DySi_Select

Version

Table of Contents

Contents:

Welcome to DySi_Select's documentation!

DySi_Select_Simulation Software

DySi_Sel_Traffic_v01 module

Spawning Traffic into the simulation

DySi_Sel_Traffic python module is used for spawning the necessary traffic for evaluation of DySi_Select concept. Due to the use of “seed”, the traffic is always spawned at the same points i.e. repeatability ensured.

DySi_Sel_Traffic_v01. spawn_traffic ( ) [source]

spawn_traffic interface of the DySi_Sel_Traffic. This is used to spawn traffic onto the CARLA world. The arguments to be parsed via CLI are available within the module itself to ensure non-interference between different involved modules.

DySi_Sel_utils module

Welcome to CARLA DySi_Select

CTRL + W : toggle constant velocity mode at 50 km/h N : next sensor F1 : toggle HUD H/? : toggle help ESC : quit

class DySi_Sel_utils. FadingText (*font* , *dim* , *pos*) [source]

Bases: `object`

Helper class that handles notifications

render (*display*) [source]
set_text (*text* , *color = (255, 255, 255)* , *seconds = 2.0*) [source]
tick (_ , *clock*) [source]

class DySi_Sel_utils. HUD (*width* , *height*) [source]

Bases: `object`

HUD class is used to display information about the actors in the DySi_Select simulation

Parameters :

object (*class*) – base class

error (*text*) [source]
notification (*text* , *seconds = 2.0*) [source]
on_world_tick (*timestamp*) [source]
render (*display*) [source]
tick (*world* , *clock*) [source]
toggle_info ( ) [source]

class DySi_Sel_utils. HelpText (*font* , *width* , *height*) [source]

Bases: `object`

Helper class to handle text output using pygame

render (*display*) [source]
toggle ( ) [source]

DySi_Sel_utils. get_actor_display_name (*actor* , *truncate = 250*) [source]

get_actor_display_name interface is used to display the name of the actors in the CARLA world

Parameters :

- **actor** (*any*) – actor in CARLA world
- **truncate** (*int , optional*) – limit the size of name. Defaults to 250.

Returns :

name of the actor from CARLA world

Return type :

str

DySi_Sel_utils. get_args ( ) [source]

DySi_Sel_v01 module

Welcome to CARLA DySi_Select

CTRL + W : toggle constant velocity mode at 50 km/h N : next sensor F1 : toggle HUD H/? : toggle help ESC : quit

class DySi_Sel_v01. CameraManager (*parent_actor* , *hud*) [source]

Bases: `object`

CameraManager provides an instance of necessary camera sensors for use in the simulation. By default, a RGB camera will be run while initializing the simulation. Alternatively, a Semantic Segmentation camera in CityScapes paletter is provided which can be visualized by pressing “n” after start of simulation. This is provided in the main python file instead of utils to avoid use in other modules.

next_sensor ( ) [source]
render (*display*) [source]
set_sensor (*index* , *notify = True* , *force_respawn = False*) [source]

class DySi_Sel_v01. SimControl (*world* , *start_in_autopilot*) [source]

Bases: `object`

SimControl provides an instance of necessary control of simulation. The instance of this class allows the client to switch between sensors and change certain simulation parameters.

parse_events (*client* , *world* , *clock*) [source]

class DySi_Sel_v01. World (*carla_world* , *hud* , *args*) [source]

Bases: `object`

World class provides an instance of the world to which the client vehicle is to be spawned. All necessary conditions for the ego-vehicle/client is set here.

destroy ( ) [source]
destroy_sensors ( ) [source]
modify_vehicle_physics (*vehicle*) [source]
render (*display*) [source]
restart ( ) [source]
tick (*clock*) [source]

DySi_Sel_v01. game_loop (*args*) [source]

game_loop() interface initializes the ego-vehicle client and runs the necessary instances for simulation.

Parameters :

args (*config*) – parsed configuration parameters from CLI

DySi_Sel_v01. main ( ) [source]

main interface of the DySi_Select simulation.

DySi_Select_Parking_v01 module

DySi_Select_Parking_v01 python module is used for prototyping the DySi_Select concept. This module provides the spawns parked vehicles in parking spots or on the kerbs of Town01,02 and 03.

DySi_Select_Parking_v01. park_vehicles ( ) [source]

Interface to spawn parked cars in Towns 01,02 and 03

SIN package

Subpackages

SIN.src package

Subpackages

SIN.src.features package

Submodules

SIN.src.features.build_features module

Module contents

SIN.src.utils package

Submodules

SIN.src.utils.SIN_utils module

SIN_utils python module provides the utility interfaces for handling data from SIN efficiently.

SIN.src.utils.SIN_utils. gen_conf_matrix (*ground_truth* , *prediction* , *op_dir*) [source]

Interface for generating the confusion matrix for the evaluation of network performance.

Parameters :

- **ground_truth** (*any*) – ground truth data
- **prediction** (*any*) – predicted data
- **op_dir** (*str*) – output directory to store results

SIN.src.utils.SIN_utils. get_label ( ) [source]

Interface for converting predicted class to label.

Returns :

class label

Return type :

str

SIN.src.utils.data_utils module

data_utils python module provides the utility interfaces for handling data efficiently. The utils from Cam2BEV is reused in most cases, to avoid rework. To ensure, modules are independent of each other, a new data_utils is created in SIN module.

SIN.src.utils.data_utils. abspath (*path*) [source]

Interface for obtaining the absolute path of a file/folder.

Parameters :

path (*str*) – file path to be converted to absolute file path

Returns :

absolute file path

Return type :

str

SIN.src.utils.data_utils. **listToString** (*s*) [source]

Interface for converting list to string.

Parameters :

s (*list*) – parsed list

Returns :

converted string

Return type :

str

SIN.src.utils.data_utils. **load_image** (*filename*) [source]

Interface for loading the image.

Parameters :

filename (*str*) – path for loading the image

Returns :

loaded image

Return type :

image

SIN.src.utils.data_utils. **parse_test_configs** ( ) [source]

Interface to parse the parameters and files for testing the SIN network.

Returns :

parameters from the parsed yml file

Return type :

loaded config file

SIN.src.utils.data_utils. parse_train_configs ( ) [source]

Interface to parse the parameters and files for training the SIN neural network.

Returns :

parameters from the parsed yml file

Return type :

loaded config file

SIN.src.utils.data_utils. parse_val_configs ( ) [source]

Interface to parse the parameters and files for validation of the SIN neural network.

Returns :

parameters from the parsed yml file

Return type :

loaded config file

SIN.src.utils.model_utils module

model_utils python module provides the utility interfaces for handling the neural network efficiently. The utils from Cam2BEV is reused in most cases, to avoid rework.

SIN.src.utils.model_utils. load_module (*module_file*) [source]

Interface for loading a python file.

Parameters :

module_file (*str*) – python file to be loaded

Returns :

loaded python file

Return type :

python file

Module contents

Submodules

SIN.src.evaluate_model module

evaluate_model python module is used for training the neural network SIN (Situation Identification Network). All parameters related to the evaluation of the network is available in the config files in the models directory. The structure is inspired by the module Cam2BEV.

SIN.src.evaluate_model. main ( ) [source]

SIN.src.predict_situation module

predict_situation python module is used to provide interfaces training the neural network SIN (Situation Identification Network). All parameters related to the evaluation the network is available in the config files in the models directory.

SIN.src.predict_situation. get_network_sin (*cam_set = 'single'* , *weights_dir = 'SIN/src/output/singlecam/stored_cp/Checkpoints/best_weights.hdf5'*) [source]

Interface for loading the SIN model.

Parameters :

- **cam_set** (*str* , *optional*) – camera setting used for evaluating the situation. Defaults to ‘single’.

- **weights_dir** (*str* , *optional*) – file path to the pre-trained SIN weights. Defaults to 'SIN/src/output/singlecam/stored_cp/Checkpoints/best_weights.hdf5'.

Returns :

loaded SIN model

Return type :

Model

SIN.src.predict_situation. **predict_situation** (*model* , *bev_img*) [source]

Interface for predicting the situation for a BEV image. The interface can take an image from single image source, from CARLA or from a frame of real-world dataset

Parameters :

- **model** (*Model*) – pre-trained SIN model instance
- **bev_img** (*str*) – lcoation of the bev image

Returns :

predicted situation class

Return type :

str

SIN.src.train_model module

train_model python module is used for training the neural network SIN (Situation Identification Network). All parameters related to the training of the network is available in the config files in the models directory. The training structure is inspired by the module Cam2BEV.

SIN.src.train_model. **main** ( ) [source]

Module contents

Submodules

SIN.setup module

SIN.test_environment module

SIN.test_environment. main ( ) [source]

Module contents

SemSeg package

Submodules

SemSeg.deeplabv3plus module

Deeplabv3+ model for Keras. This model is based on TF repo: https://github.com/tensorflow/models/tree/master/research/deeplab On Pascal VOC, original model gets to 84.56% mIOU

MobileNetv2 backbone is based on this repo: https://github.com/JonathanCMitchell/mobilenet_v2_keras

Reference - [Encoder-Decoder with Atrous Separable Convolution

for Semantic Image Segmentation](https://arxiv.org/pdf/1802.02611.pdf)

- **[Xception: Deep Learning with Depthwise Separable Convolutions]**
  (https://arxiv.org/abs/1610.02357)

- **[Inverted Residuals and Linear Bottlenecks: Mobile Networks for**
  Classification, Detection and Segmentation](https://arxiv.org/abs/1801.04381)

SemSeg.deeplabv3plus. Deeplabv3 (*weights = 'pascal_voc'* , *input_tensor = None* , *input_shape = (512, 512, 3)* , *classes = 21* , *backbone = 'mobilenetv2'* , *OS = 16* , *alpha = 1.0* , *activation = None*) [source]

Instantiates the Deeplabv3+ architecture

Optionally loads weights pre-trained on PASCAL VOC or Cityscapes. This model is available for TensorFlow only. # Arguments

weights: one of 'pascal_voc' (pre-trained on pascal voc),
'cityscapes' (pre-trained on cityscape) or None (random initialization)

input_tensor: optional Keras tensor (i.e. output of *layers.Input()* **)**
to use as image input for the model.

input_shape: shape of input image. format HxWxC
PASCAL VOC model was trained on (512,512,3) images. None is allowed as shape/width

classes: number of desired classes. PASCAL VOC has 21 classes, Cityscapes has 19 classes.
If number of classes not aligned with the weights used, last layer is initialized randomly

backbone: backbone to use. one of {'xception','mobilenetv2'} activation: optional activation to add to the top of the network.

One of 'softmax', 'sigmoid' or None

OS: determines input_shape/feature_extractor_output ratio. One of {8,16}.
Used only for xception backbone.

alpha: controls the width of the MobileNetV2 network. This is known as the width multiplier in the MobileNetV2 paper.

- **If** *alpha* **< 1.0, proportionally decreases the number** of filters in each layer.
- **If** *alpha* **> 1.0, proportionally increases the number** of filters in each layer.
- **If** *alpha* **= 1, default number of filters from the paper** are used at each layer.

Used only for mobilenetv2 backbone. Pretrained is only available for alpha=1.

 # Returns
A Keras model instance.

 # Raises

RuntimeError: If attempting to run this model with a

backend that does not support separable convolutions.

ValueError: in case of invalid argument for *weights* or *backbone*

SemSeg.deeplabv3plus. SepConv_BN (*x* , *filters* , *prefix* , *stride = 1* , *kernel_size = 3* , *rate = 1* , *depth_activation = False* , *epsilon = 0.001*) [source]

SepConv with BN between depthwise & pointwise. Optionally add activation after BN Implements right “same” padding for even kernel sizes :param x: input tensor :param filters: num of filters in pointwise convolution :param prefix: prefix before name :param stride: stride at depthwise conv :param kernel_size: kernel size for depthwise convolution :param rate: atrous rate for depthwise convolution :param depth_activation: flag to use activation between depthwise & poinwise convs :param epsilon: epsilon to use in BN layer

SemSeg.deeplabv3plus. preprocess_input (*x*) [source]

Preprocesses a numpy array encoding a batch of images. # Arguments

x: a 4D numpy array consists of RGB values within [0, 255].

 # Returns

Input array scaled to [-1.,1.]

SemSeg.deeplabv3plus. relu6 (*x*) [source]

SemSeg.sem_seg module

sem_seg python module is used to get semantic segmentation of any input frame and will be useful when the model is applied for real-world dataset. The semantic segmented image will be used as an input to the Cam2BEV module. Handling of one-hot encoding is done in Cam2BEV itself.

SemSeg.sem_seg. get_network_semseg ( ) [source]

Interface for loading the semantic segmentation model.

Returns :

pre-trained semantic segmentation network model

Return type :

Model

SemSeg.sem_seg. predict_semseg (*model* , *original_img*) [source]

Interface for obtaining the semantically segmented image from a SemSeg network and preprocess RGB image.

Parameters :

- **model** (*Model*) – pre-trained model instance of semantic segmentation network
- **original_img** (*str*) – location of the input RGB image frame

Returns :

location of semantically segmented image frame

Return type :

str

SemSeg.semseg_utils module

semseg_utils python module is used to define helper interfaces for sem_seg module.

SemSeg.semseg_utils. cs_label_colormap ( ) [source]

Interface for creating a label colormap used in Cityscapes segmentation benchmark.

Returns :

A colormap for visualizing segmentation results

Return type :

dict

SemSeg.semseg_utils. decode_semsegmsk (*mask* , *colormap*) [source]

Interface for decoding the predictions as per the Cityscapes scheme.

Parameters :

- **mask** (*array*) – mask to be used for semantic segmentation
- **colormap** (*dict*) – color map for cityscapes palette

Returns :

A colormap for visualizing segmentation results

Return type :

array

SemSeg.semseg_utils. get_network ( ) [source]

Interface for obtaining the DeepLabv3+ architecture for semantic segmentation with pre-trained weights trained on CityScapes.

Returns :

instance of the DeepLabv3+ network

Return type :

Model

SemSeg.semseg_utils. get_overlay (*image* , *pred_mask*) [source]

Interface for getting colored overlay on the image.

Parameters :

- **image** (*array*) – image as numpy array
- **pred_mask** (*array*) – predicted mask for overlaying on the image

Returns :

overlay array for segmenting an image

Return type :

array

SemSeg.semseg_utils. plot_samples (*display_ls* , *figsize*) [source]

Interface for plotting the images and saving to disk.

Parameters :

- display_ls (*list*) – number of columns on the image
- figsize (*array*) – size of the figure in the plotted image

SemSeg.semseg_utils. read_image (*image*) [source]

Interface for converting the input image to a preprocessed tensor.

Parameters :

image (*str*) – location of the input image

Returns :

the preprocessed tensor form of parsed image

Return type :

tensor

SemSeg.semseg_utils. save_eval (*outputDir*) [source]

Interface for saving the images for further processing and evaluation.

Parameters :

outputDir (*str*) – output folder directory

Returns :

the file name is appended to output directory

Return type :

str

Module contents

evaluate_real_dataset module

evaluate_real_dataset python module is used to simulate the frames of real-world dataset to create a continuous flow of frames and test the DySi_Select concept for real-world data.

evaluate_real_dataset. data_select (*sitn_cls*) [source]

Sample interface for selecting relevant data. This needs additional tracking and distance calculation algorithms.

Parameters :

sitn_cls (*str*) – identified situation class

Returns :

relevant data in a particular situation

Return type :

str

evaluate_real_dataset. flow_frames (*frame_dir* , *argument*) [source]

Interface for obtaining continuous flow of frames to simulate a real-life automotive scenario using a publicly available dataset

Parameters :

- frame_dir (*str*) – directory of the dataset with image frames
- argument (*config*) – parsed arguments from CLI

evaluate_real_dataset. main ( ) [source]

Interface for parsing the CLI arguments and run the evaluation on public dataset

evaluate_real_dataset. render_window ( ) [source]

Interface for initializing the display windows

playground_bev module

playground_bev python module is used to test the prediction of bev from a semantically segmented input image. This provides usage example of the bev generation from the created interfaces in Cam2BEV

playground_sem module

playground_sem python module is used to test the prediction of semantic segmentation from a RGB input image. This provides usage example of the semantically segmented image generation from the created interfaces in SemSeg module

playground_sin module

playground_sin python module is used to test the prediction of situation class from a bev input image. This provides usage example of the situation class prediction from the created interfaces in SIN

Cam2BEV

deeplab_mobilenet module

Deeplab-Mobilenet module to get neural network for error-correcting a homography image and get a BEV image with marked occlusions.

deeplab_mobilenet. get_network (*input_shape* , *n_output_channels*) [source]

Get DeepLab-Mobilenet instance

Parameters :

- **input_shape** (*int*) – shape of input image (H,W,C)
- **n_output_channels** (*int*) – number of output labels

Returns :

DeepLab-Mobilenet instance

Return type :

Model

deeplab_xception module

DeepLab-Xception module for error correcting a homography input to produce BEV image marked with occlusions.

deeplab_xception. get_network (*input_shape* , *n_output_channels*) [source]

Get DeepLab-Xception instance

Parameters :

- **input_shape** (*int*) – shape of input image (H,W,C)
- **n_output_channels** (*int*) – number of output labels

Returns :

DeepLab-Mobilenet instance

Return type :

Model

uNetXST module

uNetXST module provides the uNetXST neural network for image to BEV image generation.

uNetXST. decoder (*encoder_layers* , *udepth=3* , *filters1=8* , *kernel_size=(3* , *3)* , *activation=<function relu>* , *batch_norm=True* , *dropout=0.1*) [source]

uNetXST. encoder (*input* , *udepth=3* , *filters1=8* , *kernel_size=(3* , *3)* , *activation=<function relu>* , *batch_norm=True* , *dropout=0.1*) [source]

uNetXST. get_network (*input_shape* , *n_output_channels* , *n_inputs* , *thetas* , *udepth=5* , *filters1=16* , *kernel_size=(3* , *3)* , *activation=<function relu>* , *batch_norm=True* , *dropout=0.1* , *double_skip_connection=False*) [source]

uNetXST. joiner (*list_of_encoder_layers* , *thetas* , *filters1=8* , *kernel_size=(3* , *3)* , *activation=<function relu>* , *batch_norm=True* , *double_skip_connection=False*) [source]

predict_bev module

predict_bev python module is used to predict the bev from a semantically segmented input image. This is a newly developed module for the functionality of DySi_Select. This module helps in verifying the situation identification in vehicles.

predict_bev. get_bev_network (*backbone = 'unetxst'* , *weights_dir = 'Cam2BEV/model/output/unetxst_singlecam/finetuned/Checkpoints/best_weights.hdf5'*) [source]

Interface for obtaining the bev generation neural network model.

Parameters :

- **backbone** (*str , optional*) – network architecture backbone to be used for bev generation.
- **options** (*2*) – unetxst or deeplab. Defaults to 'unetxst'.
- **weights_dir** (*str , optional*) – location of the pre-trained model weights.
- **'Cam2BEV/model/output/unetxst_singlecam/finetuned/Checkpoints/best_weights.hdf5'.** (*Defaults to*) –

Returns :

the loaded model based on the parsed inputs

Return type :

model

predict_bev. parse_sample (*input_file*) [source]

Interface for parsing the input image.

Parameters :

input_file (*str*) – location of the parsed image

Returns :

pre-processed image as a numpy array

Return type :

numpy array

predict_bev. **predict_bev** (*model* , *one_hotd_label* , *sem_img* , *backbone = 'unetxst'*) [source]

Interface for obtaining the bev from a semantically segmented image.

Parameters :

- **model** (*Model*) – parsed Cam2BEV model
- **one_hotd_label** (*list*) – one-hot encoded color palette for converting CityScapes color to 4 classes.
- **sem_img** (*str*) – location of the input semantic segmented image
- **backbone** (*str , optional*) – network backbone to be used. Ensure same backbone is
- **'unetxst'.** (*parsed as that of the model. Defaults to*) –

Returns :

location of the predicted bev image

Return type :

str

predict_bev. **save_bev_eval** (*outputDir*) [source]

Interface for saving the images for further processing and evaluation.

Parameters :

outputDir (*str*) – location of the output directory

Returns :

file name appended to the output directory

Return type :

str

utils module

get_ipm module

get_ipm python module is used to convert the semantic segmented image to a BEV homography image. This is necessary if the Cam2BEV is based on DeepLab-Mobilenet network backbone

class get_ipm. Camera (*config*) [source]

Bases: `object`

Camera class instantiates a camera based on the parsed configuration.

Camera class is used to set the intrinsic and extrinsic parameters for the cameras involved.

K = *array([[0., 0., 0.], [0., 0., 0.], [0., 0., 0.]])*
P = *array([[0., 0., 0., 0.], [0., 0., 0., 0.], [0., 0., 0., 0.]])*
R = *array([[0., 0., 0.], [0., 0., 0.], [0., 0., 0.]])*
setK (*fx* , *fy* , *px* , *py*) [source]
setR (*y* , *p* , *r*) [source]
setT (*XCam* , *YCam* , *ZCam*) [source]
t = *array([[0.], [0.], [0.]])*
updateP ( ) [source]

get_ipm. get_homography (*semntc_img*) [source]

Interface for getting homography image for the input semantically segmented image. This is suitable currently for one camera only(i.e. Front). For adapting to multiple cameras, refer ipm.py and make changes.

Parameters :

semntc_img (*str*) – the semantic segmented image to be transformed

Returns :

the homography image for DeepLab-Cam2BEV

Return type :

str

get_ipm. homography_conv (*oldIPM* , *imgShape*) [source]

Interface for converting the IPM homography to handle resolutions other than Cam2BEV dataset resolution.

Cam2BEV dataset resolution: 1936x1216 –> 1936x968 (WxH format like in files)

Parameters :

- oldIPM (*array*) – old homogrpahy matrix (based on the Cam2BEV data configuration)
- imgShape (*array*) – shape of the parsed input image

Returns :

new homography matrix

Return type :

array

get_ipm. save_ipm_eval (*outputDir*) [source]

Interface for saving the images for further processing and evaluation.

Parameters :

outputDir (*str*) – output directory where the images are to be saved

Returns :

name of the file to be stored appended to the parsed directory

Return type :

str

predict module

train module

trial_ipm module

evaluate module

Indices and tables

- Index
- Module Index