

A- Erste Simulation durchführen:

- Bevor jeglicher Ausführung, eine Kopie der zu durchführende Info-Datei manuell erstellen und um „_ov_“ in der Benennung am Ende erweitern.

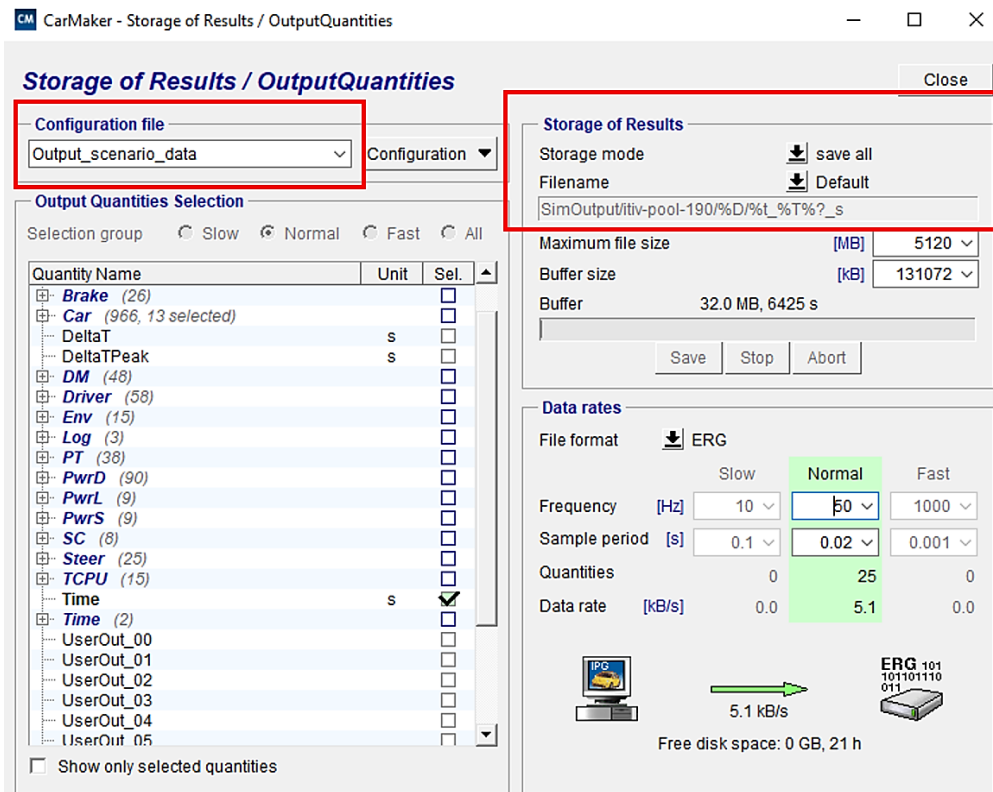
Run

reigeben Ansicht

USB DISK (E:) > Workflow-Beispiel > CM-Umgebung > Data > TestRun

Name	Änderungsdatum	Typ	Größe
Ground_Truth_label_Highwat_01	15.08.2019 21:35	Datei	224 KB
Ground_Truth_label_Highwat_01_ov	21.08.2019 19:06	Datei	224 KB
High_Way_TestRun_Sequential	15.08.2019 19:09	TS-Datei	4 KB
Highway_with_data_export	21.08.2019 01:47	Datei	19 KB
Highway_with_data_export_ov	21.08.2019 01:47	Datei	11 KB
prepdata	21.08.2019 19:06	Microsoft Access ...	533 KB

- CM öffnen und Konfiguration der Datenaufzeichnung- und Speicherung anpassen:
 - o Storage Modus auf **save all** setzen
 - o Pfad der ERG-Datei anpassen hier auf **Default** setzen
 - o Konfigurationsdatei „**Output_scenario_data**“ auswählen (kann auch benutzerdefiniert angepasst werden)



- MATLAB- Skripte **main.m** und **overwrite_infofile.m** öffnen und die Pfade hier wie folgt anpassen:

```

Editor - F:\Workflow-Beispiel\CM-Umgebung\src_cm4sl\main.m
main.m  overwrite_infofile.m  Ground_Truth_label_Highwat_01  Ground_Truth_label_Highwat_01_ov  create_tensor.m  +

1  %%
2  % main script to run : edited to process erg files from Testrun series with variations!
3  % make the necessary changes, e.g path, directory ...
4  % the simulations' output erg files are now saved in the default directory
5  % the main outputs of the main script are: a struct variable containing the
6  % tensors matrices of ego and TObjs for all Testruns
7  %% preprocess
8  - clc
9  - close all
10 - clear all
11
12 %% add necessary directories to search path
13 - dir_TR = 'F:\Workflow-Beispiel\CM-Umgebung\Data\TestRun'; % dir containing Testrun Infofiles
14
15 - maindir = 'F:\Workflow-Beispiel\CM-Umgebung\SimOutput'; % dir containing the erg files
16
17 - date_str= datestr(now,'yyyymmdd');
18 - dir2ergfile =strcat('F:\Workflow-Beispiel\CM-Umgebung\SimOutput\itiv-3041\',date_str);% dir containing the erg files
19
20 - dire2cmenv = 'F:\Workflow-Beispiel\CM-Umgebung\src_cm4sl'; % dir containing the scripts files and cmenv()
21 - addpath(dir2ergfile,maindir, dire2cmenv,dir_TR);
22

```

```

main.m  overwrite_infofile.m  Ground_Truth_label_Highwat_01  Ground_Truth_label_Highwat_01_ov  create_tensor.m
1  % this function overwrites the CM TestRun infofile with the infos contained
2  % in the tensor created within the function create_tensor.m
3
4  function overwrite_infofile(S,Filename)
5
6  - cd ('F:\Workflow-Beispiel\CM-Umgebung\Data\TestRun');
7
8  % create instance of the file
9  - handle=fopen('');
10
11 - iffile_read(handle,Filename);
12
13 % overwrite the man of ego
14 - if isfield(S,'Ego')
15
16 % initial lat offset & velocity + maneuver number
17 - ifile_setstr(handle,strcat('DrivMan.Init.Velocity'),num2str(S.Ego(2,1)));
18 - ifile_setstr(handle,strcat('DrivMan.Init.LaneOffset'), num2str(S.Ego(6,1)));
19 - ifile_setstr(handle,strcat('DrivMan.nDMan'),num2str(size(S.Ego,2)));
20
21 - nD_man=eval(ifile_getstr(handle,'DrivMan.nDMan'));
22
23 % ignore traffic ( for ego)
24 - ifile_setstr(handle,strcat('Driver.ConsiderTraffic'),num2str(0));
25
26 % if necessary delete keyvalue of distlimit or Timelimit
27 - for n=1:nD_man
28
29 - if ~strcmp(ifile_getstr(handle,strcat('DrivMan.',num2str(n-1),'.DistLimit')), '')
30
31 - ifile_setstr(handle,strcat('DrivMan.',num2str(n-1),'.DistLimit'),'');
32 - end
33

```

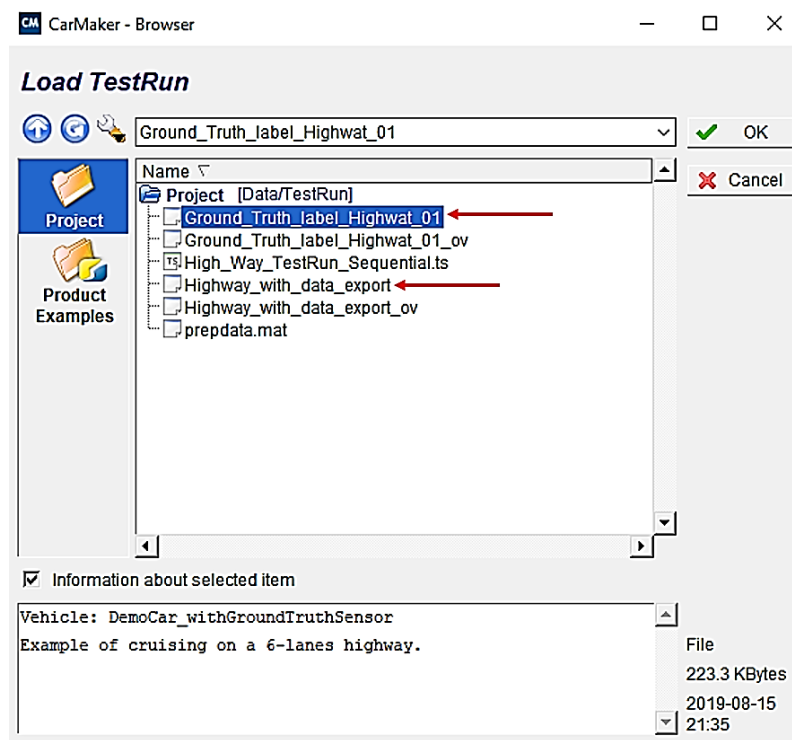
- Name der zu überschreibende „ov“-Kopie der Info-Datei im **main.m** anpassen:

```

Editor - E:\Workflow-Beispiel\CM-Umgebung\src_cm4s\main.m
main.m x create_tensor.m x create_obj_matrix.m x +
45 - path2ergfile = fullfile(dir2ergfile, TestRun.first_sim(k).name);
46 -
47 - %% create struct containing needed data from CM
48 - prepdata_filename = datapreparation(path2ergfile);
49 -
50 - %% Labeling
51 - label_data = labeling(prepdata_filename);
52 -
53 - %% Create the tensor describing the scenario based on the data from CM and the prediction m
54 - [Tensor_CM, Tensor_model,numOfTObj] = create_tensor(prepdata_filename,label_data);
55 -
56 - %% save output Tensors in the corresponding Testrun struct
57 - TestRun.first_sim(k).Tensor_CM = Tensor_CM;
58 - TestRun.first_sim(k).Tensor_model = Tensor_model;
59 - %% overwrite the infofile for resimulation: create a copy of the testRun Infofile manually.
60 - % file to overwrite = 'Highway with data export ov';
61 - file_to_overwrite = 'Ground_Truth_label_Highwat_01_ov';
62 - overwrite_infofile(Tensor_CM,file_to_overwrite);
63 -
64 - %% delete
65 - delete(prepdata_filename,label_data);
66 -
67 - %% run resimulation via DDE protocol
68 - run_sim_via_DDE(file_to_overwrite);
69 -
70 - %% define the pattern of the erg file from overwritten infofiles and get erg files from the
71 - Files_ov = dir(fullfile(dir2ergfile, '*_ov_*'));
72 - for i=1:numel(Files_ov)
73 -     [~,~,ext] = fileparts(Files_ov(i).name);

```

- Der dir2ergfile-Ordner (enthält erg-Dateien) muss vor der Ausführung der Simulation in Best Case keine Dateien enthalten.
- CM-öffnen und die zu durchführende Info-Datei laden (selbstverständlich die originale Datei und nicht die Kopie mit „_ov_“ Endung).



- Simulation durchführen

B- MATLAB-Code ausführen

Beim Ausführen des MATLAB-Codes über die **main.m** Funktion werden die folgenden Schritte automatisiert durchgeführt:

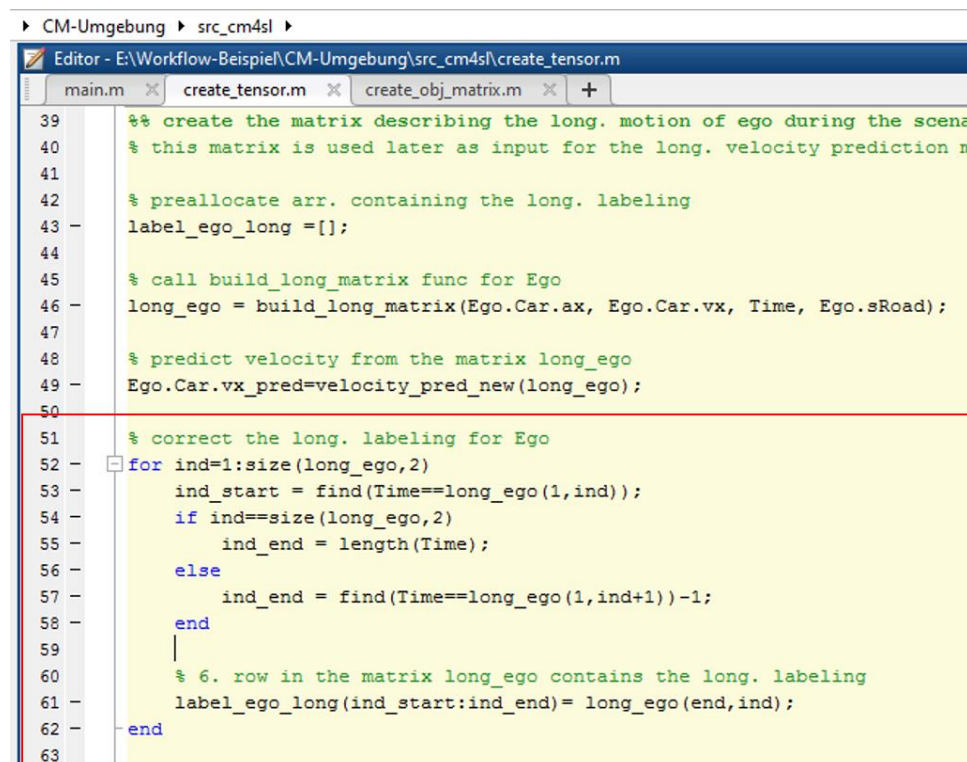
1- CM-Daten importieren und vorverarbeiten

Beim Ausführen des Codes werden die ERG-Dateien aus den durchgeführten Simulationen in MATLAB-Workspace importiert. Für die Vorverarbeitung und die Bereitstellung der Daten kommen die Funktionen **datapreparation.m** und **labeling.m** zum Einsatz.

-**datapreparation.m** dient zur Extraktion und Vorverarbeitung der relevanten Daten.

- **labeling.m** dient zur Codierung der Manöver und der Zustände in Längs- und Querrichtung.

Hinweis: die Codierung der Längsmanöver in **labeling.m** wird innerhalb der Funktion **create_tensor.m** (für Ego) und **create_obj_matrix.m** (für TObjs) korrigiert.



```
CM-Umgebung ▶ src_cm4sl ▶
Editor - E:\Workflow-Beispiel\CM-Umgebung\src_cm4sl\create_tensor.m
main.m create_tensor.m create_obj_matrix.m +
39 %% create the matrix describing the long. motion of ego during the scena
40 % this matrix is used later as input for the long. velocity prediction m
41
42 % preallocate arr. containing the long. labeling
43 - label_ego_long = [];
44
45 % call build_long_matrix func for Ego
46 - long_ego = build_long_matrix(Ego.Car.ax, Ego.Car.vx, Time, Ego.sRoad);
47
48 % predict velocity from the matrix long_ego
49 - Ego.Car.vx_pred=velocity_pred_new(long_ego);
50
51 % correct the long. labeling for Ego
52 - for ind=1:size(long_ego,2)
53 -     ind_start = find(Time==long_ego(1,ind));
54 -     if ind==size(long_ego,2)
55 -         ind_end = length(Time);
56 -     else
57 -         ind_end = find(Time==long_ego(1,ind+1))-1;
58 -     end
59 -     |
60 % 6. row in the matrix long_ego contains the long. labeling
61 - label_ego_long(ind_start:ind_end)= long_ego(end,ind);
62 - end
63
```

```
CM-Umgebung ▶ src_cm4sl ▶
Editor - E:\Workflow-Beispiel\CM-Umgebung\src_cm4sl\create_obj_matrix.m
main.m x create_tensor.m x create_obj_matrix.m x +
150
151 - if flag_resim ==0
152 -     TObj_vx_pred = velocity_pred_new(long_TObj); % predict velocity pro
153 - end
154
155 %% correct the long. labeling for TObj
156 - label_TObj_long = -9*ones(1,length(Time));
157 - for ind=1:size(long_TObj,2)
158 -     if long_TObj(end,ind) ~= -9
159 -         ind_start = find(Time==long_TObj(1,ind));
160 -         if ind==size(long_TObj,2)
161 -             ind_end = length(Time);
162 -         else
163 -             ind_end = find(Time==long_TObj(1,ind+1))-1;
164 -         end
165 -         label_TObj_long(ind_start:ind_end) = long_TObj(end,ind);
166 -     end
167 - end
168
```

2- Tensor der ersten Simulation extrahieren

Durch die Funktion **create_tensor.m** wird der Tensor zur Beschreibung der Szenarios erzeugt. Hier werden die Unterfunktionen, die durch **create_tensor.m** aufgerufen werden aufgelistet und erklärt:

- **build_long_matrix.m**: diese Funktion erzeugt die Matrix zur Beschreibung der Längsmanöver für Ego-Fahrzeug. Jede Spalte dieser Matrix stellt ein Manöver dar.
- **velocity_pred_new.m**: die Ausgabe der Funktion **build_long_matrix.m** wird an diese Funktion übergeben um das Profil der Geschwindigkeit zu schätzen. Diese Funktion ruft die Funktion **Akcelik_model.m** zur Schätzung bestimmter Abschnitt des Profils anhand des Akcelik-Modells. Diese Funktion wird auch im Rahmen der Funktion **create_obj_matrix.m** aufgerufen und die Geschwindigkeitsprofile der Verkehrsobjekte zu schätzen.
- **check_detectable.m**: überprüft die Relevanz eines Verkehrsteilnehmers basierend auf einem Schwellwert. Z.B. wird ein Schwellwert von 2 s gewählt, werden alle Verkehrsteilnehmer, deren Detektion über weniger als 2s beträgt, ignoriert.
- **create_obj_matrix.m**: diese Funktion erzeugt die Matrizen zur Beschreibung der Längs- und Quermanöver sowie die geschätzten Profile der Geschwindigkeit und laterale Position.

- **lat_pos_pred.m**: schätzt das Profil der lateralen Position basierend auf der Matrix zur Beschreibung der Quermanöver, die als Input für die oben genannte Funktion übergeben wird.
- **create_long.m**: diese Funktion wird in vielen Funktionen verwendet. Sie entfernt irrelevante Manöver oder Aktionen aus den Matrizen zur Beschreibung der Längsmanöver, z.B. Gangwechsel, der durch einen starken Ruck gekennzeichnet ist. Außerdem verbindet die Funktion die Spalten der oben genannten Matrizen basierend auf einer Zeitschwellwert, in dieser Arbeit basierend auf Gangwechseldauer=1s
- **calc_MAPE.m**: berechnet die MAPE-Fehler für die Schätzung der Profile für Geschwindigkeit (Beschleunigung und Verzögerung) und laterale Position (Spurwechsel). Diese Funktion behandelt auch der Fall des Teilens durch Null.

*Hinweis: Die Funktionen **check_adj.m** und **permute_sorted_col.m** enthalten Operationen auf Matrizen. Siehe noch die Dokumentation im Code.*

3- Schreiben in die Kopie des Infosfiles des Testlaufes

die in Schritt A erzeugte Kopie wird durch die Funktion **overwrite_infofile.m** überschrieben.

Overwrite_infofile.m nimmt der erzeugte Tensor und die zu überschreibende Datei als Inputargumente entgegen. Das Schreiben der Manöver der Infofile-Kopie basiert sich auf den Werten des Tensors.

4- Nachsimulation durchführen

Nach dem Schreiben der Manöver wird die Nachsimulation durch die Funktion **run_sim_via_DDE.m** durchgeführt.

*Hinweis: Wenn der MATLAB-Fehler „channel is not valid“ im MATLAB-CommandWindow auftritt, bitte MATLAB einfach neu starten und das Skript über **main.m** erneut ausführen.*

5- CM-Daten importieren und vorverarbeiten

Analog zur ersten Simulation werden die Daten der Nachsimulation importiert und vorverarbeitet.

6- Tensor der Nachsimulation extrahieren

Analog zur Ersten Simulation wird in diesem Schritt der Tensor der Nachsimulation durch die Funktion `create_tensor_resim.m` erzeugt. Dabei wird keine Schätzung verwendet, sondern nur GT-Daten.

7- Auswertung durchführen

Zur Auswertung des extrahierten Tensors dient die Funktion `event_dev_func.m`. Diese Funktion berechnet für jeden Manöver- bzw. Zustandsbasierten Ereignis die relative Abweichung für jeden Metaparameter.

8- Plot für Ego

Die endgültigen Abbildungen mit jeweils 2 Subplots am Ende der Ausführung des Codes stellen die Plots der Geschwindigkeit und laterale Position für Ego für jeden Testlauf (TestRun) dar.

C- Ausgabe des Codes

Hauptausgabe des Codes (im Workspace von MATLAB) ist der Struct-Variable `TestRun`. Dieses Struct enthält für jeden Testlauf die Tensor-Matrizen für Ego und TObjs jeweils für erste- und Nachsimulation sowie die berechneten Abweichungen. Darüber hinaus ist im Struct `first_sim` die Anzahl an den detektierten TObjs in der ersten Simulation zu finden.