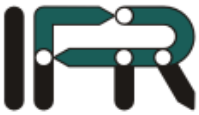
Technical University of Brunswick
Institute of Control Engineering
Department Vehicle Electronics
Prof. Dr.-Ing. Markus Maurer

# TUBS Labeling Tool

## User Manual

Author: Christopher Plachetka, M. Sc.
Published 01.07.2019

# Table of contents

# 1 Getting started

In this document, we explain the features of the TUBS labeling tool and provide an introduction into the basic concepts of semi-automatic labeling. Although we designed our tool to work with data originating from the TUBS Road User Dataset, you can integrate your own dataset as long as it provides consecutive frames. A description of the data format can be found on our website [1].

## 1.1 System requirements and setup

### 1.1.1 System requirements

Point cloud and image editing (3D bounding boxes):

- MATLAB R2015b or higher
- Mouse and keyboard
- 2x 24" monitors recommended

Image segmentation and personal data labeling:

- Image processing toolbox (used for editable rectangles and polygons)

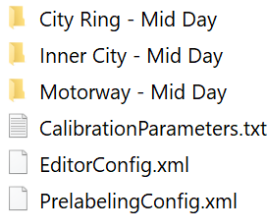
Camera calibration:

- Robotics toolbox (used for rotm2eul)

### 1.1.2 Setup

In order to setup our tool, follow these instructions:

1. Download the tool from GitHub [3]. We recommend cloning the maintained repository.
2. Register on our website and download a sensor recording, e.g. "Braunschweig City Ring - Mid Day".
3. Unpack the recording to a directory that contains the calibration and configuration files. You can use the provided directory \datasets\TUBS in the editor's directory.
    - CalibartionParameters.txt (camera intrinsics and extrinsics)
    - EditorConfig.xml (classes definition, colors, and editing history)
    - PrelabelingConfig.xml (e.g. last used object ID by the prelabeling stage)




4. Run PCEditorTool.m to start the GUI. If some tables are overlaid with scroll bars, close the window and restart the script.
5. Select the dataset directory (containing the calibration files) and the recording's directory (e.g. City Ring - Mid Day). The sequence and point cloud IDs are set automatically.
6. Press "Load" to get started.

## 1.2 Basic concepts

In this section, we briefly describe the concepts of our semi-automatic labeling procedure and the working principles of our editor.

### 1.2.1 Semi-automatic labeling

Our labeling procedure consists of two stages as shown in Fig. 1.1. In the prelabeling stage, we use our research vehicle's tracking system to label large amounts of point clouds automatically. This stage provides six classes: car, van, truck, motorbike, bicycle and pedestrian. Our tracking system is solely based on conventional algorithms. Although the system tracks moving objects reliably, it cannot detect unmoving objects. Hence, parking cars etc. need to be labeled manually.

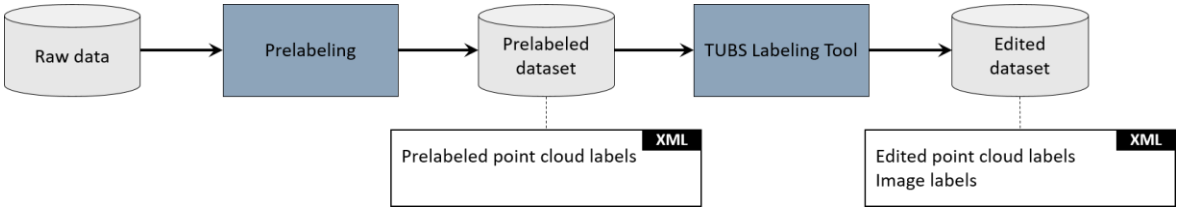



**Fig. 1.1** Semi-automatic labeling procedure

Automatically labeled data is suitable to train neural networks, as we have shown in [4]. The editor can be used to generate ground truth data or training data labeled with human performance in addition to automatically labeled data. The editor operates independently from the prelabeling stage. Point cloud labels are projected into the image planes in order to obtain labeled images with every labeled point cloud.

### 1.2.2 Editor concept

Our tool's concept is shown in Fig. 1.2. The tool minimizes manual editing effort by predicting the current object list into consecutive frames. After every prediction step, an optimization is applied. The tacking algorithm uses the manual corrections and the optimizations as input.

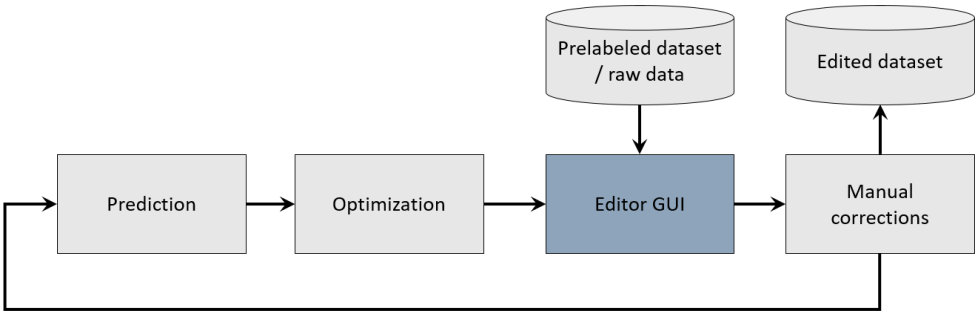



**Fig. 1.2** Editor concept

### Prediction

In order to track and predict point cloud objects, the editor uses the Interacting Multiple Model Kalman filter algorithm. For pedestrians, the editor uses only a point mass model with constant velocity. For all other classes, a point mass model with constant acceleration is fused with a circular path model with constant velocity.

### Optimization

In the optimization step, mainly three algorithms are used. At first, we apply the iterative closest point algorithm to register the current object point cloud with the previous one in order to obtain a relative transformation.

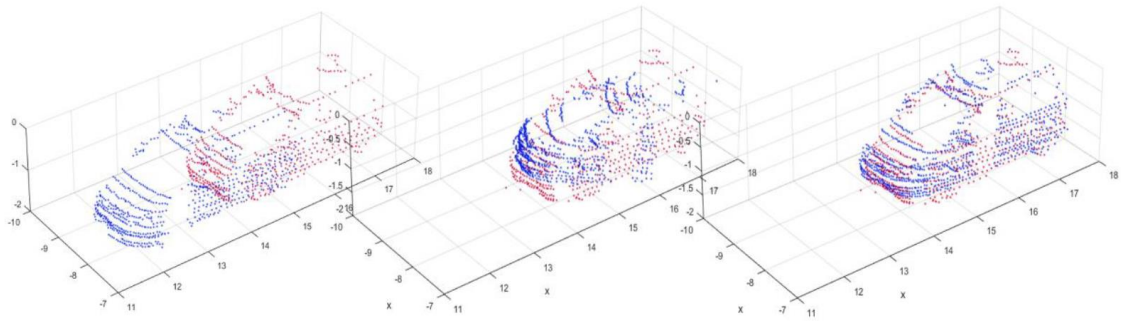


**Fig. 1.3** Iterative closest point algorithm to register object point clouds

After registering, the editor attempts to fit the predicted bounding box onto the object point cloud. Therefore, the dominant edges are computed using the RANSAC algorithm. The bounding box is then fitted by minimizing the distance to three target points on the dominant edge lines. If the rectangle-fitting algorithm fails, an ellipse is fitted using multiple algorithms.

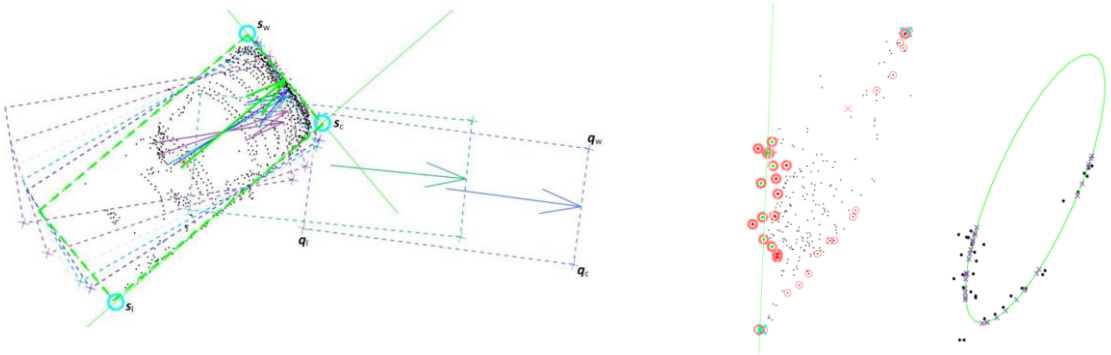



**Fig. 1.4** Left: Rectangle-fitting algorithm. Right: Fitted ellipse as fallback algorithm.

The editor provides a self-assessment of the prediction and optimization quality based on the variances of the different estimations compared to the prediction.

# 2 Overview

## 2.1 Features

**Table 2.1** Editor features

| Feature | Description | Section |
|---|---|---|
| Classes configuration | The set of object classes can be configured freely using the editor's configuration file. | 2.3 |
| Point cloud labeling | Point clouds are labeled in bird's eye view using an algorithm for easy object definition. The prelabeling stage is used to import new objects automatically. The correction mode selects uncertainly predicted objects for editing in order to minimize manual editing effort further. Moreover, the editor handles the deletion process of objects. | 3.1 |
| Projection of 3D objects | Objects defined in the point cloud are projected as 3D shapes into the image planes to obtain corresponding labeled images. | 4.1 |
| Coarse segmentation | In addition to 3D shapes, polygons can be projected and edited to generate coarsly segmented images. | 4.2 |
| Image label tracking | 3D shapes or polygons in the image planes are tracked using correspondences with point cloud objects. Additionally, user defined shapes can be tracked by manually setting correspondences. | 4.3 |
| Camera calibration | The editor has an integrated camera to laser scanner calibration functionlity in case you would like to label your own datasets. | 5 |

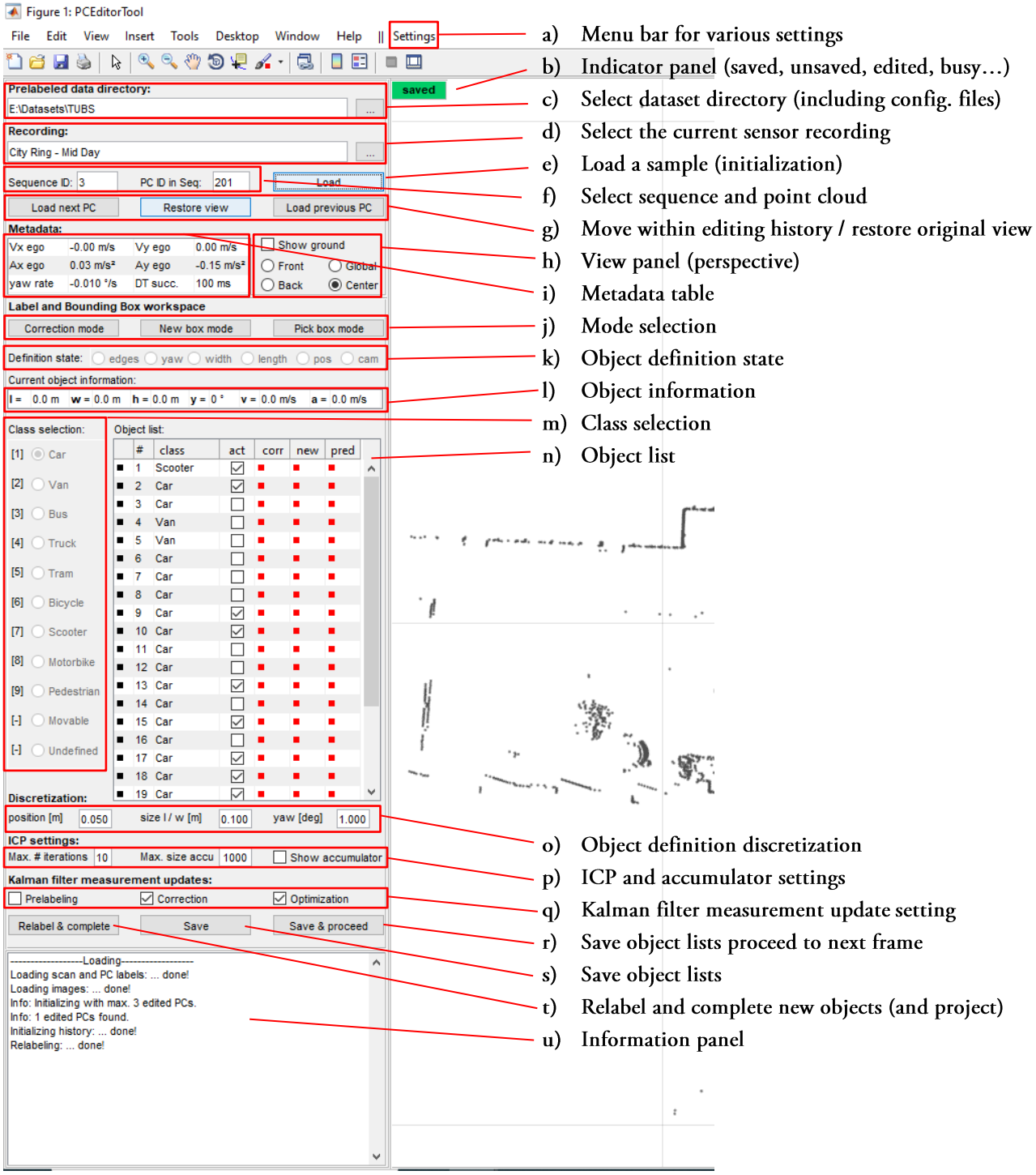
Important notices:

- Whenever you are clicking into the point cloud, e.g. for object definition or picking objects, make sure you hit a free white space within the axes next to the desired points or objects. Elsewise the respective callback will not react.

- Note the list of bugs in Table A.1.

## 2.2 GUI elements

Fig. 2.1 shows the GUI elements and provides brief description. The elements are explained in detail in Table 2.2.



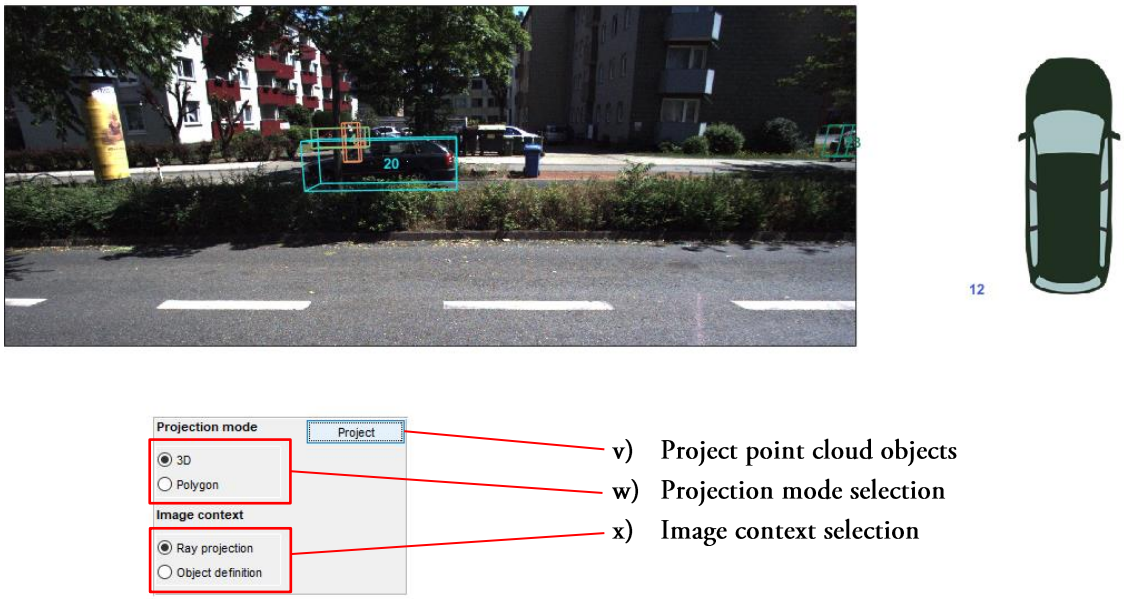



**Fig. 2.1** The editor's GUI elements

**Table 2.2** Detailed description of the GUI elements

| | Description |
|---|---|
| a) | Menu bar settings:<br>• General<br>– *Predict*: Activate the prediction of object lists. Elsewise object lists from the prelabeling stage are loaded.<br>– *Auto import*: Import new objects automatically. Recommended.<br>– *Auto delete*: Delete objects automatically (exceeding the scanner's range or sparsely populated boxes for a configurable number of consecutive frames)<br>– *Debug information*: Display debug information<br>– *Display object data*: Show dynamics and variances when zooming to an object<br>• Optimization<br>– *Enable ICP*: Enables the registration of object point clouds<br>– *Enable fitting*: Enable fitting algorithms for measurement updates<br>– *Enable edges estimation*: Enable the rectification algorithm used for object definition as fallback<br>– *Enable debug plots*: Enable plotting of the optimization algorithms<br><br>Note that more settings are available in the *Default settings* section in PCEditorToolGUI.m (explained inline). |

| | |
|---|---|
| b) | Indicator panel: Shows the current editor state, e.g. if the point cloud is saved. When moving through the editing history (load next and load previous frames) and defining new objects in previous frames, those objects are predicted through the editing history into the current frame (see section 3.4). This state is indicated by “delta”. |
| c) | Dataset directory: Select the root directory providing the recordings and calibrations files. See \datasets\TUBS for an example. If loaded successfully, the class definition set is loaded and displayed according to the EditorConfig.xml file. |
| d) | Recording selection. Select the recording to be edited. Sequence and point cloud IDs will be set automatically according to the editing history (EditorConfig.xml) or the first sample within this recording. |
| e) | Load: Load a sample (specified by its sequence and point cloud ID) and initialize the editor (e.g. setting up the Kalman filters). |
| f) | Select the sequence and the point cloud ID (PCID). |
| g) | Load previous or next samples. If those samples were edited in the current session, the editing history is loaded. |
| h) | View panel: Change camera perspective. |
| i) | Current frames metadata (in parts): Ego velocity, acceleration, yaw rate and time difference to successor frame. |
| j) | Mode selection. Correction mode: The editor’s main functionality (see section 3.2). New box mode: Define new objects. Pick box mode: Select a specific object. Mind clicking the axes *next* to the object, not the points or the bounding box. |
| k) | Object definition state. Switch pressing [q] or [e]. *Edges*: Only during object definition. *Width*, *length* and *yaw*: Change value using your scrolling wheel. *Pos*: position of the bounding box. Click onto the axes to change the bounding box’s middle to that point. *Cam*: Change the camera’s angle of view by pressing and holding the left mouse button (rotation). You can translate the camera by pressing and holding the right mouse button (same as for the whole point cloud axes). |
| l) | Information regarding the currently selected object: Length, width, yaw, velocity (absolute) and acceleration (absolute) |
| m) | Class selection as defined by the EditorConfig.xml file (see section 2.3). |
| n) | Object list. List of all point cloud objects. *Act*: Active flag indicator (switch pressing [f]). Active objects are tracked and optimized. The prediction of passive objects is simply done by compensating the vehicle’s ego movement. *Corr*: Green if vehicle was manually corrected. *New*: Red if object is new in the current frame. *Pred*: Prediction quality indicator. If the object’s prediction is uncertain, this indicator becomes red. The editor’s correction mode will automatically jump to those objects. |
| o) | Object definition discretization. Change the increment for the length, width or yaw definition (by the scrolling wheel). |

| | |
|---|---|
| p) | ICP and accumulator settings. Change the maximum number of ICP iterations (impact runtime) and the maximum size of the accumulator. The accumulator collects registered points frame by frame in order to densify an object's point cloud (to improve performance if an object is temporarily occluded). |
| q) | Kalman filter settings. *Prelabeling*: Find corresponding objects in the prelabeling stage's object list to generate a measurement update (recommended setting: off). *Correction*: Use manual corrections as measurement updates. *Optimization*: Apply optimization algorithms to compute a measurement update. |
| r) | Save and proceed button: Save current object lists (point cloud and image labels) and predict objects into the next frame. This button will also trigger: Complete newly defined objects (compute height and initialize Kalman filters), relabeling of the point cloud, the projection of point cloud objects into the image planes and performing the Kalman filter measurement updates. Most important button in combination with the correction mode. |
| s) | Save button: Same as r) but without proceeding to the next frame. |
| t) | Relabel and complete: Same as s) but without saving the object lists. |
| u) | Information panel. The editor gives information about hot key assignments and possible user interactions in every state. Moreover, this panel displays information regarding the current processing step (e.g. warnings or errors). |
| v) | Project point cloud objects into the image planes. This is typically done automatically by the editor. Pressing this button will delete all existing (and possibly edited) image labels and reproject the point cloud objects. |
| w) | Projection mode selection. Choose between projecting 3D shapes or polygons. Changing the projection mode and triggering the projection using v) will delete all existing labels before reprojecting them using the respective mode. |
| x) | Change image context. *Ray projection*: Left clicks will project rays of sight into the point cloud as visual support during editing (e.g. to identify objects in the point cloud). Right clicks will delete those rays in reverse order. *Object definition*: A right click will open a context menu to add user defined shapes (currently license plates or faces). You can add more classes here. |

## 2.3 Editor configuration

In order to change the set of classes, open the editor's configuration file (EditofConfig.xml in \datasets\TUBS) and add a new class definition (name, label and display color) as shown in Fig. 2.2. Choose an unassigned ID to ensure compatibility with the TUBS dataset. Changes made to the configuration file will be applied upon the next startup of the editor. Currently, only classes in section <MovableClasses> will be taken into account. You may want to change the `fVertDist` value (space between classes within the GUI) in \gui\generateClassSelection if you consider more classes then currently configured.

```
</Class>
<Class>
   <Name>Tram</Name>
   <LabelID>22</LabelID>
   <Color_RGB>[167 244 66]</Color_RGB>
</Class>
<Class>
   <Name>Bicycle</Name>
   <LabelID>11</LabelID>
   <Color_RGB>[255 165 0]</Color_RGB>
</Class>
<Class>
   <Name>Scooter</Name>
   <LabelID>21</LabelID>
   <Color_RGB>[128 29 249]</Color_RGB>
</Class>
<Class>
   <Name>Motorbike</Name>
   <LabelID>10</LabelID>
   <Color_RGB>[238 130 238]</Color_RGB>
</Class>
<Class>
   <Name>Pedestrian</Name>
   <LabelID>4</LabelID>
   <Color_RGB>[77 77 204]</Color_RGB>
</Class>
```

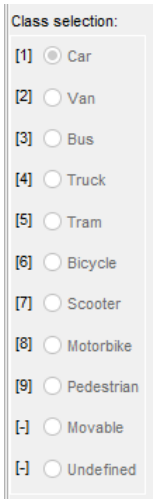



**Fig. 2.2** Class set configuration using the editor's configuration file

## 2.4 Camera handling

You can change the camera's perspective at any time. Change the rotation by clicking and holding an arbitrary point within the axes. Clicking and holding the right mouse button will move the camera in a horizontal plane. Use the mouse wheel to zoom in and out the point cloud. You can restore the viewing angle at any time by pressing "Restore view".

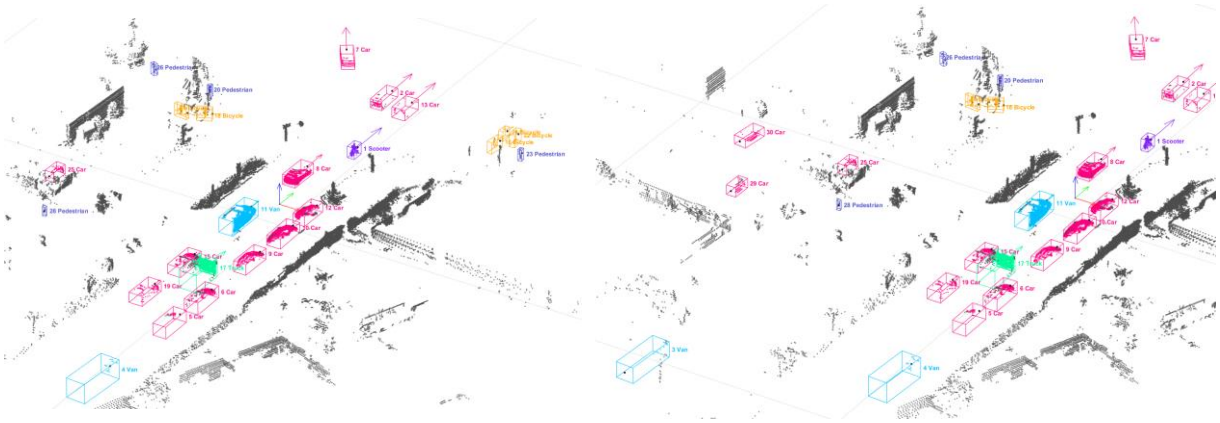


**Fig. 2.3** Camera handling. Left: Changed perspective using left mouse button. Right: Horizontal movement.

You can change the camera perspective when correcting or defining an object by setting the definition state to "cam". The camera is then handled in the same way as stated above.

# 3 Point cloud labeling

## 3.1 Object definition

In order to define a new object, press “New box mode”. You are then asked to click a point on the point cloud axes (click the white space and not a point; the callback will not react otherwise) in order to zoom to that specific point. Now, define four corner points of the bounding box clock or counter clockwise and a fifth point that indicates the front edge (bounding box orientation). Fig. 3.1 shows an example.

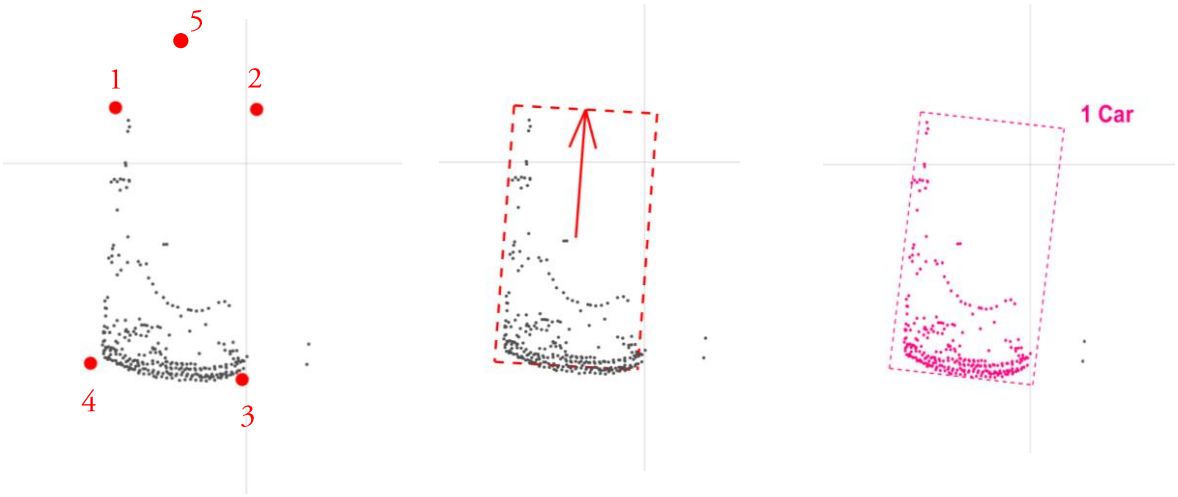



**Fig. 3.1** Object definition example

A rough estimate of the corner points is sufficient. The editor will rectify the bounding box and change into the object definition state. If you would like to change the camera perspective while defining the edge points, change the definition state to “cam” by selecting the respective radio button. Press [q] or [w] to change between the definition states and use the scrolling wheel to adjust width, height or the bounding box yaw. You can use [w], [a], [s], [d] to move the object’s position horizontally. For greater distances, you can change to “pos” and change the position by clicking a point within the point cloud axes. Select the object’s class by either using the class selection GUI or pressing the respective hot key [1-9].

Press [f] to mark the object as active (tracked using Kalman filtering and subject to the optimization algorithms). Parking cars or motorbikes as well as sitting pedestrians are passive objects. Active objects cause a higher runtime when proceeding to the next frame.

The object is still a flat rectangle. We recommend defining all new objects in a first step. Afterwards, you can press “Relabel and complete” in order to calculate the object’s height and initialize its Kalman filter. The point cloud will be relabeled accordingly. In order to delete an object, press [del] twice.

## 3.2 Correction mode

The correction mode is the editor's main functionality. Objects that need editing after the prediction and optimization step will be selected automatically. Enter this mode by pressing "Correction mode". By pressing [t], the editor proceeds to the next object that needs manual review (marked with a red predicted flag - *pred* - in the object list GUI element). You can use the up and down arrow on your keyboard to move through the object list manually.

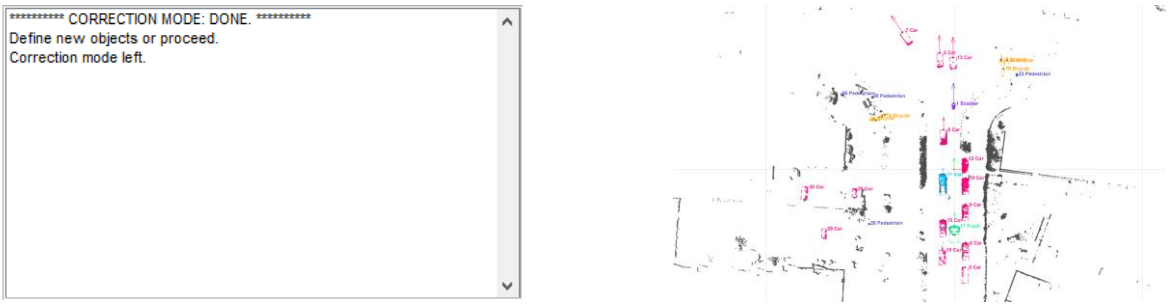



**Fig. 3.2** Exiting the correction mode

When all uncertain objects are corrected, the editor exists the correction mode automatically and restores the viewing angle as selected in the view panel. Any object will be marked as uncertainly predicted in their first predicted frame. In specific intervals, all objects have to be corrected, regardless their prediction flag. You can specify that interval by `nGlobalCorrInterv` in the *Default settings* section in PCEditorToolGUI.m.

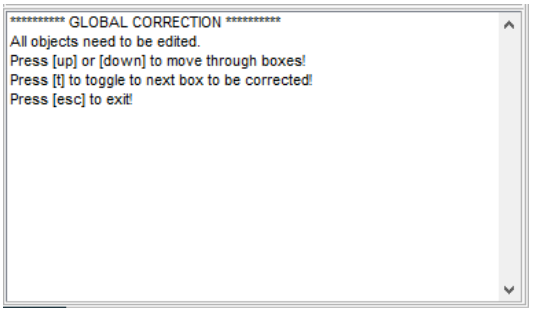



**Fig. 3.3** Global correction event

## 3.3 Pick box mode

You can manually select and define specific objects by pressing "Pick box mode" and clicking a white space within the point cloud axes next to the desired object. Pressing [t] in this mode will switch to the next closest object. You can change the object as explained in section 3.1.

## 3.4 Delta prediction

It happens to miss objects during the editing process. Whenever realizing that a newly found object must have been existed for a couple of frames already, you can easily go back in the editing history to the frame that the object first occurred. Define this new object and press "Save and proceed" (or "Save" and "Load next") to predict only the missing objects through the editing history. This delta prediction is indicated by the information panel and the indicator panel as shown in Fig 3.4.

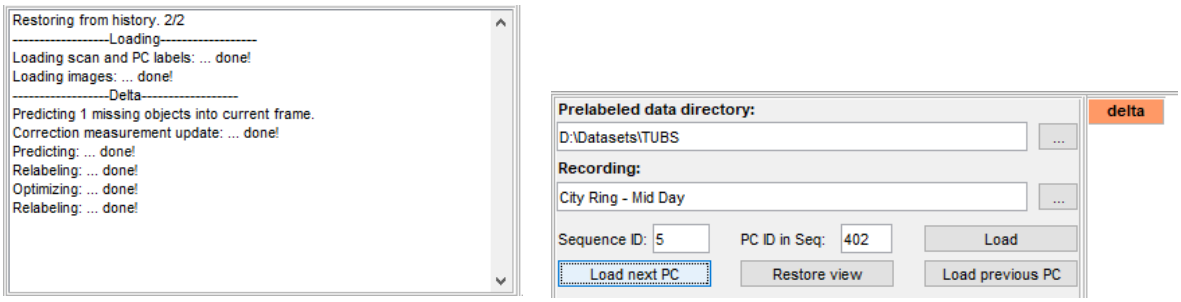



Fig. 3.4 Delta prediction

## 3.5 Visual support

You can project rays of sight into the point cloud axes to identify point cloud segments more easily. Rays are projected by clicking the respective pixel in the image axes. Fig. 3.5 shows an example.

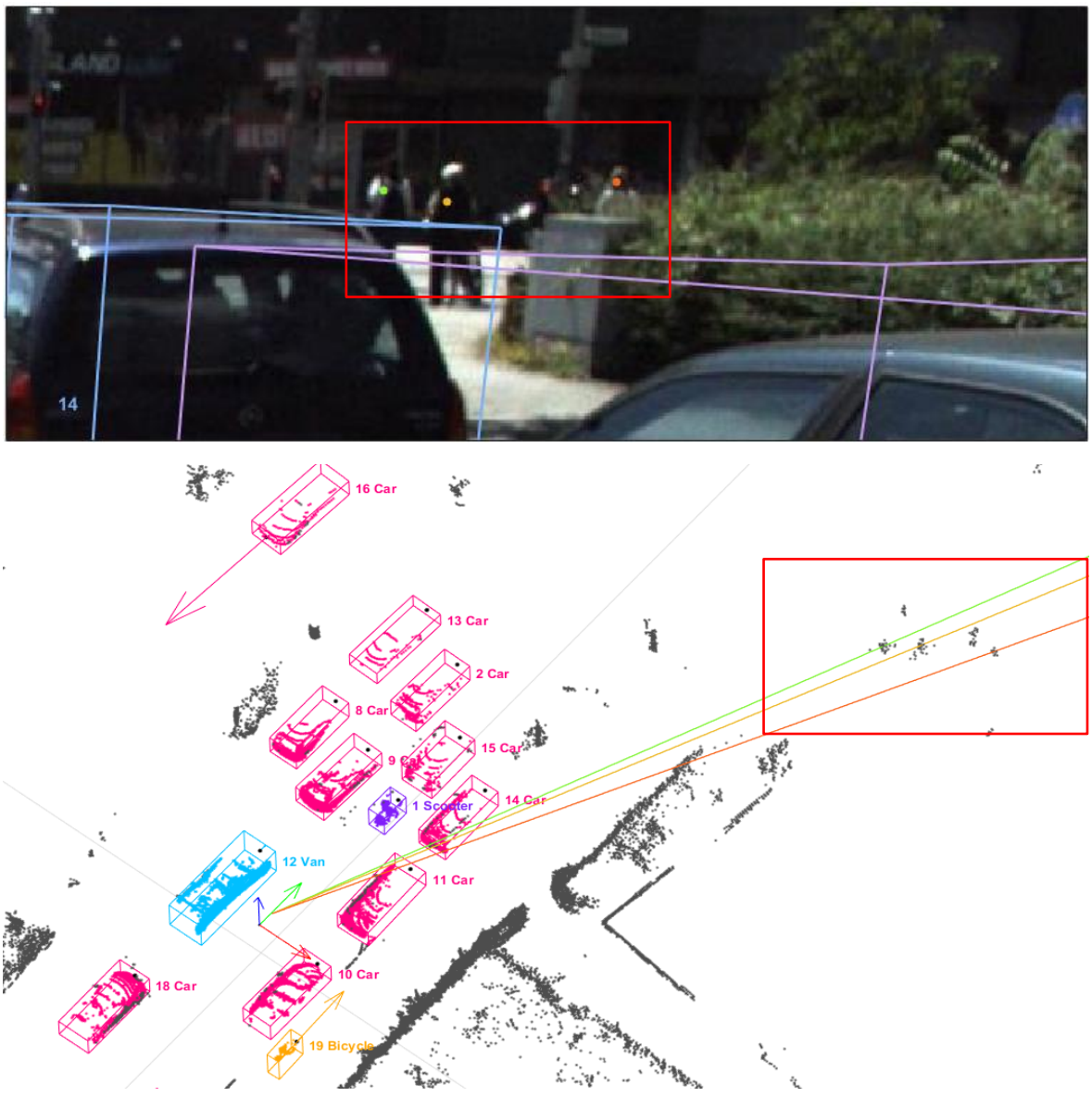



Fig. 3.5 Visual support by projecting rays of sight

# 4 Image labeling

## 4.1 3D projection

Point cloud objects are projected into the image planes automatically. You can trigger the projection manually by pressing “Project” in the image panel. This will delete all labels (possibly edited). Fig. 4.1 shows an example. Our tool supports up to four cameras.

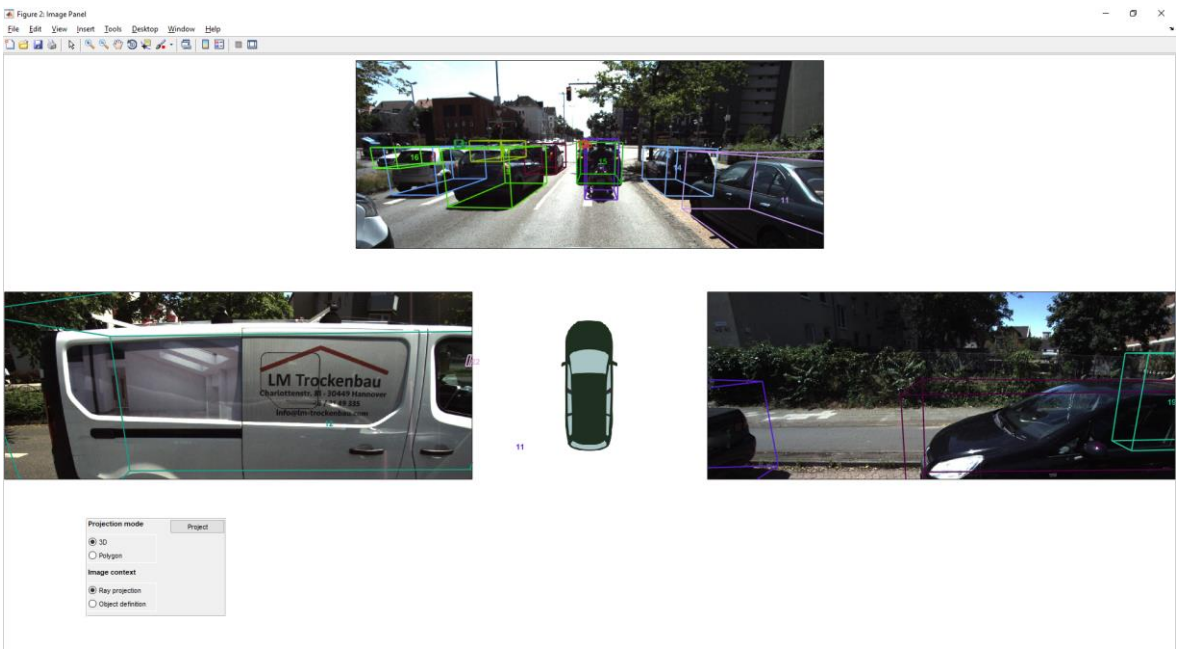



**Fig. 4.1** Result of the 3D bounding box projection

The calibration between laser scanner and cameras is accurate enough to rely on the automatic labeling. You can still move projected 3D bounding boxes using drag and drop to correct the bounding box position further. This becomes important for the right sided images. Due to the incremental working principle of the rotating Velodyne laser scanner, an imperfect synchronization between image and laser scan occurs at the scan’s breaking point at 0° (see Fig. 4.2). Grey circles represent example measurements and brighter colors indicate a lower time difference towards the scan’s end.

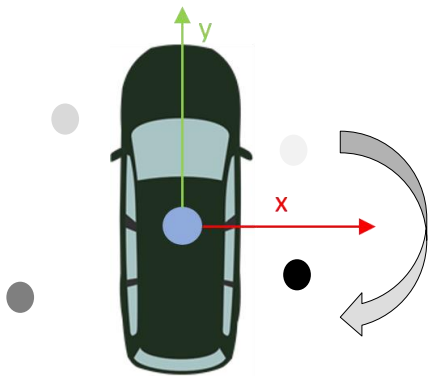



**Fig. 4.1** The Velodyne’s reference frame and rotation direction.

The images of the rightward camera are triggered just below the x-axis. Hence, a time difference between point cloud measurements and image of approx. 100 ms occurs. This leads to an offset between laser scan object and the respective projected bounding box as shown in Fig. 4.2.

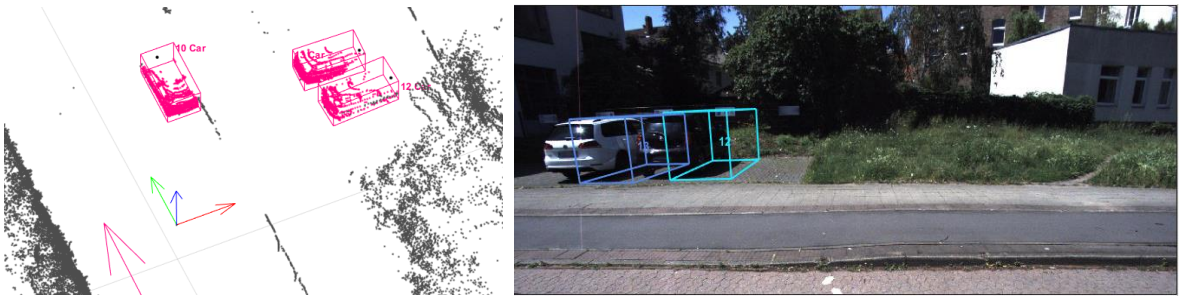



**Fig. 4.2** Synchronization issue regarding the rightward camera at the scan's breaking point

In this case, you can manually correct the bounding box's position using drag and drop (callback reacts only on the edge lines or corners) as shown in Fig 4.3. The issue does not persist for the other half of the rightward camera.

**Fig. 4.3** Left: Manually corrected projection. Right: Automatic projection after the scan's breaking point.

## 4.2 Coarse segmentation

By changing the projection mode on the image panel (see Table 2.2), you can project polygons in order to obtain a coarse segmentation as shown in Fig 4.4.

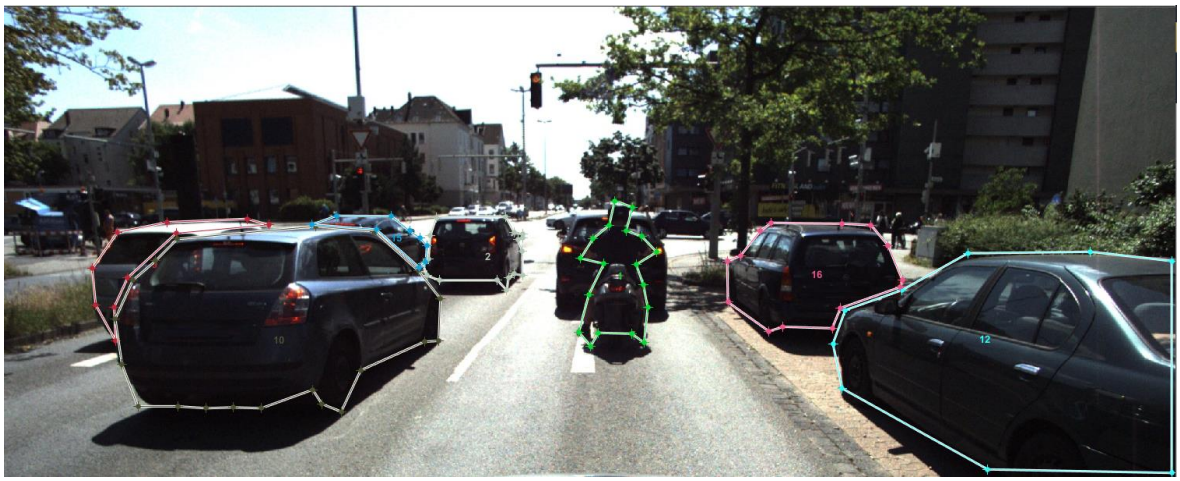


**Fig. 4.4** Projecting polygons to obtain a coarse segmentation

Polygons need to be edited manually. Therefore, press and hold [a] after projecting the point cloud objects as polygons. Click the respective polygon to add vertexes or use drag and drop to change position. Polygons are tracked in the same way as 3D projections using LiDAR correspondences. Figure 4.5 shows an example (bottom picture: two frames after the top picture).

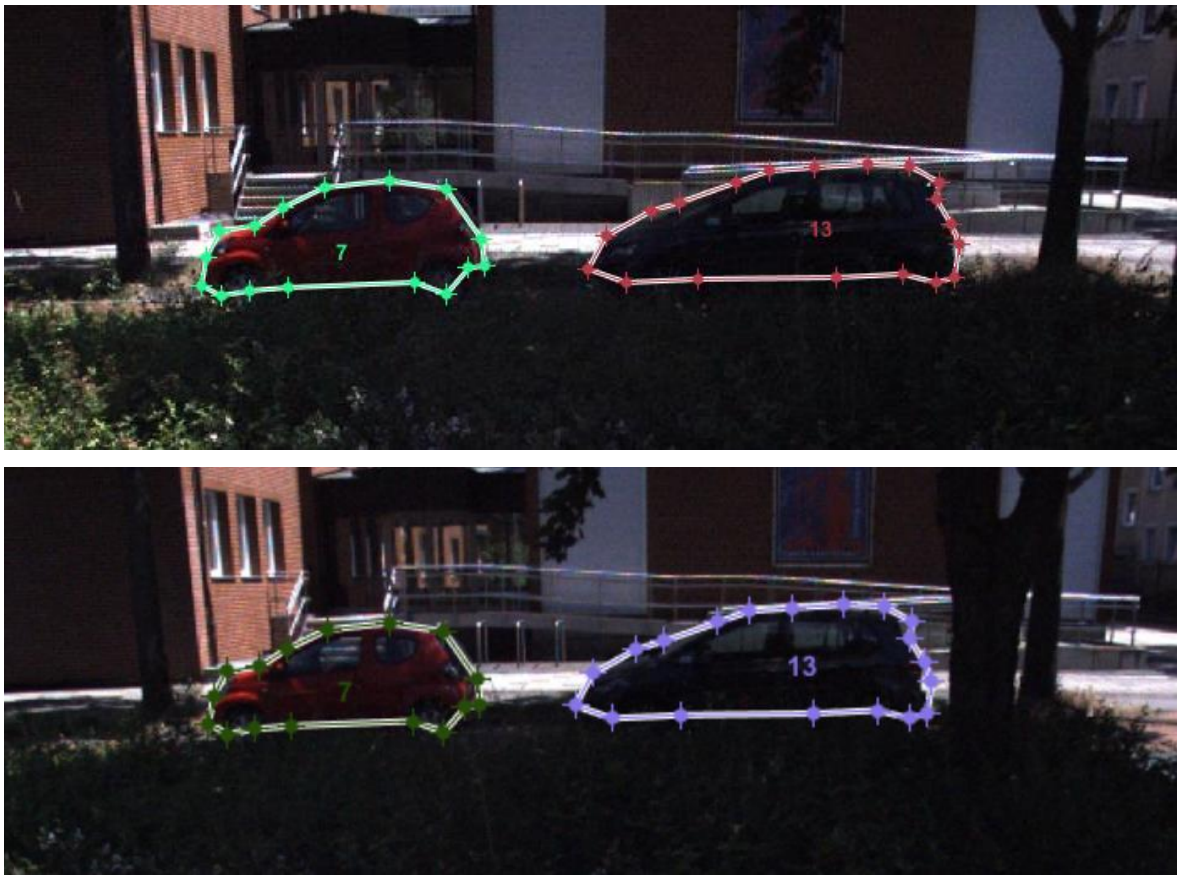



**Fig. 4.5** Tracked polygons using LiDAR correspondences

## 4.3 Adding user defined shapes

You can add more image labels by changing the image context to object definition (see Table 2.2). A right click will open a context menu in order to select a label to add. If a LiDAR correspondence is set for a user label, it will be predicted into consecutive frames. To test this functionality, `bDisplayPD` in the *Default settings* sections of PCEditorToolGUI.m must be true. Fig. 4.7 shows an example.

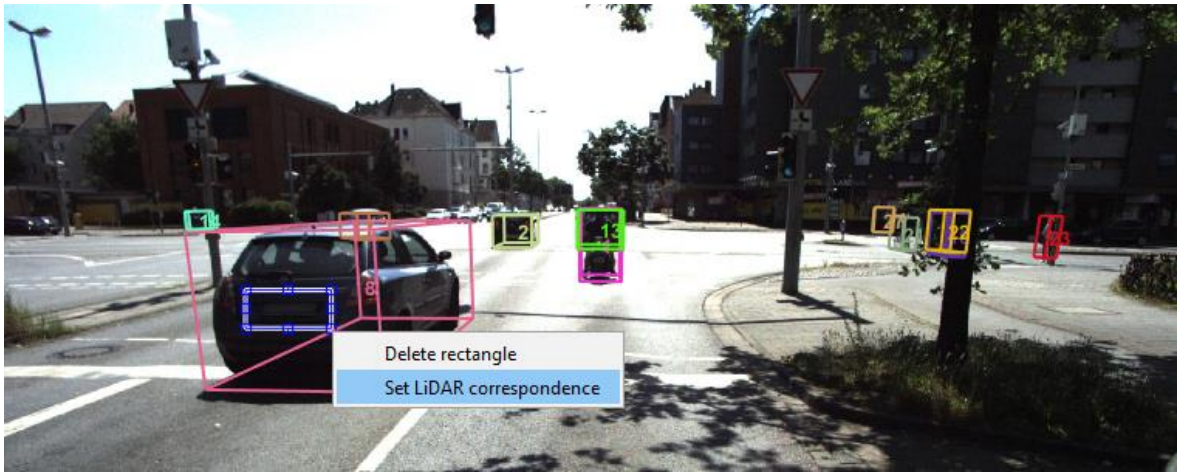



**Fig. 4.6** Defining additional labels and setting the LiDAR correspondence

# 5 Calibration

The editor provides an integrated calibration mode. Point correspondences between laser scan and image are generated by placing calibration targets into the scan and the respective image. Targets must to be easy to find within the laser scan (e.g. big rectangular shaped boxes, cars or tables as in our case).

The editor uses its object definition functions in order to define the point cloud targets. For calibration, all defined objects need to be orientated in the same direction as shown in Fig. 5.1 (here: towards left side of the image). The bounding box side facing towards the camera is used for generating the point correspondences.

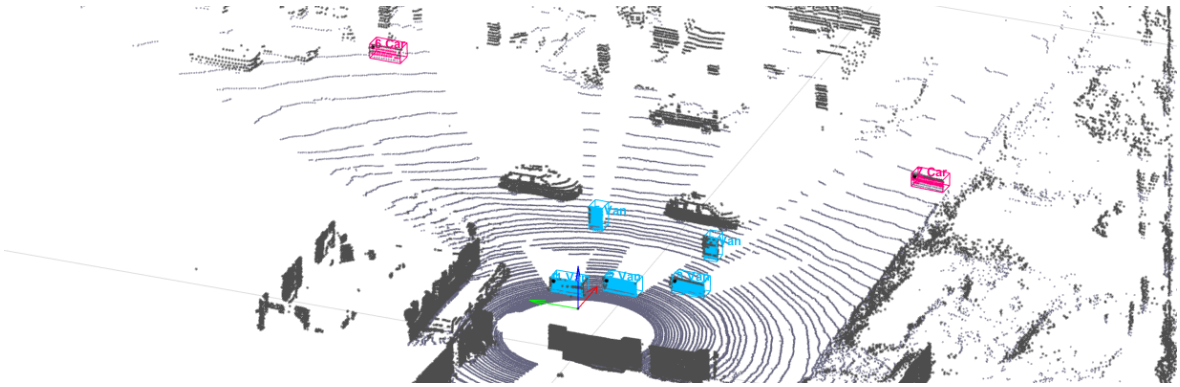


**Fig. 5.1** Defined calibration targets in the laser scan

In the following, the editor's calibration mode must be active and the number of calibration targets must be specified (`bCameraCalibrationMode` and `nNumTargets` in the *General settings* section, respectively). The corresponding image points are then manually defined in the respective image (see Fig. 5.2). The points must be defined in a specific order: top left, bottom left, bottom right and top right. The definition of image objects must follow the same order as in the point cloud.

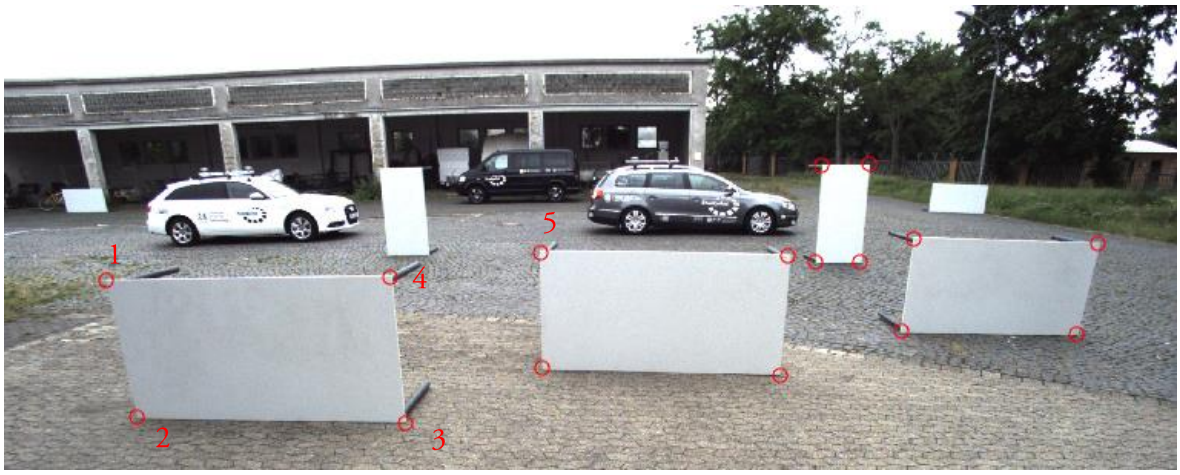



**Fig. 5.2** Defined point correspondences in the image plane

After the definition of all correspondences, a calibration container (Calibration_Container.mat) is saved to the workspace. You can now start the script calib.m (in \calibration) to start the calibration process. We provide an example container. When calibrating your own sensors, mind to unset the `bDemoMode` flag in calibrateCamera.m. The calibration function needs an initial guess for the calibration parameters (also stored within the calibration container, previously read from CalibrationParameters.txt. We implemented a stochastic optimization based on averaging parameter sets that minimize the projection error. Fig. 5.3 shows the first optimization step. Points marked with a cross are projections according to the current (best) parameter set.

**Fig. 5.3** First optimization step and temporary projected LiDAR points

After the specified number of optimization steps, the calibration results are stored as CalibrationResults.mat and a convergence plot (Fig. 5.4) is generated that shows the absolute difference compared to the initial guess. We found that best practise is to run the calibration twice. Once with only targets that are close to the camera (see Fig. 5.3) and another time using all targets.

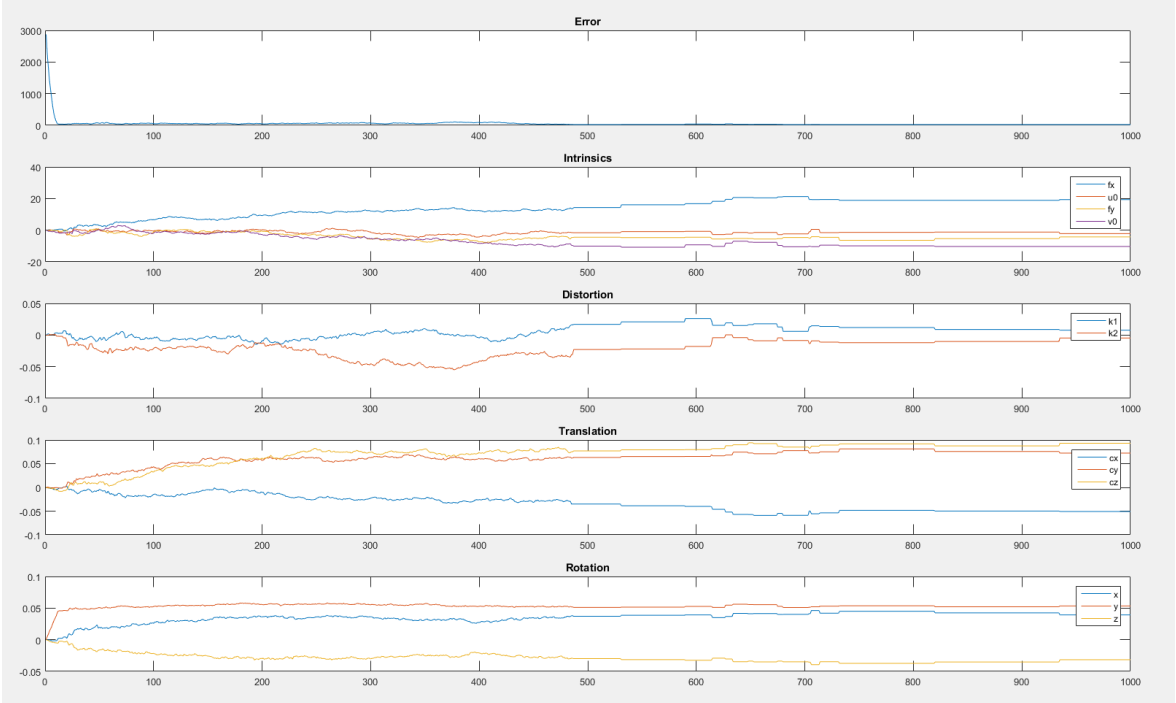



**Fig. 5.4** The calibration's convergence plots

# 6 Typical workflow

In this section, we describe a typical labeling session. For this example, we consider the 3D projection functionality. First, load a sample (e.g. taken from City Ring - Mid Day; we manually edited the first 4 sequences already) by selecting the dataset and recording directory as well as setting the sequence and PCID (editor will jump to sequence 5 and PCID 401 automatically if City Ring - Mid Day is edited).

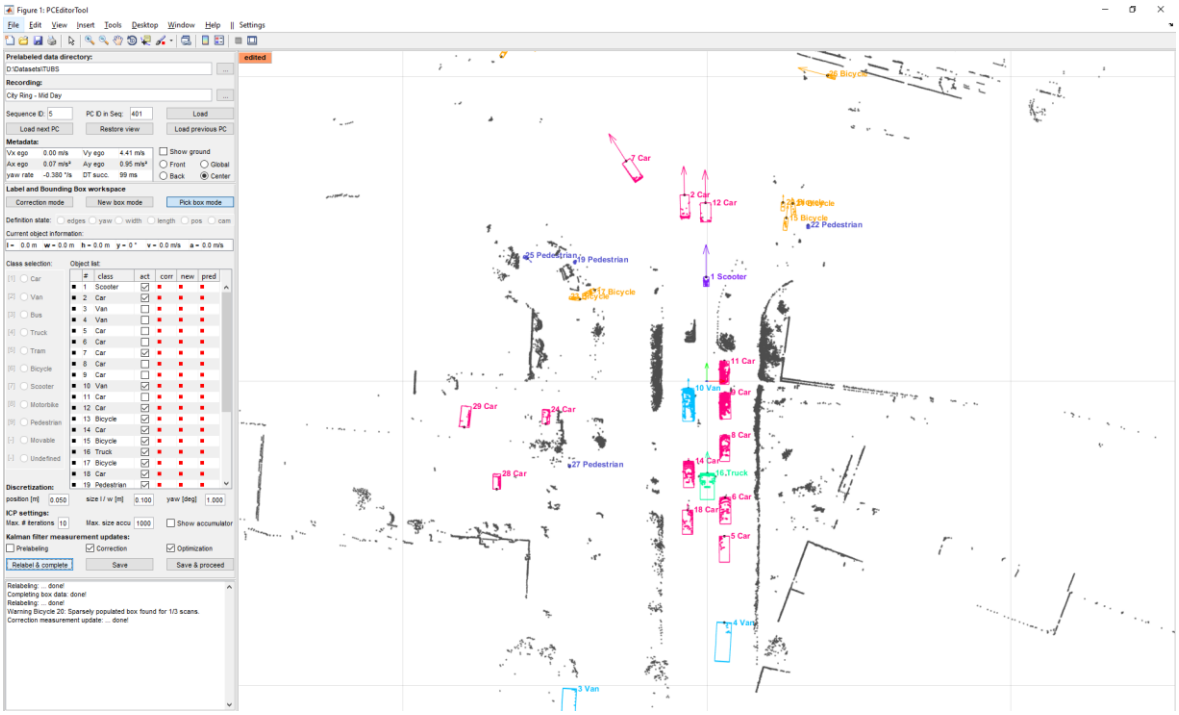


**Fig. 6.1** Sample loaded from the TUBS - City Ring - Mid Day recording

Now, add missing objects by using “New box mode”. For this example, we previously deleted an object close to the ego vehicle. Set the vehicle to “active” by pressing [f]. The result is shown in Fig 6.2.

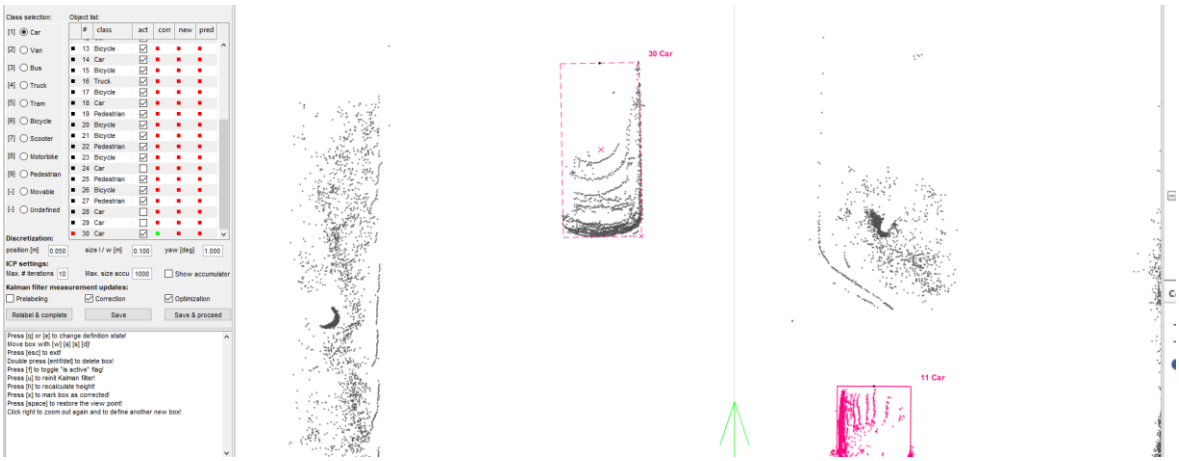


**Fig. 6.2** Add missing objects to the point cloud

After all objects are defined press “relabel and complete” to calculate the objects heights and to relabel the point cloud accordingly. Check the projection result in the image panel.

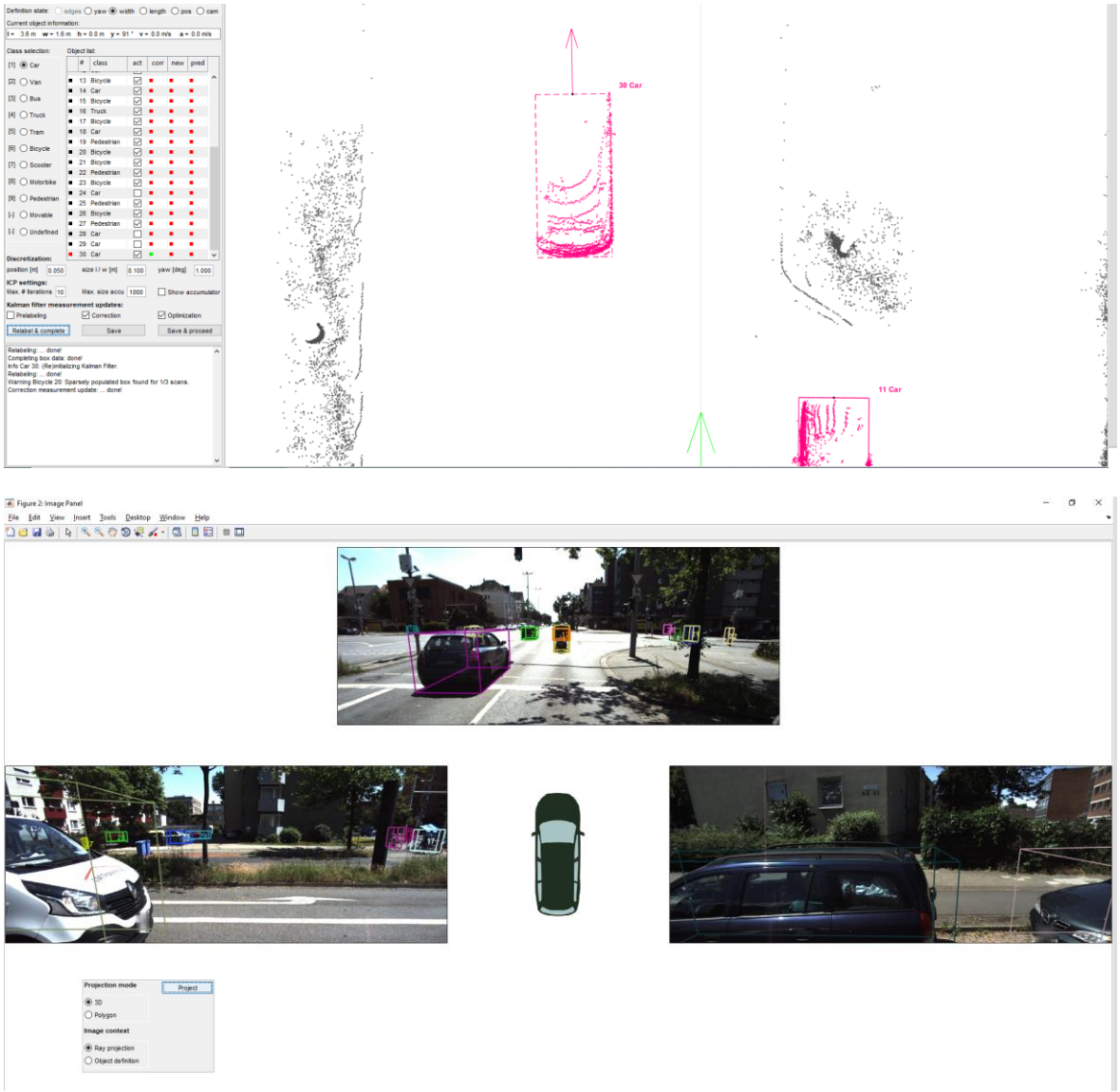



**Fig. 6.3** Point cloud and image panel after completing newly defined objects

Now press “Save & proceed” to save the current object lists and predict them into the next frame. Depending on the recording you are editing, you might be asked to confirm the saving process because the frame has been saved previously. In the new frame, press “Correction mode”. Toggle through objects that need to be corrected using [t] and correct the objects according to section 3.1. Because this is the first prediction, you will be asked to correct all objects. If an object needs no correction, you can press [x] to mark the object as corrected. If you are unsure about the object’s position, keep it in place and trust the editor’s prediction.

We have labeled City Ring - Mid Day regarding the following prerequisites:

- Objects can only grow in size. In case of occlusion in subsequent frames, the object keeps it size.
- We omit vehicle’s mirrors for labeling and align the bounding boxes regarding the dominant lines.

If the height of an object is calculated wrongly, you can enforce a recalculation pressing [h] (applied at the next completion process, e.g. when relabeling or saving the point cloud). If the dynamics calculation is faulty, you can reinitialize the Kalman filter by pressing [u]. Whenever you have changed the camera perspective during editing, you can press [space] to restore the original perspective.

If the setting "Auto delete" is active (recommended), the editor takes care of the deletion process. If a box is empty for a specific number of frames (`nDeleteDelay`) or exceeds a specific range (`fMaxRange`), the object will be deleted automatically. Also, if the setting "Auto import" is active (recommended), the editor imports new objects from the prelabeling stage automatically. This is especially helpful for small moving point clouds as pedestrians. Currently, our prelabeling stage is not able to detect not moving objects.

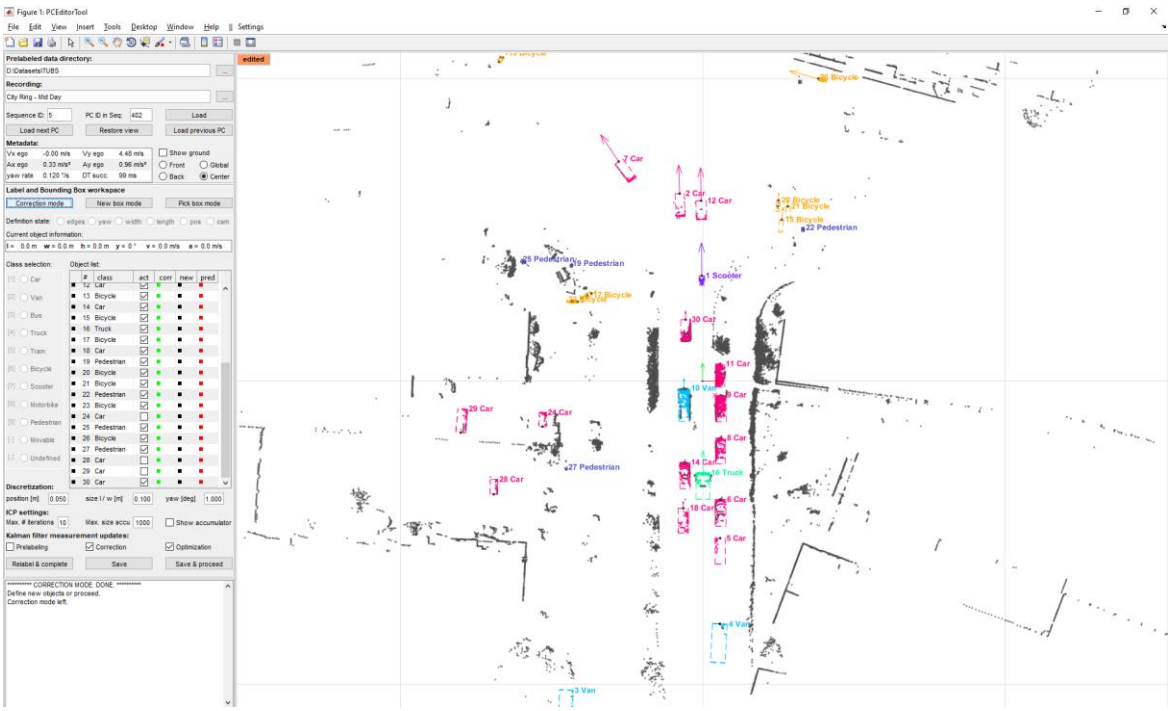


**Fig. 6.4** Editor GUI after correcting all objects

You can now press "Save & proceed" and enter the correction mode for the following frame. On pressing [t], the editor now selects objects to be corrected automatically (red predicted flag in the object list). Therefore, passive objects are skipped and active objects are evaluated based on the differences between prediction and optimization estimations. Fig. 6.5 shows the respective object list. In the next frame, a global correction event will enforce all objects to be corrected.

An editing session mostly consists of "Save & proceed", "Now box mode" and "Correction mode". We hope you will find the editor helpful in your research.

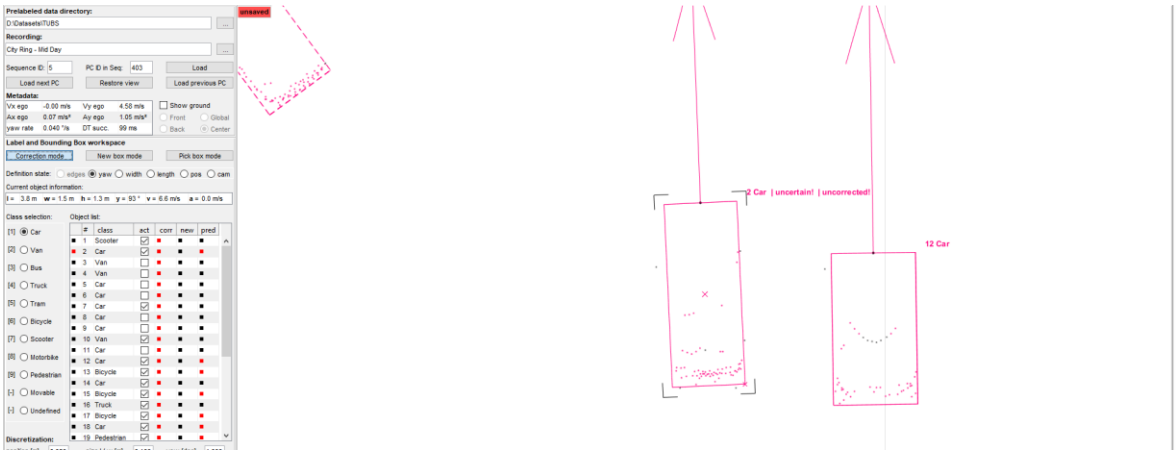



Fig. 6.5 Automatic selection of objects to be corrected

## Attachments

[1] https://dataset.ifr.ing.tu-bs.de/tubs-dataset/download-01.php

[2] https://dataset.ifr.ing.tu-bs.de/tubs-dataset/tooling.html

[3] https://github.com/TUBSDataset/TUBS_LabelingTool

[4] C. Plachetka, J. Rieken, and M. Maurer, "The TUBS Road User Dataset: A New LiDAR Dataset and its Application to CNN-based Road User Classification for Automated Vehicles", in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Maui, Hawaii, USA: IEEE, 2018.

**Table A.1** Bug list

| ID | Module | Description |
|---|---|---|
| 1 | I/O | License plate and face annotations are overwritten when the setting "Init from edited" is not active. |
| 2 | 3D projection | If objects are projected that are not fully covered in the image plane, falsely projected points may occur. Typically, those invalid points are indicated by negative values. However, we observed the case that invalid points (e.g. when a 3D object is behind the camera) were projected to valid positions within the image. We intercept those cases by testing bottom and top lines for intersections (see images/projectPCObjects). This leads to the rejection of objects that have mixed valid and invalid points in an image. |