7.0, the XSD schema file erroneously does not permit `UserData` records for `LaneBorder` entries.

If you encounter any of the above issues when importing the `LaneBorder`-based output file, you may want to try the `LaneWidth`-based output file instead.

12.4.2. OpenDRIVE elements in road detection

Element	Description
OpenDRIVE	The xmlns attribute is omitted.
OpenDRIVE/header	• The name attribute matches the output filename. • The date attribute corresponds to the creation date and time of the output file. • The version denotes the version of the revision of the OpenDRIVE standard.
OpenDRIVE/header/ geoReference	The proj4 projection string defines the georeference of the map, specifically the corresponding UTM projection and its origin within the UTM zone. For information on the parameters of the proj4 projection string see Georeference (page 75).
OpenDRIVE/header/ offset	• The values of the hdg, x, and y attributes correspond to the relative distance between the map origin and the origin of the UTM zone (as defined in geoReference). • The z attribute gives the height of the reference origin. This is generally the altitude above mean sea level according to WGS84.
OpenDRIVE/header/ userData → code="RoadsVersion"	Denotes the version of MVIS Auto Annotation that was used to generate the output file.
OpenDRIVE/header/ userData → code="latitudeDeg"	Map origin in WGS84 coordinate (latitude)
OpenDRIVE/header/ userData → code="longitudeDeg"	Map origin in WGS84 coordinate (longitude)
OpenDRIVE/road	• The id attribute is always an integer and unique across the map. • The junction attribute is always **-1** because the road detection algorithm does not provide junction detection.
OpenDRIVE/road/ planView	This element is present in all road elements. It contains exactly one geometry element.
OpenDRIVE/road/ planView/geometry	• This element occurs exactly once in every planView element, and not more than once. • The attributes (s, x, y, hdg and length) are all included as specified by OpenDRIVE Rev 1.5. • One paramPoly3 child element is included in every geometry element. No other child elements are used.
OpenDRIVE/road/ planView/geometry/ paramPoly3	• The pRange attribute is always set to **arcLength**. • The remaining attributes (aU, bU, cU, dU, aV, bV, cV, dV) are all included as specified by OpenDRIVE Rev 1.5.

Element	Description
`OpenDRIVE/road/link`	This element is present for all roads that have either a predecessor link or a successor link.
`OpenDRIVE/road/link/predecessor` `OpenDRIVE/road/link/successor`	The `predecessor` and `successor` elements are present when there is a corresponding linked road. • The `elementType` attribute is always **`road`**. • The `elementId` attribute is the ID of the linked road. • The `contactPoint` attribute is always **`start`** for successor elements and **end** for predecessor elements.
`OpenDRIVE/road/lanes`	This element is present for all `road` elements. It contains exactly one `laneSection` element.
`OpenDRIVE/road/lanes/laneOffset`	Applies to `LaneWidth`-based output only. Provides a geometric baseline relative to which the lane width elements are defined. At least one element per road.
`OpenDRIVE/road/lanes/laneSection`	This element occurs exactly once in every `road` element. The attribute `s` is always **`0.0`**.

`OpenDRIVE/road/lanes/laneSection/left`
`OpenDRIVE/road/lanes/laneSection/center`
`OpenDRIVE/road/lanes/laneSection/right`
	These elements will be included when they contain child `lane` elements. Otherwise, they are omitted.
	As specified per OpenDRIVE Rev 1.5, the center tag is always present and contains a non-drivable lane that serves as a reference to left and right lanes.

`OpenDRIVE/road/lanes/laneSection/*/lane`	• The `id` attribute is always **`0`** or negative because lanes are detected in driving direction only. • The `type` attribute is **`driving`** for drivable lanes, **`stop`** for emergency stopping lanes and otherwise **none** (for example for placeholder lanes).
`OpenDRIVE/road/lanes/laneSection/*/lane/link`	This element is available for all lanes that have predecessor and/or successor links.
`OpenDRIVE/road/lanes/laneSection/*/lane/link/predecessor` `OpenDRIVE/road/lanes/laneSection/*/lane/link/successor`	• The `predecessor` and `successor` elements are available when there is a corresponding linked lane, in a linked road. • The `id` attribute is set to the ID of the linked lane.
`OpenDRIVE/road/lanes/laneSection/*/lane/border`	• Applies to `LaneBorder`-based output only. • Polynomial description of a lane border. This element occurs at least once per `lane` element. • Attributes (`sOffset`, `a`, `b`, `c`, `d`) are defined as specified in OpenDRIVE Rev 1.5.

Element	Description
OpenDRIVE/road/lanes/laneSection/*/lane/border/userData → code="accuracy[auto-gen]"	• Applies to LaneBorder-based output only. • Accuracy estimation for lane border geometries. Encodes positional uncertainty in meters. Note that this internal probabilistic quantity is not to be confused with actual accuracy compared to ground truth geometries, respectively.
OpenDRIVE/road/lanes/laneSection/*/lane/width	• Applies to LaneWidth-based output only. • Polynomial description of a lane border. This element occurs at least once per lane element. • Attributes (sOffset, a, b, c, d) are defined as specified in OpenDRIVE Rev 1.5.
OpenDRIVE/road/lanes/laneSection/*/lane/roadMark	• This element occurs exactly once per lane element. • The width attribute is set based on the measured marking width in meters. • The type attribute is a value from {**solid**, **broken**, **solid solid**, **solid broken**, **broken solid**}. In case the type could not be determined, the typeConfidence value is set to **0.0**. • The color attribute is always set to **standard**.
OpenDRIVE/road/lanes/laneSection/*/lane/roadMark/userData → code="typeConfidence[auto-gen]"	Confidence value for road marking type classification. Range: **[0,1]**, with **0.0** corresponding to the lowest confidence and **1.0** to the highest.
OpenDRIVE/road/lanes/laneSection/*/lane/userData → code="existenceConfidence[auto-gen]"	This element occurs at least once per lane element that has a different ID than **0**. The existenceConfidence attribute indicates an algorithmic existence confidence for the respective lane. Range: **[0,1]**, with **0.0** corresponding to the lowest confidence and **1.0** to the highest.
OpenDRIVE/road/objects	This element is present for all roads that have at least one object.
OpenDRIVE/road/objects/object	• This element is used to represent detected road barriers. • The id, height, t, s, and length attributes are included as specified by OpenDRIVE. • The type attribute is fixed as **barrier** and subtype is one out of **custom** and **curbstone**. • The remaining attributes have fixed values: – dynamic: **no** – orientation: **none** – hdg, pitch, roll, width, zOffset: **0.0**

Element	Description
`OpenDRIVE/road/objects/object/repeat`	• This element is present for all objects. • The `distance` attribute is always **`0.0`**. That means the object is continuous. Note that therefore the `length` attributes (`lengthStart`, `lengthEnd`) are omitted. • The standard attributes (`s`, `length`, `tStart`, `tEnd`, `widthStart`, `widthEnd`, `heightStart`, `heightEnd`, `zOffsetStart`, `zOffsetEnd`) are all included. • The `zOffsetStart` and `zOffsetEnd` are always **`0.0`**.
`OpenDRIVE/road/userData → code="timeStamp[auto-gen]"`	Provides a time stamp that associates each road with the corresponding instant in the recording that is used to generate the output file. The time stamp of the ego vehicle position that matches the start point of each road is sampled. The formatting of the time stamp is `YYYY-MM-DDThh:mm:ss.ffffff` in accordance to ISO 8601.
`OpenDRIVE/road/signals`	This element is present for all roads that contain at least one signal (traffic sign or traffic lights).
`OpenDRIVE/road/signals/signal`	This element contains the properties of a signal (traffic sign or traffic lights). • The `type` attribute contains the signal type. • The `subtype` attribute contains the signal subtype. • The `s` attribute contains the `s` position in the current road. • The `t` attribute contains the `t` position in the current road. • The `id` attribute contains a unique signal ID. • The `dynamic` attribute encodes whether the element changes over time, for example, in case of a traffic light. • The `name` attribute contains a generic name of the object, for example, **`trafficSign`** or **`trafficLight`**. • The `zOffset` attribute contains the height above ground in meters of the bottom end of the signal element. • The `height` attribute contains the height in meters of the signal's face.

12.4.3. Georeference

Example "geoReference" tag

```
<geoReference>
     <![CDATA[+proj=utm +zone=32 +south +datum=WGS84 +units=m +no_defs +type=crs]]
</geoReference>
```

Overview of parameters

Parameter	Description
`proj`	UTM projection is used.

Parameter	Description
`zone`	UTM zone value
`south`	Optional: in case that the transformation is used on the southern hemisphere.
`datum`	Datum used with the coordinates.
`units`	Meter, US survey feet, etc.
`no_defs`	No default values
`type`	Projection type

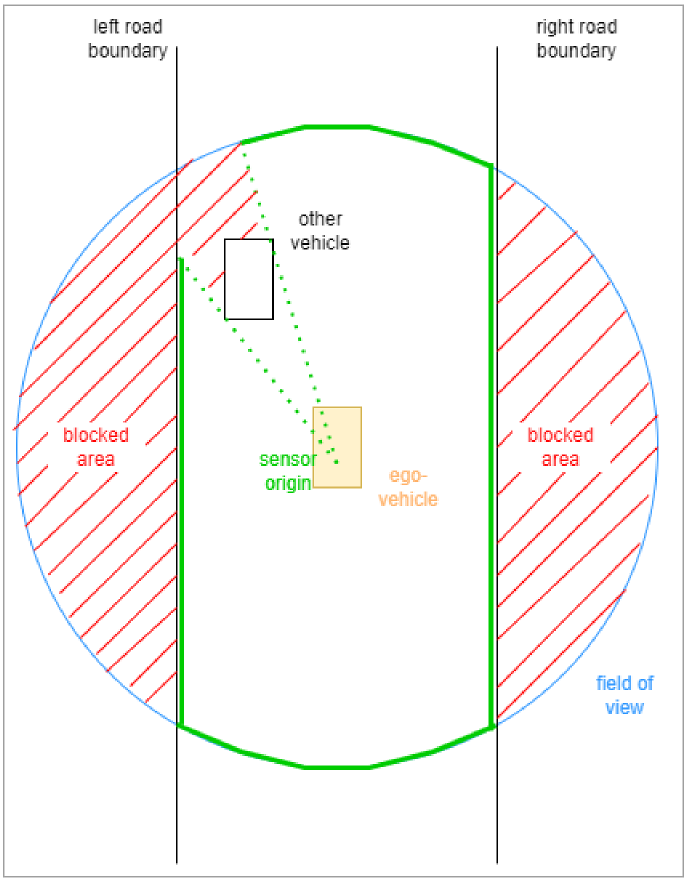
13. Free space detection

With free space detection, you can automatically calculate free space based on road boundaries and potentially dynamic objects in a highway environment.

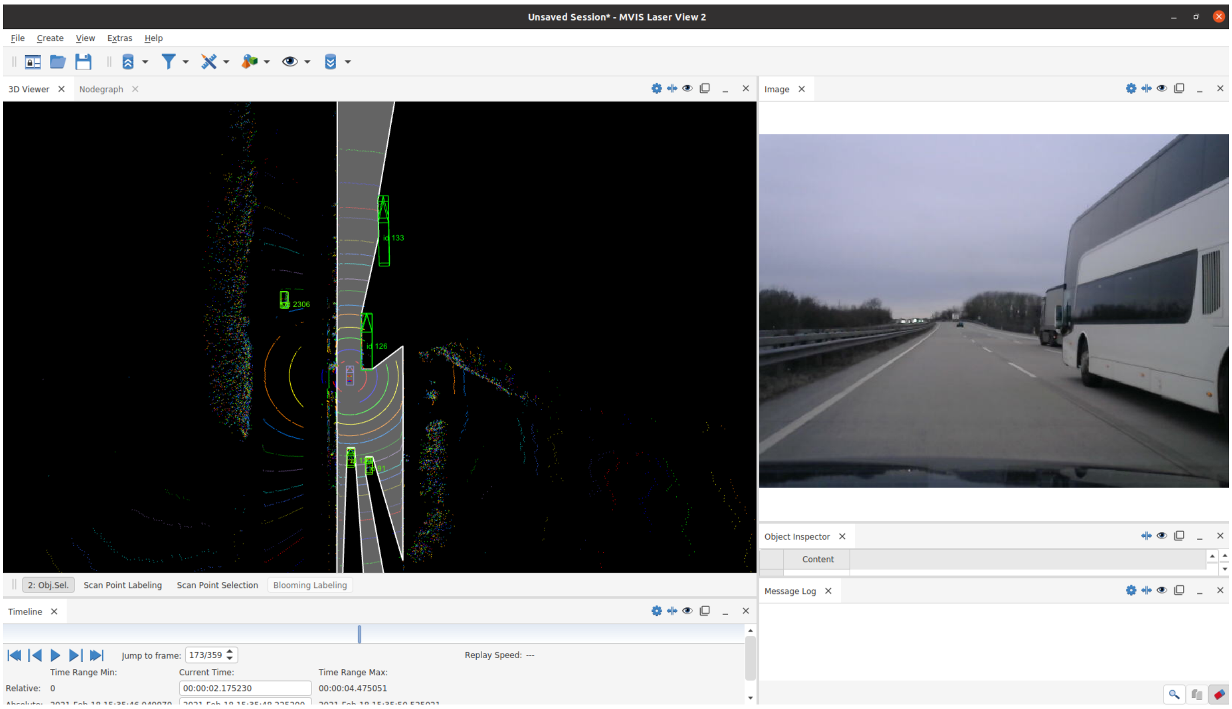
Free space is defined as the area which meets the following conditions:

- The space is not blocked by potentially dynamic objects.
- The space is not blocked by road boundaries.
- The space is within the sensor field of view.

As shown in the following image, free space (outlined in green) is computed based on the sensor field of view (outlined in blue). Road boundaries and objects and the areas they block, that is the areas that are not free space (in red) are removed.



Resulting free spaces are exported in a CSV file in the form of well-known text (WKT). Free space results can be visualized in MVIS Laser View 2:



13.1. Limitations for free space detection

- Free space detection is developed only for highway environment.
- Free space detection is not based on point cloud.
- Free spaces are computed based only on road boundaries and potentially dynamic objects. Parked cars, for example, will be used for computation but trees will be ignored.
- Free space detection is developed in 2.5 dimensions (2.5D). That means, free space detection can only provide 2D free spaces on XY-plane for different heights.
- Free space detection is able to export interpolated free spaces at specific given timestamps from a TXT file. If no timestamps are specified, free spaces will be computed at object list timestamps.
  Given timestamps should have the following format:
  - Format: `Y-M-D hh:mm:ss`
  - Example: `2021-Mar-24 10:44:44.879498`
- Sensor fields of view are in 2D. If sensors are mounted with large pitch angle or roll angle, the field of view of this sensor will be projected onto XY-plane for free space computation.

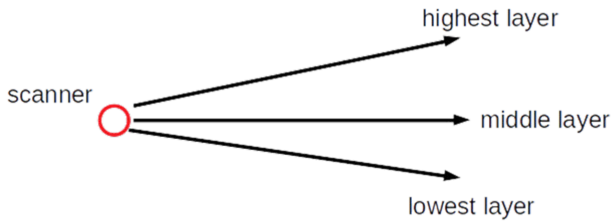
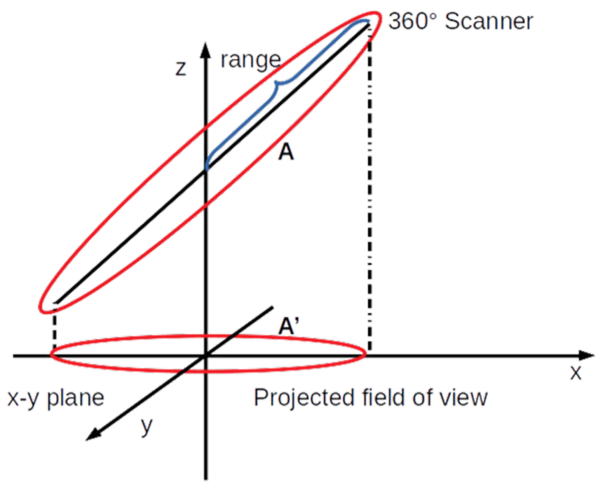
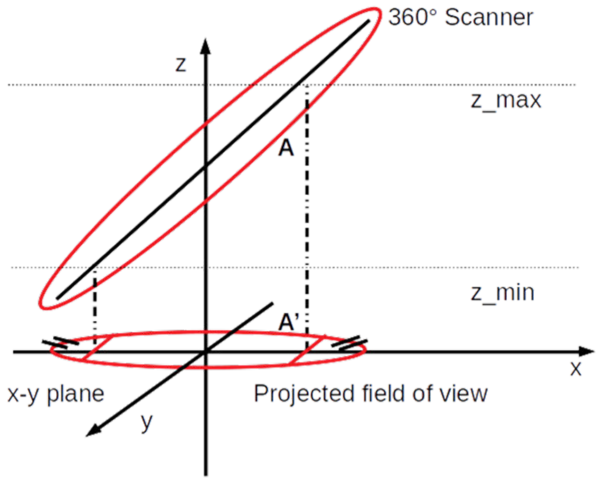
13.2. Prerequisites for free space detection

This feature depends on:

- Ego state estimation
- Object tracking
- Road detection (optional)

13.3. Projection

3D scanner fields of view will be projected onto XY-plane in the following steps.

Step	Description	Illustration
1	Create a field of view area **A** with start angles, end angles, and a maximum range.	–
2	Take **A** as the middle layer (horizontal layer of the scanner) and transform it to **A'** according to the mounting position and the yaw, pitch, roll angles.	
3	Project the transformed layer **A'** onto the XY-plane and take the result as the field of view of this scanner.	
4	Set a maximum height and a minimum height so that points higher than the maximum or lower than the minimum are not considered as valid.	

13.4. Output description for free space detection

The output of free space detection is located in the output path in the `freespace` folder.

The free space output is a CSV file.

Structure of the CSV file

```
ScanID;Layer;Height;ObjectListTS;FreespacePolygon
n;band_name;h;ts;wkt
```

Description of the CSV file

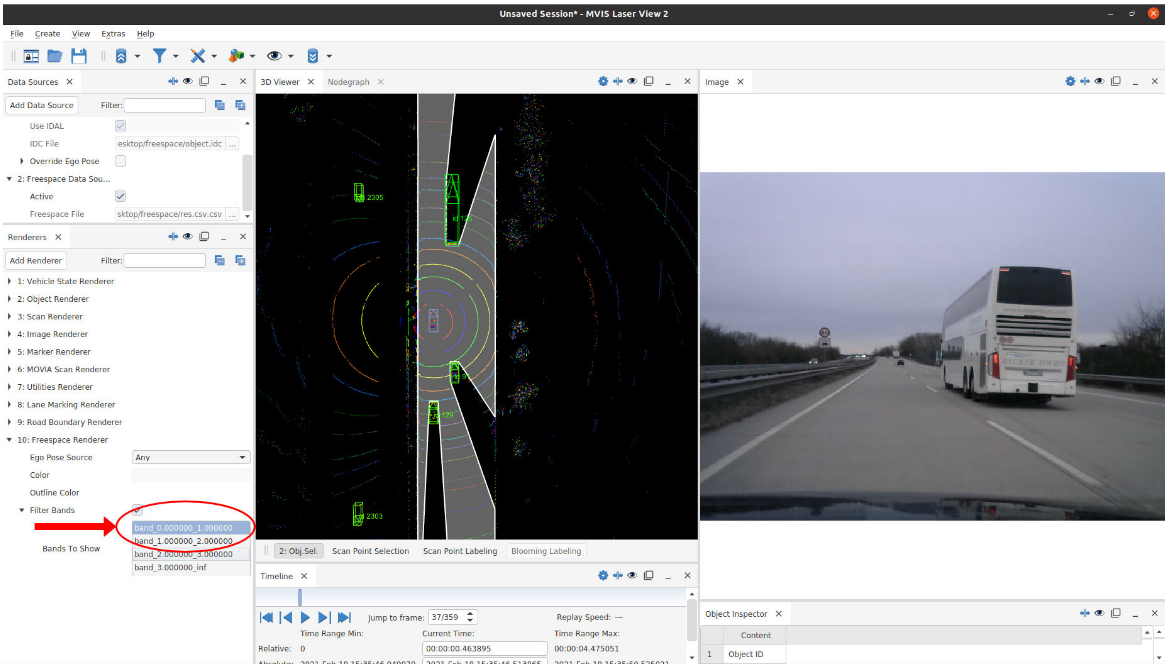
Value	Description
`n`	`n` is an integer number with the following properties: • `n = -1` if the `wkt` of that line denotes a maximum field of view which is valid starting from timestamp `ts`. • `n > 0` (increasing sequentially, starting from `0`) if the `wkt` of that line denotes the free space at the timestamp `ts`. That means `0` denotes the first free space polygon, `1` the second etc.
`band_name`	`band_name` is a string that describes the height interval within which the free space is computed. If the interval is `(0.333, 0.666)` the `band_name` is `band_0.333000_0.666000`. For example height intervals, see Example height intervals (page 81).
`h`	`h` is a double value that describes the higher value of the band. This value is used as a hint to the visualization how to render the specific free space. For the band `band_0.333000_0.666000` this value is `0.666`.
`ts`	`ts` is a timestamp in `YYYY-mm-dd HH:MM:ss` format. The seconds are floats.
`wkt`	`wkt` is a string in well-known text format. In the current implementation, the string is always a `MULTIPOLYGON` and its scope is the height interval described by `band_name`.

Example CSV file

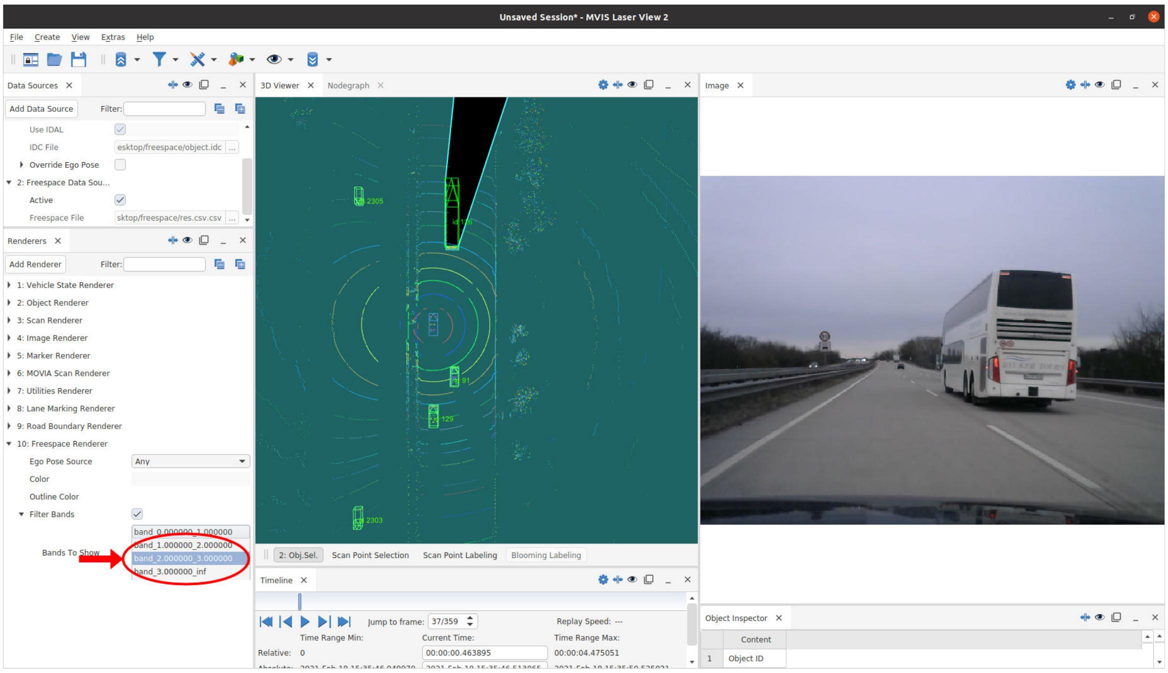
```
ScanID;Layer;Height;ObjectListTS;FreespacePolygon
-1;fov;0;-1;MULTIPOLYGON(…)
0;band_0.000000_0.333000;0.333000;2022-01-01 12:00:00.00000000;MULTIPOLYGON(…)
0;band_0.333000_0.666000;0.666000;2022-01-01 12:00:00.00000000;MULTIPOLYGON(…)
0;band_0.666000_1.000000;0.666000;2022-01-01 12:00:00.00000000;MULTIPOLYGON(…)
0;band_1.000000_inf;          inf;2022-01-01 12:00:00.00000000;MULTIPOLYGON(…)
-1;fov;0;-1;MULTIPOLYGON(…)
1;band_0.000000_0.333000;0.333000;2022-01-01 12:00:01.00000000;MULTIPOLYGON(…)
1;band_0.333000_0.666000;0.666000;2022-01-01 12:00:01.00000000;MULTIPOLYGON(…)
1;band_0.666000_1.000000;0.666000;2022-01-01 12:00:01.00000000;MULTIPOLYGON(…)
1;band_1.000000_inf;          inf;2022-01-01 12:00:01.00000000;MULTIPOLYGON(…)
```

Example height intervals

- The height interval is set to 0 m to 1 m (`band_0.000000_1.000000`).
  Within this interval, road boundaries and objects will be considered for free space computation.



- The height interval is set to 2 m to 3 m (`band_2.000000_3.000000`).
  Within this interval, there are no road boundaries, only trucks. Therefore, the free space of the same scene looks different.



14. Scenario detection

With scenario detection, you can automatically extract scenario data from your recorded trip data.

14.1. Limitations for scenario detection

The following limitations for scenario detection apply:

- The scenario detection is mainly intended for data that was recorded on a highway.
- A common distance limit of 150 meters applies for reference objects due to reduced positional accuracy at this distance.
- Most labels are only exported if they are valid for longer than a minimal duration (for example 1 second). In case of micro interruptions due to inaccuracies in the inputs, labels of the same type are merged. This ensures a smoother timeline for the labels.
- If reference data is processed (as opposed to ground truth):
  - Limitations of the previous processing steps also apply to the scenario detection.
  - Issues in the previous processing steps directly affect the quality of the exported labels.

14.2. Prerequisites for scenario detection

Scenario detection is based on high-level data and relies on the following previous processing steps:

- Ego state estimation
- Object tracking
- Road detection

14.3. Basic scenarios

The automated labeling covers basic scenarios, for example:

- Traffic participants and traffic volume
- Properties of the test vehicle (speed, acceleration, driving lane)
- Road layout (number of lanes, curvature, inclination)
- Driving maneuvers (lane change, overtaking, follow drive)
- Time of day (day, night, dawn/dusk)

14.4. Labels

A label consists of the following:

- A label category
- A scenario element name
- Start time and end time
- An ID that is unique within a trip
- A string that indicates if the label source was automated or manual.

Where applicable, IDs of the involved traffic participants are included.

14.5. Traffic participant IDs

Traffic participant IDs are split into the following:

- Active traffic participant (for example, a vehicle that overtakes another vehicle)
- Passive traffic participant (for example, a vehicle that is being overtaken)

The vehicle IDs can be cross-referenced with the exported traffic participants to obtain the vehicle type.

14.6. Features for scenario detection

Unless otherwise specified, "distances" always refer to the distances to the rear center of an object from the center rear axis of the ego vehicle in scenario detection.

14.6.1. Time of day

The time of day labels are grouped under the "time" scenario category with the options day, night, and dawn/dusk.

Civil twilight definition is used to distinguish dawn/dusk from night.

Prerequisites
This feature depends on:

GPS information

14.6.2. Location

Location labels map GPS information to a continent and a country. These labels are grouped under the "continent" and "country" scenario category.

Prerequisites
This feature depends on:

GPS information

14.6.3. Velocity and acceleration ranges of the test vehicle

Labels for the test vehicle's velocity and acceleration are exported for specific ranges.

Velocity labels with gaps of less than 5.0 seconds are merged. Velocity labels with a duration of less than 0.5 seconds are not exported. These labels are grouped under the "ego property" scenario category.

Acceleration labels with gaps of less than 1 second are merged. Acceleration labels with a duration of less than 0.1 second are not exported.

Prerequisites
This feature depends on:

Ego state estimation

Labels for ego velocity

- Ego velocity [-1000.0:- 0.1] kmph
- Ego velocity [-0.1:0.1] kmph
- Ego velocity [0.1:30] kmph
- Ego velocity [30:50] kmph
- Ego velocity [50:70] kmph
- Ego velocity [70:80] kmph
- Ego velocity [80:100] kmph
- Ego velocity [100:130] kmph
- Ego velocity [130:1000] kmph

Labels for ego acceleration

- Ego acceleration [-100.0:-4.0] m/s^2
- Ego acceleration [-4.0:-2.0] m/s^2
- Ego acceleration [-2.0:-0.5] m/s^2
- Ego acceleration [-0.5:0.5] m/s^2
- Ego acceleration [0.5:2.0] m/s^2
- Ego acceleration [2.0:4.0] m/s^2
- Ego acceleration [4.0:100.0] m/s^2

14.6.4. Number of lanes, driving lane of the test vehicle, and road boundaries

Labels are exported for the number of lanes of the road that the test vehicle is on and the lane number the test vehicle is in as well as the presence of stop lanes.

Only the following lanes are counted:

- Lanes in the driving direction of the test vehicle.
- Lanes that were detected as driving lanes in the road detection processing.

Labels with gaps of less than 0.5 seconds are merged. Labels with a duration of less than 0.5 seconds are not exported. This feature is optimized for highway driving situations.

Prerequisites
This feature depends on:

- Ego state estimation
- Road detection

Lane number labels
The number of lanes is labeled according to these elements and is grouped under the "lane type" scenario category:

- Single lane one-way
- 2-lanes one way
- 3-lanes one way
- Multi-lane one way
- stop lane

Ego vehicle position labels
The position of the ego vehicle is labeled according to the standard of the OpenDRIVE format. The left-most lane in driving direction is lane -1 with decreasing numbers toward the right. These labels are grouped under the "ego property" scenario category:

- ego in lane -1
- ego in lane -2
- ego in lane -3
- ego in lane -4
- ego in lane -5
- ego in lane < -5
- ego in lane 1
- ego in lane 2
- ego in lane 3
- ego in lane 4
- ego in lane 5
- ego in lane > 5
- ego in stop lane

Road boundary labels
Labels are created to indicate the type of road boundary present. These labels are grouped under the "static object" scenario category:

- Guard rail
- Guard rail left-hand side
- Guard rail right-hand side

> **NOTE**
> Guard rail labels include concrete traffic barriers.

Road boundary labels of the same type which are interrupted by less than 200 milliseconds are merged. Then, road boundary labels with a duration shorter than 200 milliseconds are removed.

14.6.5. Road course

The labeling of the road curvature radius and the road inclination is handled via properties of the test vehicle. This data is inflicted with two issues: erratic steering movements and bumps in the road surface. Both can be mitigated by smoothing the original data, which affects the start time and end time of the exported labels. As a compromise between these two issues, the data is smoothed with a running mean in the range of ± 50 meters.

Road curvature labels, which are based on the steering angle of the test vehicle, usually exhibit a common sequential structure (straight → slight curve → strong curve → slight curve → straight), because a driver enters a curve gradually. These sequences are merged, maintaining the strongest curvature label with the duration of the full sequence. This merging also applies if a road truly does exhibit this sequential structure.

Labels with a duration of less than 1 second are deleted. Labels with gaps of less than 1 second are merged.

Prerequisites
This feature depends on:

Ego state estimation

Slope data labels
The slope data is labeled according to the following elements and is grouped under the "road layout" scenario category:

- road inclination [-100:-6) %
- road inclination [-6:-2) %
- road inclination [-2:2) %

- road inclination [2:6) %
- road inclination [6:100) %

Curvature radius labels
For the curvature, the following labels are placed and are grouped under the "road layout" scenario category:

- right curve with radius [1:120) m
- right curve with radius [120:450) m
- right curve with radius [450:1000) m
- right curve with radius [1000:2500) m
- straight road with with radius > 2500 m
- left curve with radius (1000:2500] m
- left curve with radius (450:1000] m
- left curve with radius (120:450] m
- left curve with radius (1:120] m

Limitations

- Road curvature labels are not exported within a time range of plus minus two times the duration of a lane change of the test vehicle.
- If reference data is processed, undetected lane changes of the test vehicle can cause a sequence of left and right curves if the lane information is imperfect.
- Right turns and left turns at intersections are labeled as curves.

14.6.6. Traffic participants

Traffic participant labels are exported for every detected object in a recorded drive. The output contains the classification and the ID of the participants. These labels are grouped under the "traffic participant" scenario category.

Prerequisites
This feature depends on:

Object tracking

14.6.7. Objects in longitudinal range

When traffic participant labels are requested, in addition labels are created to indicate whether there is any traffic participant present in certain longitudinal ranges ahead of the ego vehicle.

A traffic participant is within the longitudinal range if the center of the object box is in the given longitudinal range.

Prerequisites
This feature depends on:

Object tracking

Labels
The following labels are set in the "traffic volume" scenario category:

- Object in longitudinal range [0; 30] m
- Object in longitudinal range [30; 60] m
- Object in longitudinal range [60; 90] m
- Object in longitudinal range [90; 120] m
- Object in longitudinal range [120; 150] m

14.6.8. Traffic density

The traffic density is categorized into sparse, medium, and dense traffic. The density is given as a number of traffic participants per distance and per lane. The chosen limits are:

- Sparse: traffic density < 0.0078
- Medium: 0.0078 <= traffic density < 0.03
- Dense: 0.03 <= traffic density

Labels with gaps of less than 50 seconds are merged. Labels with a duration of less than 10 seconds are deleted. These labels are grouped under the "traffic volume" scenario category.

Prerequisites
This feature depends on:

- Object tracking
- Lane detection

14.6.9. Traffic jam

Traffic jams are labeled under the traffic volume category on highways if the following conditions are given:

- Vehicle comes to a full stop.
- Average velocity is lower than 20 kmph.

Labels with gaps of less than 30 seconds are merged. Labels with a duration of less than 10 seconds are deleted.

Prerequisites
This feature depends on:

- Ego state estimation
- Online user tag "highway"

14.6.10. Road user behavior: lane change and cut-in

Lane changes are labeled if a traffic participant completely switches from one driving lane to another.

The following sequence identifies a lane change:

1. Start: an object center gets closer than 1.0 meters to the lane boundary.
   The distance to the lane boundary at the start and end condition of 1.0 meters is motivated by the average width of compact cars.
2. Middle: the object crosses the lane boundary.
3. End: the object center moves away further than 1.0 meters from the lane boundary on the other side.

The following sanity checks guard against invalid lane change labels:

- The traffic participant has a positive speed.
- The lane change duration is expected to be smaller than 10 seconds.
- Gaps in the lane information of lane changing traffic participant may not exceed 1 second.

Additionally, a cut-in is labeled if a vehicle changes into the driving lane of the test vehicle. The following conditions are checked:

- A vehicle performs a lane change into the driving lane of the test vehicle.
- No other traffic participant is between the vehicle and the test vehicle in the driving lane.
- Negative relative velocity of the other vehicle (configurable).
- The other vehicle is within 70 meters (configurable).

Prerequisites
This feature depends on:

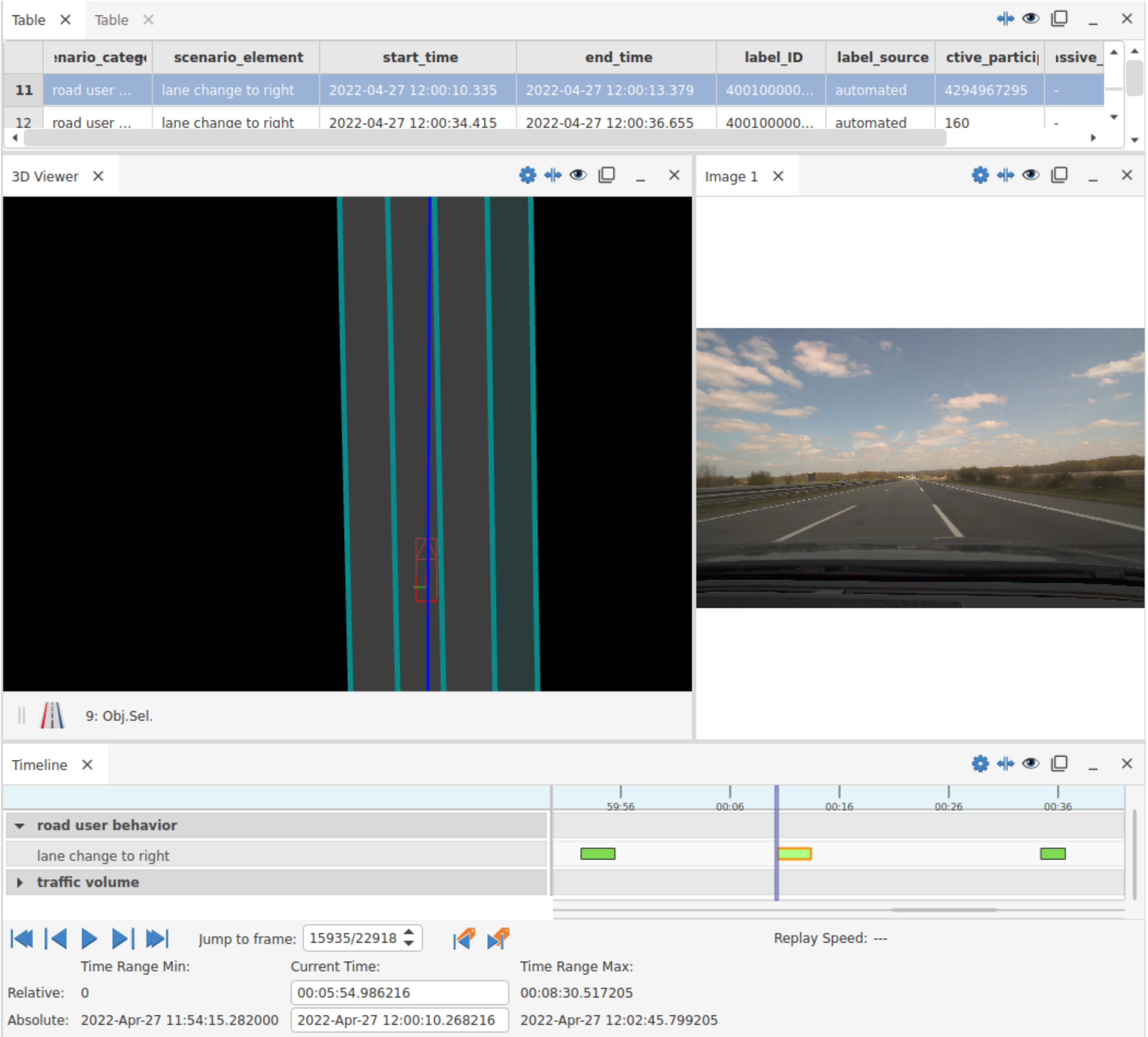
- Ego state estimation
- Object tracking
- Road detection

Labels

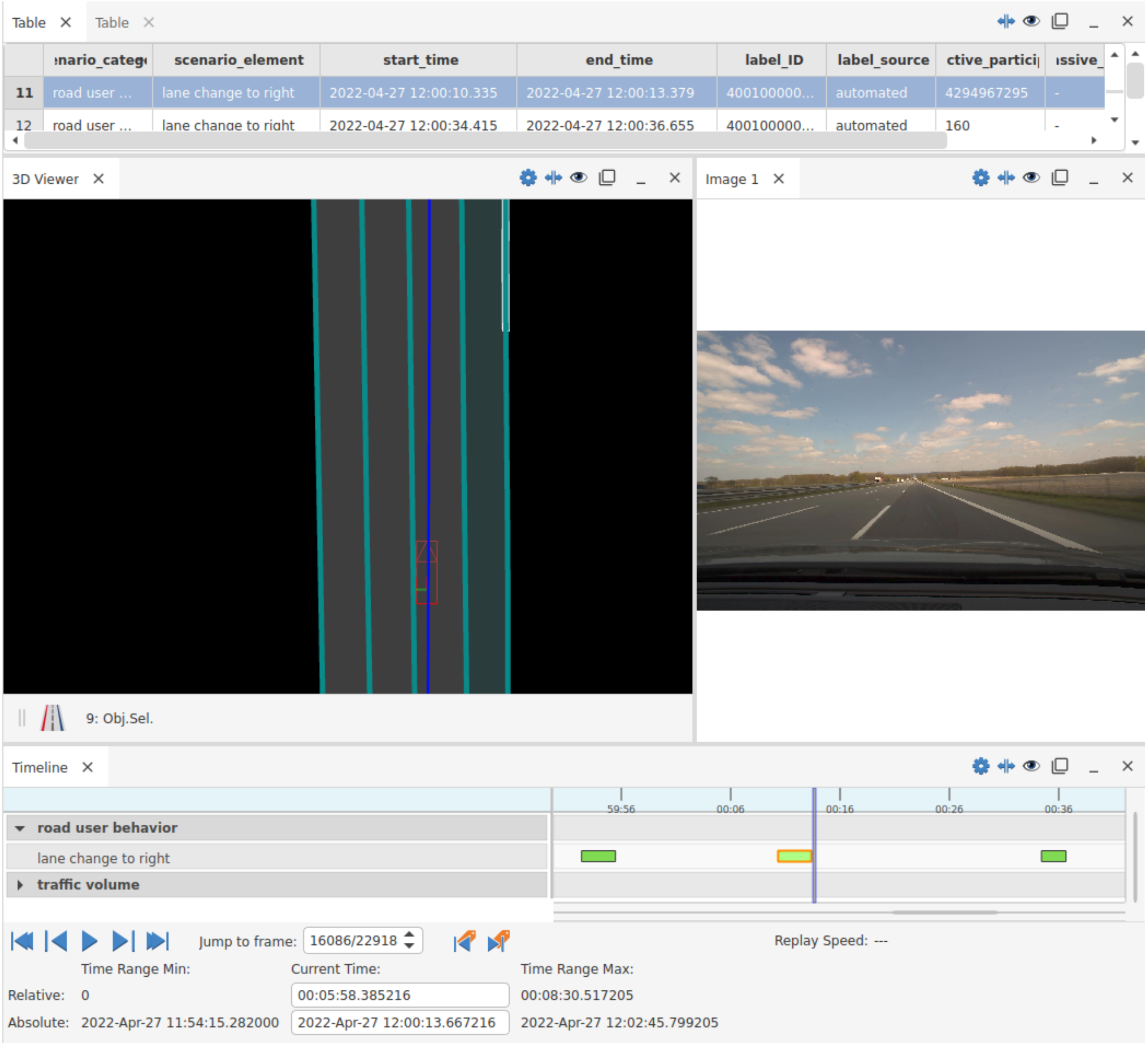
- Lane change to the left
- Lane change to the right
- Cut-in from left

- Cut-in from right

Start of a lane change of the test vehicle



End of the lane change



14.6.11. Road user behavior: overtaking

Overtaking maneuvers are labeled if a traffic participant overtakes the test vehicle or the test vehicle overtakes another traffic participant in adjacent lanes.

Two labels are exported for each maneuver: a general overtaking on the left-hand side or right-hand side and a label specifying if the maneuver was started or ended by a lane change of the overtaking vehicle.

The following sequence identifies an overtaking maneuver:

1. Start: the distance of the active vehicle becomes smaller than the negative distance limit + the relative velocity exceeds 5 km/h + the absolute velocity of both vehicles exceeds 10 km/h.
   The distance limit is half of the passive vehicle velocity in meters, for example, 60 meters for a speed of 120 km/h, with a minimal distance limit of 10 meters.

2. Middle: when the vehicles are beside each other their distance is smaller than 6 meters.
3. End: the distance to the active vehicle becomes larger than the positive distance limit.

Prerequisites

This feature depends on:

- Object tracking
- Ego state estimation
- Internal lane change detection

Classification

A classification into different lane change patterns of the active vehicle is available:

- No lane changes
- Starting with a lane change
- Ending with a lane change
- Starting and ending with a lane change

Lane changes from the driving lane of the passive vehicle into the overtaking lane are included in the maneuver if they occur within twice the duration of the overtaking maneuver. Other lane changes of the active vehicle are excluded.

Lane changes of the passive vehicle are also excluded from the duration of the overtaking maneuver, for example, if the passive vehicle changes lane away from the overtaking vehicle.

If the classification is selected, all general overtaking maneuvers are maintained, but their duration is adapted.

General labels

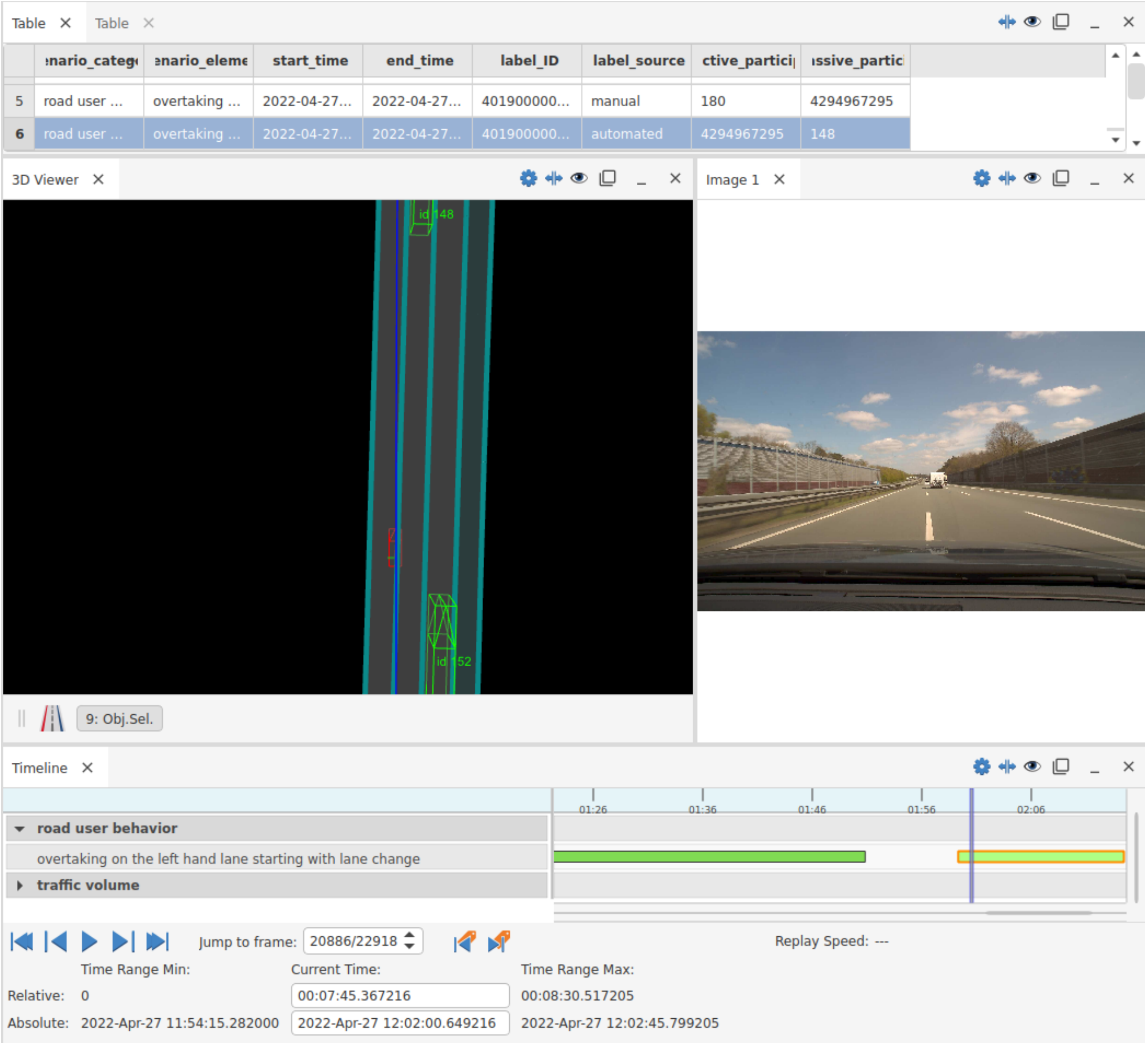
- Overtaking on the left-hand lane
- Overtaking on the right-hand lane

Specific labels

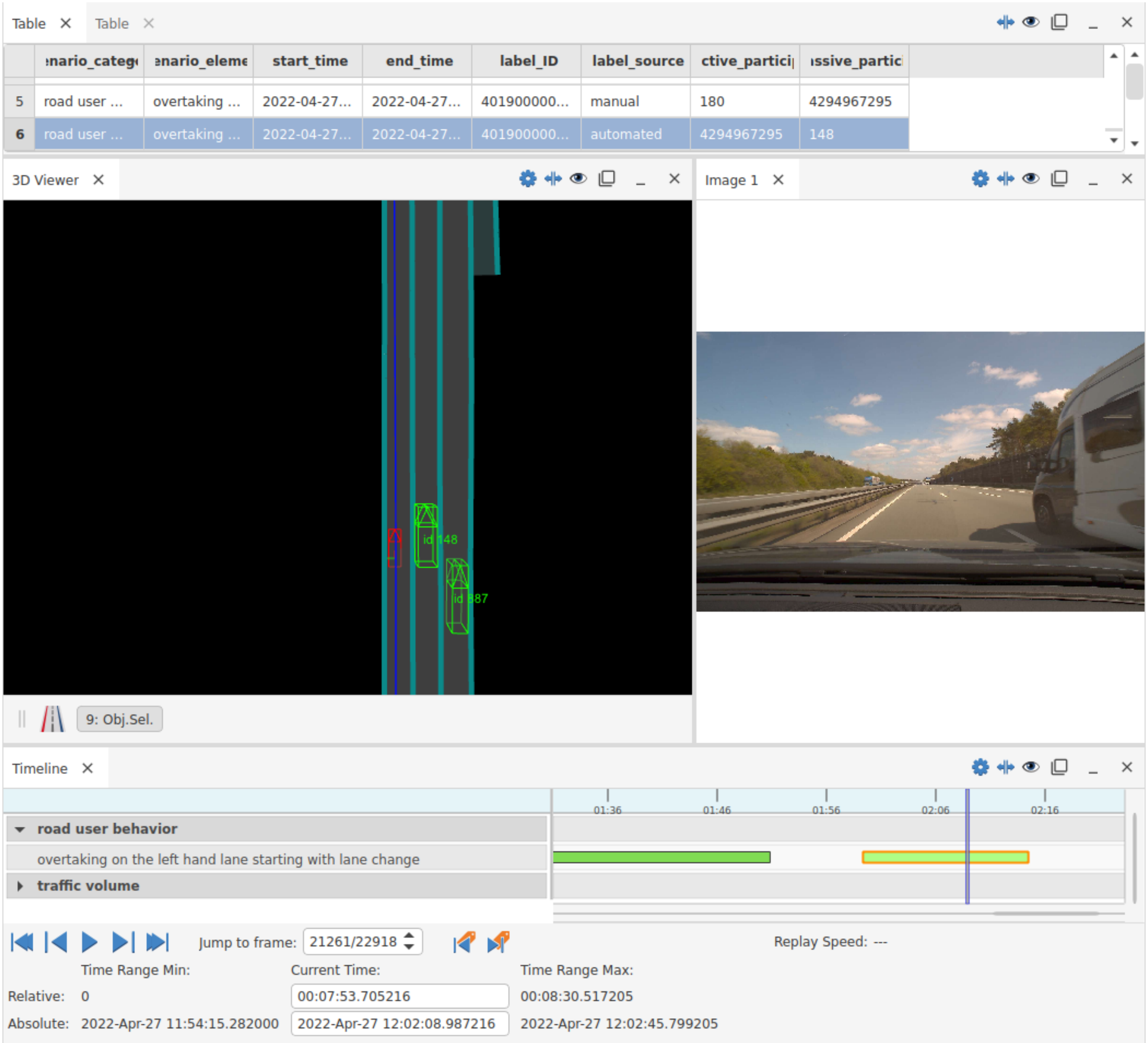
- Overtaking on the left-hand lane with no lane changes
- Overtaking on the left-hand lane starting with lane change
- Overtaking on the left-hand lane ending with lane change
- Overtaking on the left-hand lane starting and ending with lane changes
- Overtaking on the right-hand lane with no lane changes
- Overtaking on the right-hand lane starting with lane change
- Overtaking on the right-hand lane ending with lane change

- Overtaking on the right-hand lane starting and ending with lane changes

Start of an overtaking maneuver



Middle of an overtaking maneuver



14.6.12. Road user behavior: following preceding traffic participant

The ego vehicle is following a preceding traffic participant within a defined time gap.

Defined time gaps

- 0.5 - 1.5 seconds
- 1.5 - 2.5 seconds
- 2.5 - 3.5 seconds

Prerequisites

This feature depends on:

- Ego state estimation
- Road detection

- Object tracking

Labels

- following preceding traffic participant [0.5:1.5) sec gap (default; parametrizable interval)
- following preceding traffic participant [1.5:2.5) sec gap (default; parametrizable interval)
- following preceding traffic participant [2.5:3.5) sec gap (default; parametrizable interval)

14.6.13. Euro NCAP scenarios

The Euro NCAP testing protocols assess specific Safety Assist (ADAS) technologies in specific driving scenarios.

Relevant driving scenarios for the assessment are detected by the scenario detection to validate AEB (Autonomous Emergency Braking) Car-to-Car and LSS (Lane Support Systems).

Parameters are available to configure the relevant scenarios such as a minimum distance limit to lane borders that indicates a failed lane support system if surpassed or time spans of interest for time-to-collision detection.

The time-to-collision labels and blind spot labels include a reference to scenario evoking traffic participants.

Prerequisites
This feature depends on:

- Ego state estimation
- Object tracking
- Road detection

Labels for lane support systems

- driving in the blind spot on the left
- driving in the blind spot on the right
- lane keep failure broken lane marking
- lane keep failure solid lane marking
- lane departure warning broken lane marking
- lane departure warning solid lane marking

Labels for AEB Car-to-Car

- time to collision [x:y) s

Local hazards

- stopped vehicle on hard shoulder
- wrong way driver

14.7. Output description for scenario detection

The output of scenario detection is located in the output path in the `scenarios` folder.

The following files are exported:

- A list file (`scenariolabels.lst`) that contains the names of all files containing the actual labels.
- A meta information file that contains relevant information about the processed trip.
- One CSV file per scenario category that contains the labels.
  The labels are exported as comma-separated values (CSV) files. One file is exported for each scenario element category. In the individual files, the labels are sorted according to the scenario element and then according to the end time of the element.
  CSV files contain the following columns:

Column	Description
Scenario category	Category of the scenario element
Scenario element	Scenario element of the label
Start time	Start time of the label
End time	End time of the label.
Label ID	Label ID that is unique within a trip.
Label source	Source of the label For scenario detection, this will always be "automated."
ID active participant	ID of the active participant Only included if applicable.
ID passive participant	ID of passive participants Only included if applicable.