Machine vision project

Creating a deep convolutional model that can be used for self driving car

Petar Petrov

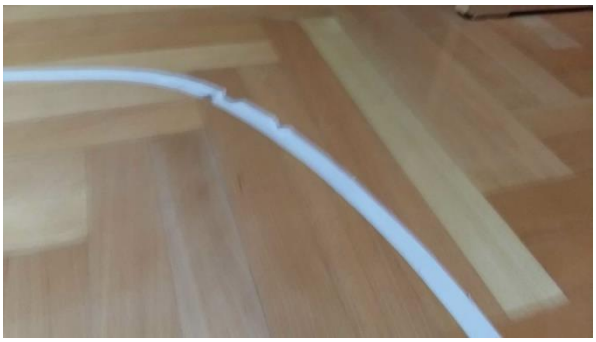
Index: 161052

Idea

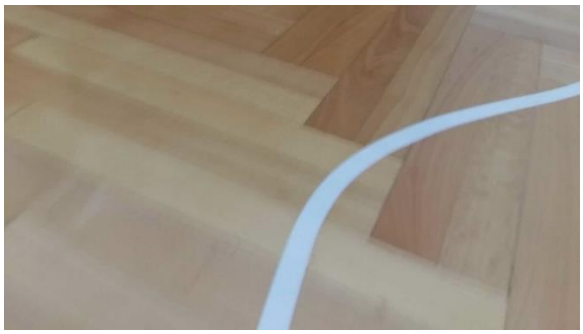
Creating a *simple* model that could be used for deciding which action the autonomous vehicle should perform. The model's output layer has 3 neurons, the three actions that the vehicle could perform: driving straight, turning left or turning right.

Data

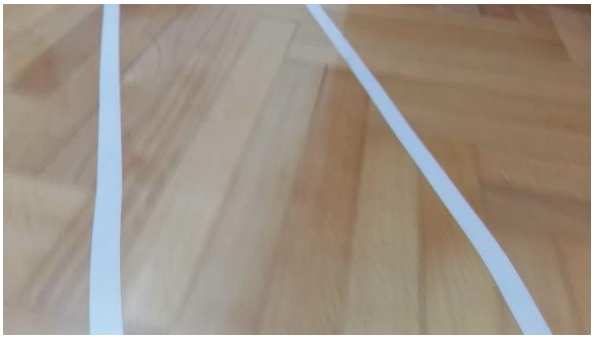
Data obtained from video taken during manually driving a vehicle and saving the particular frame into one of three folders: "w" (straight), "a" (left), "d" (right), corresponding to the action being taken at the moment. - saved in ***original_pictures*** folder.



Example picture from the ***a*** folder (turn left)



Example picture from the ***d*** folder (turn right)



Example picture from the w folder (keep straight)

Preprocessing

The raw pictures are resized and preprocessed so that only the lines are detected. The goal was to detect only the lines that the car should drive inbetween ("edges of the road"). **Canny** used as edge detector, **Hough Line Transform** used for lines detection.

In the ***line_detector.py*** file, the code is used to detect the lines and save the pictures into ***detected_lines*** folder. Essentially this creates the training data used to train the model.

Creating and training the model

Pictures with detected lines are used in order to create training data and train a simple *Convolutional deep learning model* capable of deciding what action the car should perform (left, right or straight). **Keras** and **Tensorflow** used for creating and training the model. Whole code is included in the model_creator.py file.

The jupyter notebook ***convolutional_model_training.ipynb*** contains the model summary and training procedure.

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_35 (Conv2D)           (None, 80, 80, 256)       12800
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 78, 78, 256)       590080
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 39, 39, 256)       0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 37, 37, 128)       295040
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 18, 18, 128)       0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 16, 16, 64)        73792
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
flatten_10 (Flatten)         (None, 4096)              0
_________________________________________________________________
dense_37 (Dense)             (None, 256)               1048832
_________________________________________________________________
dropout_22 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_38 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_23 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_39 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_40 (Dense)             (None, 3)                 99
=================================================================
Total params: 2,057,667
Trainable params: 2,057,667
Non-trainable params: 0
_________________________________________________________________
```

Summary of the created Deep learning model.

The model contains 4 Convolutional layers, Pooling layers and 4 Dense (Fully connected) layers, the latest being the output layer of the model. The output layer consists of 3 neurons corresponding to the 3 actions that can be taken – left, right or straight.

```
Train on 3407 samples, validate on 379 samples
Epoch 1/10
3407/3407 [==============================] - 29s 9ms/step - loss: 0.4203 - accuracy: 0.8268 - val_loss: 0.3418 - val_accuracy: 0.8997
Epoch 2/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.2696 - accuracy: 0.8984 - val_loss: 0.1975 - val_accuracy: 0.9367
Epoch 3/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.2216 - accuracy: 0.9210 - val_loss: 0.1867 - val_accuracy: 0.9314
Epoch 4/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.1776 - accuracy: 0.9387 - val_loss: 0.1679 - val_accuracy: 0.9340
Epoch 5/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.1632 - accuracy: 0.9480 - val_loss: 0.1566 - val_accuracy: 0.9446
Epoch 6/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.1424 - accuracy: 0.9554 - val_loss: 0.1491 - val_accuracy: 0.9499
Epoch 7/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.1230 - accuracy: 0.9589 - val_loss: 0.1703 - val_accuracy: 0.9551
Epoch 8/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.0999 - accuracy: 0.9674 - val_loss: 0.1512 - val_accuracy: 0.9525
Epoch 9/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.0988 - accuracy: 0.9662 - val_loss: 0.1538 - val_accuracy: 0.9551
Epoch 10/10
3407/3407 [==============================] - 29s 8ms/step - loss: 0.0965 - accuracy: 0.9680 - val_loss: 0.1205 - val_accuracy: 0.9657


<keras.callbacks.callbacks.History at 0x190bea02cc8>
```

Training procedure of the model.

Results and discussion

Training of the model was performed on a Nvidia GTX 980 GPU. 10 Epochs were selected, training it with more than 10 epochs did not yield any better results (probably overfitting).

The validation accuracy (accuracy of the validation set) of the model is 96.5%. Being very close to the training accuracy of 96.8%, the model performed really well on the validation set (pictures never seen before).

Output of the model directly translates to the action performed by the vehicle, so the movements may seem strange. On top of that, the model only outputs the direction of the turn and not at which angle the steering should be (full or partial turn, when the road curve is not so sharp). Because of that, the movement of the vehicle could seem more robotic and not so smooth.

Plans for future work include implementing this model to a vehicle and testing the real life performance.

Petar Petrov

Index: 161052