End-to-End Multiview Gesture Recognition for Autonomous Car Parking System

Hassene Ben AMARA[1], Fakhri KARRAY[1]

(1.*Centre for Pattern Analysis and Machine Intelligence (CPAMI), Electrical and Computer Engineering, The University of Waterloo*)

Abstract: The use of hand gestures can be the most intuitive human-machine interaction medium. The early approaches for hand gesture recognition used device-based methods. These methods use mechanical or optical sensors attached to a glove or markers, which hinder the natural human-machine communication. On the other hand, vision-based methods are less restrictive and allow for a more spontaneous communication without the need of an intermediary between human and machine. Therefore, vision gesture recognition has been a popular area of research for the past thirty years. Hand gesture recognition finds its application in many areas, particularly the automotive industry where advanced automotive human-machine interface (HMI) designers are using gesture recognition to improve driver and vehicle safety. However, technology advances go beyond active/passive safety and into convenience and comfort. In this context, one of America's big three automakers has partnered with the Centre of Pattern Analysis and Machine Intelligence (CPAMI) at the University of Waterloo to investigate expanding their product segment through machine learning to provide an increased driver convenience and comfort with the particular application of hand gesture recognition for autonomous car parking. The present paper leverages the state-of-the-art deep learning and optimization techniques to develop a vision-based multiview dynamic hand gesture recognizer for a self-parking system. We propose a 3D-CNN gesture model architecture that we train on a publicly available hand gesture database. We apply transfer learning methods to fine-tune the pre-trained gesture model on custom-made data, which significantly improves the proposed system performance in a real world environment. We adapt the architecture of end-to-end solution to expand the state-of-the-art video classifier from a single image as input (fed by monocular camera) to a Multiview 360 feed, offered by a six cameras module. Finally, we optimize the proposed solution to work on a limited resource embedded platform (Nvidia Jetson TX2) that is used by automakers for vehicle-based features, without sacrificing the accuracy robustness and real time functionality of the system.

Key words: Deep Learning; Video Classification; Dynamic Hand Gesture Recognition; Multiview; Embedded Platform; Automotive; Vehicle Self-Parking

1 General Concepts

1.1 Introduction

Nowadays, the most natural and intuitive means of communication for human-machine interaction is voice, thanks to the recent advances in the areas of automatic speech recognition and natural language processing. However, humans are a visual species by nature, which means that the most human-to-human communication we use is gesture, which can be manifested in our regular voluntary gestures and in our subconscious body language to convey most of the meaning of what we say while communicating with other humans. Gesture is a communication channel in the field of Human Machine Interface (HMI), which has many recent advancements: the hand gesture can be used to point (e.g., to replace the mouse), to manipulate objects (for augmented or virtual reality), to reinforce speech in a noisy environment or to communicate with a computer. The use of hand gestures can be the most natural and intuitive human-machine interaction medium. The early approaches for hand gesture recognition used device-based methods. These methods use mechanical or optical sensors attached to a glove or markers that transform hand motions to electrical signals, to determine posture. Using these methods, one can ac-

quire different information namely, angles, joints of the hand, and so on. However, these methods require that the person is wearing a device, which can be bulky and heavy, and hinders the natural interaction. On the other hand, vision-based methods are less restrictive and allow for a more natural communication without the need for an intermediary between human and machine. Therefore, vision gesture recognition has been a popular area of research for the past thirty years and, thanks to the success achieved in the field of artificial intelligence (deep learning, cognitive computing and multi-modal gesture recognition in particular), several gesture recognition applications have been developed: sign language recognition[1-2], virtual reality (where the hand is used to manipulate virtual objects and trigger actions, or navigate in a virtual environment)[3], augmented reality (where the physical world is augmented with virtual information, for example by back-projection)[4], multi-modal human-computer interface (which involves combination of gesture and speech recognition)[5], biometry (for example, personal recognition using hand shape and texture)[6], just to name a few. Similar to the voice recognition technology, gesture recognition is seeing its early days of real life application in the automotive industry. Choosing a car as an early adoption host for a new human-machine interaction technology is ideal, because the vehicle presents a controlled environment with a subset of possible interactions to test with. This makes the automotive industry a perfect test bed for testing gesture recognition in real life.

The automotive gesture recognition market value is estimated to reach USD 13.6 billion by 2024 according to a research report published in 2019 by Global Market Insights, Inc.[7]. As modern cars continue to offer more and more functionalities that require a growing number of commands, the options to control those features present important flaws when it comes to driver vehicle interaction, as they usually require visual attention of the user. The application of gesture recognition to advanced driver assistance systems allows the driver to use hand gestures to control various features of the car (e.g., infotainment system), thus reduces distracted driving risks and improves driving safety. Technology advances are not limited to driving safety. Many innovations in luxury cars pertain to passenger convenience and comfort. Automated driving is a hot topic in the car industry, with the automakers racing to be the first to bring a self-driving vehicle to the market. Some current Tesla, BMW and forthcoming Audi models will squeeze themselves into and out of tight parking spaces remotely without a driver need to be in the car. In this context, one of America's big three automakers has partnered with the Centre of Pattern Analysis and Machine Intelligence (CPAMI) at the University of Waterloo to investigate expanding their product segment through deep learning to provide an increased driver convenience and comfort with the particular application of hand gesture recognition for driver-less auto parking. This is an open-ended request that enables this work to explore a broad range of deep learning techniques and allows the goals to best suit the applicability of deep learning techniques in embedded environments for gesture recognition. We denote this automotive partner as "the automaker" in the remainder of the paper.

1.2 Problem Statement

Data released by Mercedes and BMW in 2015 show that while vehicle sizes have increased by up to 25 percent over the last 40 years, in many cases, the number of garages and parking spaces have remained constant[8]. In modern society, there is an everincreasing number of vehicles, and this has led to increasing difficulty to find large-enough spaces in busy parking lots. Parking can be a stressful endeavor, with the challenge of maneuvering into and out of a parking spot. Hence, drivers are in need of novel ways to park their cars without being behind the wheel. Being on the outside of the vehicle, the driver has a much better view of the hazards surrounding his car, hence he can park in tighter spots and it helps him avoid situations, such as illustrated in Fig.

1, where his vehicle is boxed in, which might cause a paint damage while trying to exit the car, or even put the driver in an embarrassing situation (e.g., exiting through the trunk).

Fig. 1 Driver squeezing himself out of his car parked in a tight parking space.

Several automakers have tackled this problem in the late 2016s and have developed some solutions, yet only few of them are in currently produced vehicles. The proposed systems make use of a mobile application on a smart phone to automatically park a car without a driver[9], where a button on the vehicle display key to activates the remote-controlled parking feature or a smart watch[10] that recognizes a configurable wave gesture and transmit it to the vehicle over wireless connection to trigger the parking action. All of the above mentioned solutions are device dependant (remote key, smart phone, smart watch). This represents a major drawback for these systems because of the disadvantages that come along with the usage of an additional hardware. In fact, any malfunctioning of the device (e.g., damage caused by water, low battery level) or unfavorable conditions (e.g., rainy or snowy weather) will render the self-parking feature unusable. Additionally, even though the above mentioned systems use a very common human-machine interaction medium (e.g., touch screen), it still presents an inconvenience to the users as it requires an intermediary medium between them and the car. To overcome the above mentioned weaknesses of the existent solutions, the automaker would like to, through this work, investigate the potential of deep leaning techniques in developing a multiview vision-based system for real life vehicle self-parking.

2 End-to-End Multiview Gesture Recognition: System Architecture

2.1 End-to-End System Overview

The vision-based multiview gesture recognition for self-parking system consists of two main modules: person detection and frames extraction module, and gesture recognition module. The input stream is a multiview 360 degree feed, offered by a six cameras system. The person detection module performs the detection of all subjects present in the six frames of video input. The resulted frames are then passed to a dynamic hand gesture classification module, which finally decides whether to initiate the

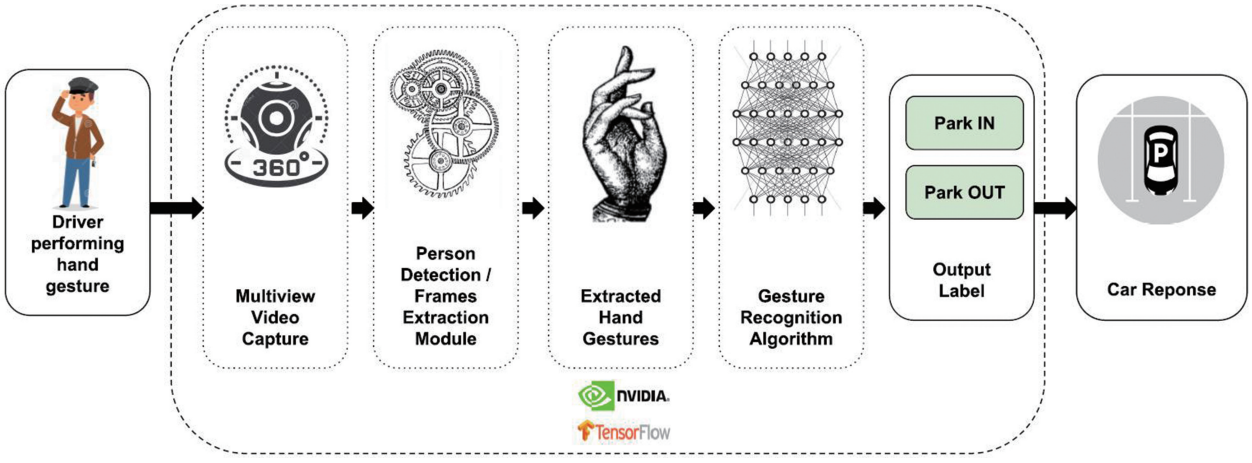

Fig. 2 End-to-End Multiview Dynamic Hand Gesture Recognition System Overview.

parking operation or not. The overall architecture of the end-to-end system is shown in Fig. 2.

2.2 Video Hardware Choice

For capturing 360 degree video frames, many hardware solutions exist on the market, and our choice was the HexCamera (e-CAM30_HEXCUTX2) from e-con systems, which consists of a multiple camera solution for NVIDIA Jetson TX1/TX2 developer kit. The setup consists of six cameras with 3.4MP each and an adaptor board to interface with the J22 connector on the Jetson. The camera can stream 720p (HD) and 1080p (Full HD) at 30 frames/s in uncompressed YUV422 Format. The camera system is presented in Fig. 3.

Fig. 3 e-CAM30_HEXCUTX2 - Six synchronized full HD cameras for NVIDIA Jetson TX1/TX2.

2.3 Person Detection and Frame Extraction Module

This module is the first core component of our end-to-end system. As mentioned in section 2.1, the output of the multiview camera is a six frame stream representing 360 degree view. One of the main challenges is to adapt the output of the multiview camera to the hand gesture recognition module, which will be presented in section 3. The 3D-CNN gesture recognition module is trained on video frames of size 176x100 where every video consists of one single subject performing the hand gesture. Given that the multiview camera output may contain multiple subjects in crowded environments (e.g., parking lot), the first step that the person detection and frames extraction module performs is the detection of all subjects present in the six frame video input. This module detects all the persons present in the 360 camera view feed, calculates the bounding box coordinates and finally crops over every 30 frames (required input length for the gesture recognition module) and saves separate image files for every subject. Once this step is complete, the resulted frames are passed to the gesture classification network to perform the hand gesture recognition. It is important to note that the authentication of the car owner is out of the scope of the present work, which means any subject performing one of the two gestures relevant to our system (Swiping Hand Left and Swiping Hand Right) would trigger the parking operation. The person detection module uses an underlying object detection library. The present paper evaluates two object detection tools that have been released recently using the convolutional neural networks: Faster R-CNN and YOLO. We have chosen these tools because YOLO allows to get the best results on VOC20072 data and VOC20123 and Faster R-CNN is one of the most used CNN methods so far. In the next subsections we will highlight our evaluation details and also the choice of the object detection library used in our final system.

2.3.1 Faster R-CNN

The first classifier tested for person detection is the algorithm created by S.Ren et al. [11], which relies on a detection made entirely with convolutional neural networks (Fig. 4):

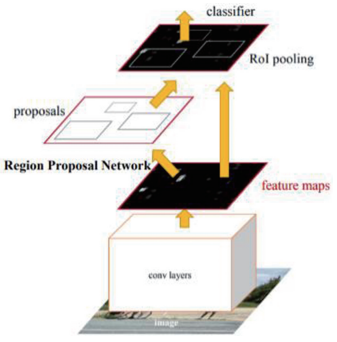

Fig. 4 Faster R-CNN: single, unified network for object detection.

- A first convolutional neural network takes as the input any image of any size and outputs regions where the objects to be detected might be.

- The second network takes as the input the regions proposed by the first network and decides whether they contain the object to be detected.

2.3.2 YOLO: You-Only-Look-Once

The second considered classifier, which is the final selection, is the algorithm developed by Redmon et al. in 2016[12]. Shorthand for You-Only-Look-Once, YOLO is a neural network capable of detecting what is in an image and its location, in a single forward pass of the image through the network, which was a shortcoming of previous methods. It gives the bounding boxes around the detected objects, and it can detect multiple objects at a time that was invented with the purpose of being able to perform real-time inference. This algorithm is based on two steps that are applied to images of predefined size when learning (Fig.5):

- Object detection using a convolutional neural network (CNN).

- A bounding box on the image where the predicted class of the object, if it exists (in our case a person or nothing).

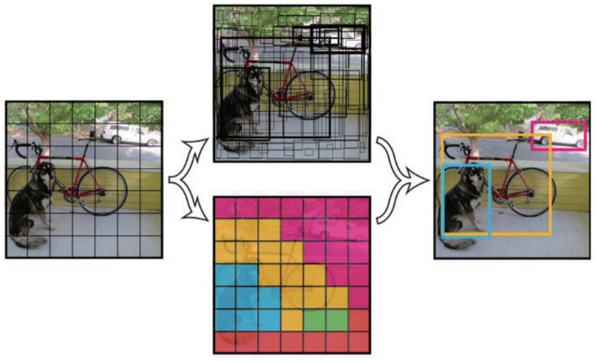

Fig. 5 Real-time object detection with YOLO: the algorithm model detection as a regression problem. It divides the image into an even grid and simultaneously predicts bounding boxes, confidence in those boxes and class probabilities.

The network first divides the input image into a grid (13 by 13 cells are used in the present paper). Each cell is responsible for predicting a fixed amount of bounding boxes (5 bounding boxes per cell is used in the present paper). Along with every bounding box prediction, where each box is associated with a confidence score prediction and a class prediction. The confidence score is a value of how certain the network is that the predicted bounding box encloses an object of any kind.

2.3.3 Comparison of the Two Techniques

One of the main differences between YOLO and Faster R-CNN is the computation time, YOLO allows a detection rate of 37 frames per second for an image of 445x445x3 while Faster R-CNN allows only 5 frames per second. Moreover, on the VOC2012 and VOC2007 data sets, YOLO seems to give better results. For our final implementation, YOLO v3, the latest version which is extremely fast and accurate, is used in conjunction with the underlying meaty part of the network: Darknet (a framework to train neural networks, it is open source and written in C/CUDA and serves as the basis for YOLO). For the purpose of our use case, we limit the object detection in YOLO v3 to only one class: Person.

2.3.4 Person Detection Workflow on Multiview Frames

Our six camera video system streams a live video feed in the format shown in Fig.6. It consists of the concatenation of six frames covering a 360 degree view.

Fig. 6 Sample output of the e-con hexcam six camera video output.

Fig. 7 Example of YOLO v3 real-time person detection and bounding boxes calculation.

The implemented algorithm continuously captures 30 frames (expected sequence length by the

dymanic hand gesture classification model) at 12 frames/s and passes them to the YOLO based person detector. The latter, only looks at the first frame of the 30 frame input and detects all the persons present, calculates the bounding boxes (example shown in Fig. 7) and finally crops all the 30 frames based on the respective boxes coordinates. The cropped videos are then saved in memory (Numpy array) to be passed one at a time to the gesture recognition module. An example of the cropped videos is shown in Fig.8.

Fig. 8 Sample cropped persons images: Left image shows a detected person performing the "Swiping Hand Left" gesture (used as Park IN trigger). Right image shows a detected person not performing any hand gesture. Both outputs consist of 30 frames each and are both passed to the 3D-CNN gesture recognition network.

2.4 Gesture Recognition Module

The hand gesture recognition module represents the second core component of our end-to-end system after the person detection module. It encompasses mainly our dynamic hand gesture classifier whose details will be presented in section 3. Once the person detection and cropping step is complete and the output frames (such illustrated in Fig. 8) are extracted, we resize them to match the dimensions of the expected input video by the 3D-CNN classifier, in our case with height *x* width *x* frames equal to 176 x 100 x 30. Then, we pass every input (consisting of 30 frames) to the hand gesture recognition network for classification. The output of this module can be one of the three classes: Swiping Hand Left (Park IN), Swiping Hand Down (Park OUT) or Doing Other Things (ignored by the system). Once one of the relevant classes (Park IN or Park OUT) is detected, the algorithm drops the rest of the input cropped videos and trigger the corresponding parking action.

3 Deep Learning-based Multiclass Hand Gesture Classifier

Hand gesture recognition can be treated as a multiclass classification problem that maps an input video sequence to one of the three classes our model has learned: C1 = Swiping Hand Left (Park In), C2 = Swiping Hand Down (Park Out) and C3 = Doing Other Things. The experimental results presented in section 3.5 will serve to evaluate the performance of the proposed gesture model, and to compare it with the state of the art method applied to this paper's use case. In this section, we present the machine learning methodology we followed in order to build the dynamic hand gesture classifier. First, the deep neural network architecture and the training process are presented. Then, the different experiments leading to the tuning of the hand-gesture classification network are described. Finally, the different test scenarios conducted while assessing the classifier as well as their corresponding results are reported.

3.1 Training Dataset: 20BN-JESTER Dataset

The 20BN-JESTER dataset consists of a large collection of densely-labeled video sequences taken by a static camera (webcam or laptop camera) that show humans performing predefined hand gestures. This dataset was collected thanks to a large number of crowd workers and made available by the German company TwentyBN[13], free of charge for academic research. In this database, we find a total of 148092 video sequences. The data was provided as a big archive containing directories numbered from 1 to 148092. Each folder corresponds to one video clip (single gesture) and contains JPEG images that were extracted at 12 frames/s having a height of 100px and variable widths. The length of sequences differs from one sequence to another. The dataset groups together 27 classes that represent the different hand gestures, namely: Swiping Hand Left, Swiping

Hand Down, Rolling Hand Forward, Doing Other Things, No Gestures, and so on. In each class, the hand gesture is performed by participants who represent a generalized distribution of gender, age, skin color, and at different speeds. The latter makes this dataset one of the largest data collections available to build a robust deep learning-based gesture classifier.

A study of our dataset revealed that the hand can produce avast diversity of gestures. However, it is extremely difficult to recognize all the possible configurations of the hand from its projection in an image. Indeed, according to the orientation of the hand in relation to the camera, some parts of the hand can be hidden. It is then necessary to consider an appropriate subset of gestures related to our application. For the present paper, the goal is to recognize three dynamic hand gestures for parking in and parking out actions and also other gestures (including no gesture). The two gestures that we want to recognize are: Swiping Hand Left (Park In trigger action) and Swiping Hand Down (Park Out trigger action). We considered these two gestures because they are among the most used in human-human interaction, and can be perfectly adapted to a natural man-car interaction. Furthermore, among other possible gestures available from 20BN-JESTER dataset, the high neural discriminability (i.e., decodability) between the two chosen gestures contributed in giving us the best model performance during our experiments. Fig. 9 shows a sample of the "Swiping Hand Left" gesture from 20BN-Jester dataset.

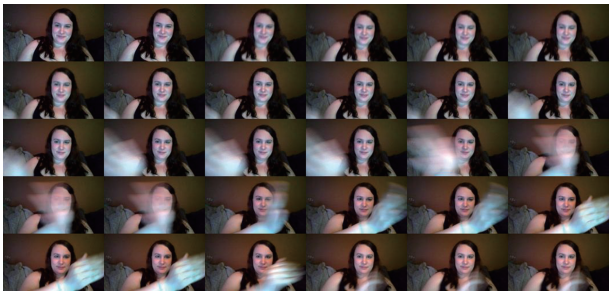

Fig. 9 Sample Swiping Hand Left hand gesture from 20BN-JESTER dataset.

3.2 Data Preprocessing

Due to the fact that video sequences from 20BN-JESTER dataset have different length (variable number of frames), the first step in our data preparation phase is to sub-sample every video down to 30 frames. So a 31-frame video and a 45-frame video will both be reduced to 30 frames, with the 45-frame video essentially being fast-forwarded. The decision to fix the sequence length to 30 was made after the inspection of the 20BN-JESTER dataset, which is mostly composed of videos with a length that varies from 27 to 46 frames. Also, a data cleaning step was performed to limit samples to only videos having a duration greater than the sequence lengths, therefore discarding all shorter videos (e.g., 28 frames).

For the context of our work, our model is trained to classify three gestures: C1 = Swiping Hand Left (Park In), C2 = Swiping Hand Down (Park Out) and C3 = Doing Other Things (which covers other possible gestures including no gesture). For that purpose, we used 20BN-JESTER dataset to extract a subset of data containing all videos for the aforementioned three classes. We divided our dataset into three subsets: training, validation and evaluation, the latter two being generally smaller than the first. It is through the ratio (Training: 75%, Validation: 12.5%, Testing: 12.5%) that we can ensure the capacity of the model to generalize well and avoid overfitting. Our training data set consists of 2601 video sequences for the C1, 2641 videos for C2 and 8601 videos for C3. The latter class has more than 3 times the number of video sequence to represent real life scenarios, as most gestures do not belong to the first two classes. The validation dataset consists of 437 videos for C1, 428 videos for C2, and 2438 for C3. Finally, to evaluate our model, the evaluation dataset contains 430 videos for both C1, C2 and 2437 videos for C3.

There are many machine learning methods in the literature that work well on temporal classification tasks as encountered in our dynamic hand ges-

tures classifier. After a review of many of these methods and reported results, we limited our experiments to the following learning algorithms: 3D Convolutional Neural Networks and Long-term Recurrent Convolutional Networks.

3.3 Dynamic Hand Gesture 3D-CNN Classifier

Using supervised convolutional neural network (CNN) models on image recognition tasks has achieved a dramatic progress. Also, a number of extensions have been proposed to tackle the challenging video recognition problem. Based on the work done by Molchanov et al.[14] on hand gesture recognition using 3D Convolutional Neural Networks, we trained and fine-tuned a variant of 3D-CNN classifier.

Three types of layers are usually used to form a convolutional network. A 2D-CNN is composed of convolution layer(s), Pooling layer(s) and finally fully connected layer(s). The fully connected layer(s) are often used as the network output. Usually, a convolution layer is followed by an activation function and then a pooling layer, and this sequence can be repeated several times up to the fully connected layer to form a convolution network that is often denoted under the CONVNET notation. It is also common to use more than one fully connected layer before the output of the network. On the other hand, 3D-CNN applies a third dimensional filter to the dataset and the filter moves in 3-directions (x, y, z) to learn the low-level feature representations. Their output shape is a three dimensional volume space such as a cube. Our model therefore consists of eight convolution layers, five layers of max-pooling, two fully connected layers, and finally a softmax output layer. Fig. 10 shows the final 3D-CNN architecture of our gesture model for classifying three different types of dynamic hand gestures.

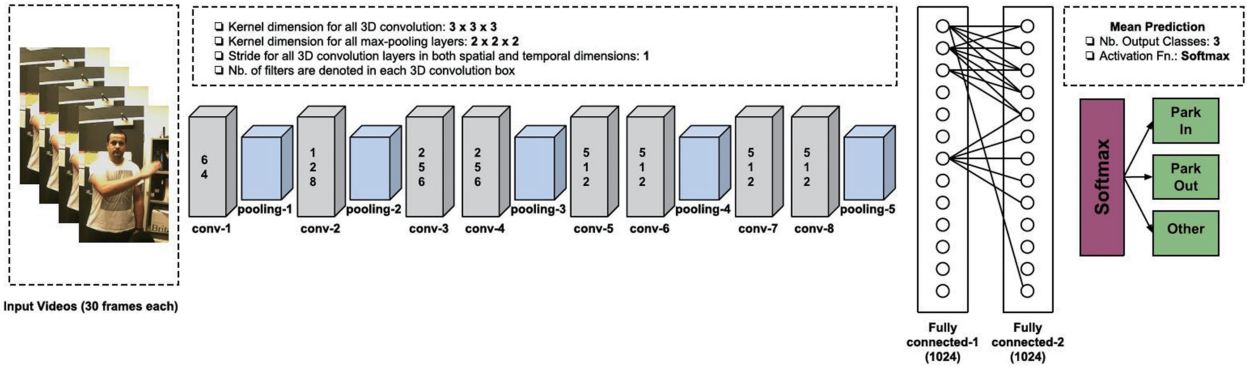

Fig. 10 Our final model architecture: 3D Convolutional Neural Network for dynamic hand gesture Recognition.

To determine the optimal architecture and parameters of our gesture model (number of convolution layers, number of neurons per layer, number of max-pooling layers, etc.), many models with different configurations have been trained on the dataset presented in section 3.1. The results obtained from those experiments have helped us determine the best model architecture that produced the best performance for our use case. One of the drawbacks of DNN is the difficulty to select the network hyper-parameters, which makes the network tuning one of the major phases in a connectionist modeling-based machine learning application. The techniques followed to tune the 3D-CNN gesture neural network are given now.

Dropout Selection Dropout is a regularization technique for neural networks proposed by Srivastava et al. in their 2014 paper[15] consisting of randomly dropping units along with their connections from the neural network architecture during the training phase. Hence, avoiding overfitting. Dropout is mainly applied to connected layers and requires a parameter to know the number of units to be eliminated at each iteration, which is often expressed as the rate of units

to be conserved. For example, a rate of 10% will eliminate 90% of connections. This technique indicates dropping out hidden and visible units of a neural network, which results in deleting them from the network along with the outgoing and incoming connections. In Keras, dropout is simply implemented by randomly selecting nodes to be dropped-out with a given probability (e.g., 50%). For our 3D-CNN model training, a dropout of 50% was used between the two dense fully connected layers after convolutional and pooling layers.

Early stopping when training a learner with an iterative method, such as gradient descent, early stopping consists of a regularization technique that involves stopping the training of the neural network when the minimal validation error is achieved (or in other words validation accuracy starts to decrease). Thus, preventing the generalization performance from degrading and falling into overfitting. We use the validation dataset to determine when to stop the optimization by monitoring the progress of the calculated error on that validation data and stopping the optimization when the error starts to increase. In this work, and after experimenting with various values, early stopping method was used with a patience set to 5, which represents the number of epochs before the validation loss stops improving.

3.3.1 Optimization Parameters

We used Adam (adaptive momentum estimation) optimization algorithm which is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Two parameters are required for Adam optimizer: adaptive learning rate and decay.

Adaptive Learning Rate Learning rate is a hyper-parameter that controls the step ratio while adjusting the weights of our network with respect to the loss gradient. The lower the value the slower we step along the downward slope. While this might be a good idea (using a low learning rate) in terms of making sure that we do not get stuck in any local minima, it could also mean that it will take a long time to converge, especially if we get stuck on a plateau region. We denote α as the learning rate. The following formula shows the relationship:

$$\theta_1 := \theta_1 - \alpha \frac{0}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow. If α is too large, gradient descent can overshoot and miss the global minima. It may fail to converge, or even diverge. In our algorithm, we chose the value of the learning rate as 1e-5, which gave us the best model performance after running many experiments with various values.

Learning Rate Decay One of the things that might help speeding up the learning algorithm, is to slowly reduce the learning rate overtime. This allows to adjust the learning rate, which is called learning rate decay. Common learning rate decays include timebased decay, step decay and exponential decay. The following formula illustrates how the learning rate α is updated as the number of epochs increases:

$$\alpha = \frac{1}{1 + decayRate * epochNr} \alpha_0$$

Whereas the learning rate decay does help speeding up training, during our experiments, we noticed that its importance in terms of the hyper-parameters to be tuned was lower than the learning rate α that had a huge impact on the model performance if well tuned. The final decay value used in our training algorithm was 1e-6.

3.3.2 Model Training

The purpose of our classification is to decide whether a video contains one of the two relevant gestures for our use case: Swiping Hand Left (Parking in) or Swiping Hand Down (Parking out). To resolve this problem, training the classifier was performed using a subset of labeled RGB images from 20BN-Jester dataset. For the implementation of the training algorithm, we used Keras, an open source neural network library written in Python and TensorFlow as the backend. Table 1 gives the specifications tof the computer that is used for our implementations

and experimentation. The input to the 3D-CNN network is 30 frames from the training dataset of a given dynamic hand gesture reshaped to (176x100x3) size.

Table 1 Specifications of the computer used for implementing, training and testing the gesture recognition classifiers.

Property	Specification
Processor	Intel Core i5-6600 CPU @ 3.50 GHz
GPU	NVIDIA GeForce GTX 1070/PCIe/SSE2
Memory	16 GB
OS	Ubuntu 16.04.5 LTS

We used the back-propagation algorithm to adjust the network weights and ReLu as the activation function, with a batch size of 6 (experimented with higher values of batch size to speedup training but rapidly hit the memory limit). Also, a dropout of 0.5 was used between the two fully connected layers, which helped avoid overfitting. We compute the validation error (also known as--aka loss) after each epoch with an early stopping patience value set to 5, to stop the training once the validation error stops decreasing for 5 consecutive epochs. Training our classifier took ~11 hours and went through a total of 10 epochs. Fig. 10 and Fig. 11 show the loss and accuracy curves for validation and training when the network is being trained.

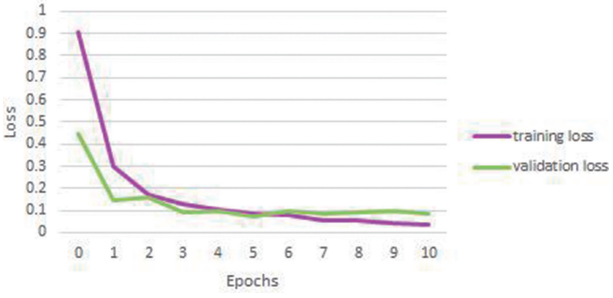

Fig. 11 Training and validation loss vs training epochs (3D-CNN).

We based our interpretation of these results on the following definitions:

- Underfitting: Refers to a model that can neither model the training data nor generalize to new data. In this case, the training and validation losses are both high and the accuracy is low.

- Overfitting: An overfit model is one where the loss on the training set is low and continues to decrease, whereas the validation loss decreases to a point and then begins to increase at the same time where accuracy begins to degrade.

- Good fit: A good fit model has a good accuracy on both training and validation sets. This can be diagnosed from the loss curve where the training and validation losses decrease and stabilize around the same point.

- Unknown adjustment: The validation loss in this case continues to decrease until it reaches a low value, but at the same time that of the training continues to increase to higher levels.

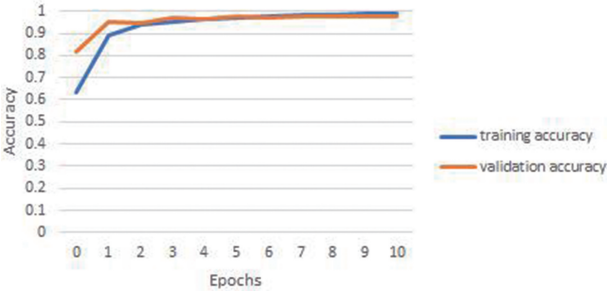

Fig. 12 Training and validation accuracy vs training epochs (3D-CNN).

Our trained 3D-CNN gesture classifier is considered as a good fit model based on the obtained loss/accuracy curves. We can see that our model did not experience a blatant case of overfitting. Validation loss reached its lowest value of 0.074 at the 5th epoch where the validation accuracy was at 0.977. However, training loss continued decreasing to reach a minimum of 0.039 at the 10th epoch with a training accuracy of 0.989.

3.4 Dynamic Hand Gesture Recognition with LRCN

This model proposed by Donahue in 2016 represents a Long-term Recurrent Convolutional Network (LRCN) which combines a deep hierarchical visual feature extractor (such as a CNN) with an LSTM

model that can learn to recognize and synthesize temporal dynamics for tasks involving sequential data, visual, linguistic, or otherwise[16]. The reported state-of-the art results in this paper on three vision problems (activity recognition, image description and video description) made Long-term Recurrent Convolutional Network one of the approaches we considered to solve our problem of dynamic hand gesture recognition. The steps of the LRCN model training are detailed in the following subsections.

3.4.1 Model Architecture

CNNs have been proved powerful in image related tasks like computer vision, image classification, and object detection. LSTMs are used in modelling tasks related to sequence-based predictions. LSTMs are widely used in NLP related tasks like machine translation, sentence classification and generation. LRCN, also known as CNN-LSTM model, was specifically designed for sequence prediction problems with spatial inputs, like images or videos. As shown in Fig.13, we trained an LRCN network on the same gesture dataset used for training our 3D-CNN model by feeding 30 input frames representing one hand gesture to a feature extractor layer (CNN), and combined it with LSTMs to support the sequence prediction.

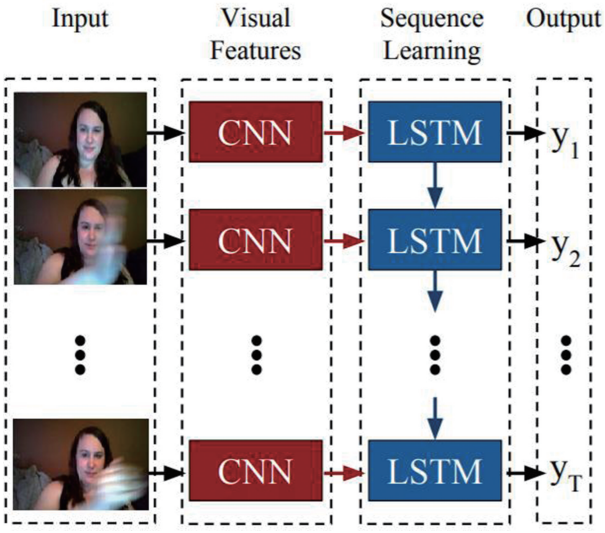

Fig. 13 Long-term Recurrent Convolutional Network (LRCN) architecture.

3.4.2 Model Training

The training of the LRCN classifier was performed using the same computer specifications (GPU, RAM, etc,) used for training the 3D-CNN classifier and on the same training/validation dataset. We used the Adam optimizer with a learning rate of 1e-5, decay of 1e-6 and applied a dropout of 0.5 before the LSTM layer for dimensionality reduction. The total duration of the training using a batch size of 6 was ~36 hours, which is more than 3 times longer than the training duration of the 3D-CNN classifier due to the much slower training speed of LSTMs[17]. It went through a total of 37 epochs before the model started to converge. We examined the training and validation loss curves, shown in Fig. 14 and Fig. 15, when the network was being trained, and we observed that the validation loss stopped decreasing after the 23rd epoch to reach a minimum of 0.248 at the 32nd epoch and then started increasing again until the early stopping mechanism triggered to stop the training, 5 epochs later.

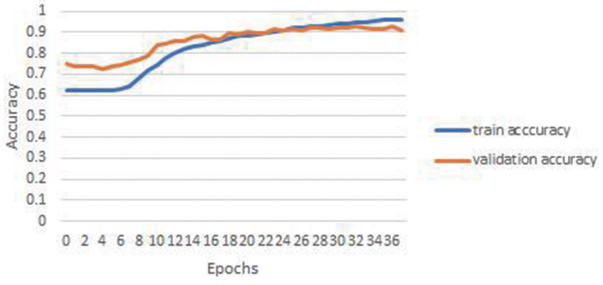

Fig. 14 Training and validation loss vs training epochs (LRCN).

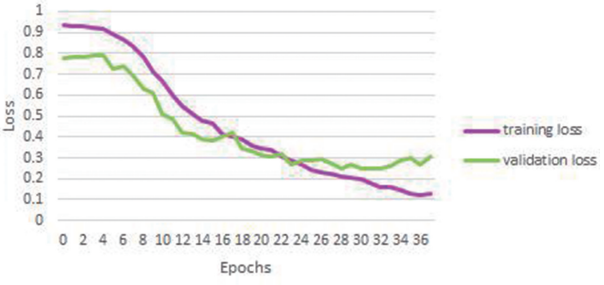

Fig. 15 Training and validation accuracy vs training epochs (LRCN).

In comparison with the 3D-CNN classifier training, a much lower validation loss of 0.074 was reached in much shorter amount of time. At the same time, validation accuracy reached a maximum of 0.927 at the 36th epoch compared to 0.977 for 3D-CNN. Based on the LRCN training data analysis, the obtained model can be considered as a good fit model since no remarkable overfitting symptoms are observed. It is still difficult at this stage to draw conclusions about the model that allows the best performance on our task of dynamic hand gesture recognition. Therefore, in the following section, we will present our evaluation results of both models tested on our evaluation dataset.

3.5 Experimental Results and Discussion

This section is devoted to the presentation of the experimental results relating to the two models introduced in sections 3.3 and 3.4, namely the 3D-CNN model and the LRCN model. As mentioned earlier, one of the most important motivations behind the introduction of these two models is their generalisation capacity in comparison with other approaches in literature. The experiments therefore were carried out on dynamic hand gesture recognition task using our evaluation dataset. In addition, the experimental results presented in this section will serve to evaluate the performances of the two model, and to compare the proposed 3D-CNN model to the state of the art on the studied issue.

Table 2 shows the performance of two different classifiers on each class. The results show that the 3D-CNN classifier is dominating LRCN on the three classes (Swiping Hand Left, Swiping Hand Down and Doing Other Things). The performance of the two classifiers is similar on the average precision, especially for the Swiping Hand Left and Swiping Hand Down classes, than that on recall and F1-measure. Furthermore, the table shows that while the LRCN classifier achieved a high precision for Swiping Hand Left and Swiping Hand Down classes that is comparable to that of the 3D-CNN, we notice that a relatively poor recall on the same two classes; which means the LRCN system classifies more samples into Doing Other Things, hence the high recall value for the latter class and poor precision. Going back to our use case in the present project: a gesture recognition self-parking system, both metrics, precision and recall, are important; we wish to achieve a high precision on the Swiping Hand Left (Park In) and Swiping Hand Down (Park Out) classes, but most importantly a high recall value to give a better user experience to the driver using the system while avoiding cases where the driver needs to repeat the hand gesture many times to trigger the parking action.

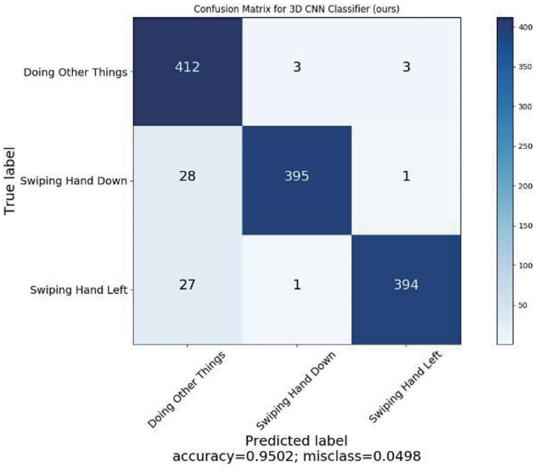

Fig. 16 Confusion matrix for 3D-CNN Gesture Classifier (ours).

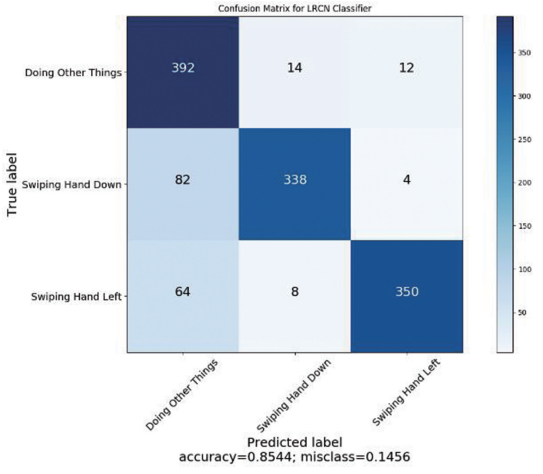

Fig. 17 Confusion matrix for LRCN gesture classifier.

The confusion matrices on the validation setare

given in Fig. 17. The overall accuracy of the LRCN classifier is 0.8544 whereas that of our 3D-CNN classifier is at 0.9502. The confusion matrix for LRCN shows clearly that many samples of Swiping Hand Left and Swiping Hand Down are classified as Doing Other Things, hence the poor recall noticed earlier. As expected, the LRCN model performed poorer than 3D-CNN. A possible explanation is that the position of the hand in each of the 30 frames will differ from sample to sample, which leads to feeding the LSTM with different positions of the hand in the respective indices of the 30 frames. This could confuse the LSTM ,which is reflected in the the number of false negatives.

Table 2 Per class performance comparison between LRCN and 3D-CNN.

Class	LRCN			3D-CNN (ours)		
	Precision	Recall	F1-Measure	Precision	Recall	F1-Measure
Swiping Hand Left (Park In)	0.96	0.83	0.89	0.99	0.93	0.96
Swiping Hand Down (Park Out)	0.94	0.80	0.86	0.99	0.93	0.96
Doing Other Things	0.73	0.94	0.82	0.88	0.99	0.93
Average	0.88	0.85	0.86	0.95	0.95	0.95

3.6 Transfer Learning and Final Model Fine-tuning

One of the big challenges in machine learning applications is that training data can be slightly different from the real-world data faced by the algorithm. Hence, the performance of the end-to-end system, once faced with real data, may not be at the desired level. We noticed that the trained 3D-CNN classifier did not perform well when tested live. Two main factors had a major impact on the performance of our system:

- Driver To Camera Distance: The closer the driver is to the multiview camera, the higher is the accuracy of the system. A camera distance within a range of [60cm, 110cm] produced the best performance., whereas, in the real-world scenario, the car driver would have a distance of at least 2 meters from the car to trigger the auto parking.

- Height of the Multiview Camera System: We noticed also during our end-to-end testing of the system that the camera system needs to be at a certain height (slightly lower than the user) in order to achieve the best system performance and accuracy. Once we place the multiview camera at the same level or slightly higher than the user, gesture detection accuracy starts to degrade.

The previous observations can be explained by the nature of the 20BN-Jester dataset we used to train our gesture model. In fact, most of the video samples in 20BN-Jester dataset are taken using a laptop or desktop computer webcam placed at a relatively close distance (50 to 100cm) and slightly lower level from the user. Hence, the sensitivity of our end model to those factors. In order to overcome these limitations, enhance the system performance and end user experience, we fine-tuned our model using transfer learning techniques. The following section describes the contingency steps that were taken to overcome the aforementioned challenges.

A dataset containing generalized gesture videos that are relevant to our use case of autonomous parking was not available. Therefore, we collected a second dataset in our lab and used data augmentation techniques to generate more data samples. The created dataset consists of a reasonable quantity of videos where many subjects performed the Swiping Hand Left and Swiping Hand Down gestures at variable distance from the camera system and at different setup heights. On the background of the user, we placed a green screen that enabled us to use the Chroma Keying technique (aka green screen keying, used for decades in film studios by placing human

characters in outerworldly situations without them having to leave the studio) to create new videos using parking lot backgrounds, reflecting the real-world scenario of parking situations, and various other backgrounds for data augmentation purpose. Fig. 18 shows the process we followed to generate the new dataset using Chroma Keying.

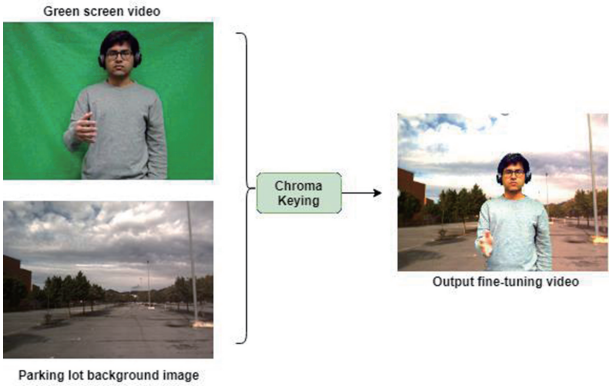

Fig. 18 Data generation using Chroma Keying.

The early layers of the 3D-CNN network already trained with the 20BN-Jester large dataset can extract generic features, so we used methods that further tunes a pre-trained model. Given the relatively small dataset (220 videos) compared to the 20BN-Jester dataset, we did only fine-tune the last layer of our 3D-CNN, which enhanced significantly the classification performance. The evaluation metric for live tests was empirical and based on the automaker satisfaction of the performance. After the dataset augmentation, the automaker reported a twice as good performance.

4 Deployment on an Embedded Platform

The person detection framework in the present development has been able to operate in real-time together with dynamic hand gesture recognition classifier on an NVIDIA Jetson TX2 board. The live inference software was implemented to run on the Jetson platform using the GPU-accelerated version of Tensorflow framework. One of the first challenging steps was the installation of the econ hexcam six cameras system on the Jetson TX2 board, including setting up of the drivers, which were only compatible with an older version of L4T r27v1.0 (the operating system of the NVIDIA Jetson board). Much effort was required to run that hardware with a newer version of CUDA and CUDNN libraries, as required by the person detection and gesture recognition modules. Once the six cameras system was installed and functional, the next challenge was to run both the person detection module, which includes loading the Darknet model (object detection network) and YOLOv3 network configuration, and our 3D-CNN hand gesture recognition model. Due to the limited available RAM on the Jetson board (8GB), we faced many issues in running the end-to-end system where the GPU was running out of memory while loading the 3D-CNN model (1.1GB) into memory. To overcome these issues, the following optimization techniques were applied:

- Optimize model saving: Keras offers different model saving methods. The most common method, which we used during our initial test, is *model.save*(). This method saves the architecture of the model allowing to re-create it, the weights of the model, as well as the training configuration (loss, optimizer) and the state of the optimizer. This resulted in a heavy model of 1.1GB. To significantly reduce the memory footprint of the 3D-CNN model, we used *model.to_json*() to make a JSON string of the architecture and save it, and *model.save_weights*() to make a separate file containing the weights. The resulted files were considerably smaller at 379.2MB, which is about third the size of the full *model.save*() result.

- Use Half-precision floating-point format (aka FP16): This consists of setting the format to 16-bit to represent the network weights, as opposed to the standard 32-bit floating point, or FP32. This allowed us to reduce the memory by cutting the size of our tensors in half without sacrificing the accuracy of the model.

- Tune Tensorflow GPU configuration: By default, TensorFlow maps nearly all of the GPU mem-

ory of all GPUs visible to the process. This is done to more efficiently use the relatively precious GPU memory resources on the devices by reducing memory fragmentation. On way to optimize memory allocation, is to limit the percentage of the overall size of memory that Tensorflow process can allocate. In our case, we set that limit to 25%, which reduced significantly the initial memory allocation requirements.

The present paper's use case is a multiview hand gesture recognition system for autonomous car parking; therefore, we mounted the Jetson TX2 board on a tripod at a height comparable to a mid-size car roof height (~180cm) as shown in Fig. 19. Mounting our system on a vehicle was out of the scope of the present project. Hence, we created an animated web application that our demo application triggers via HTTP requests based on the output of the inference program. The processing time for the end-to-end inference (person detection + hand gesture classification) was on average ~2 seconds. The main portion of that latency comes from the person detection and cropping part. The gesture recognition took less than 1 second during our tests.

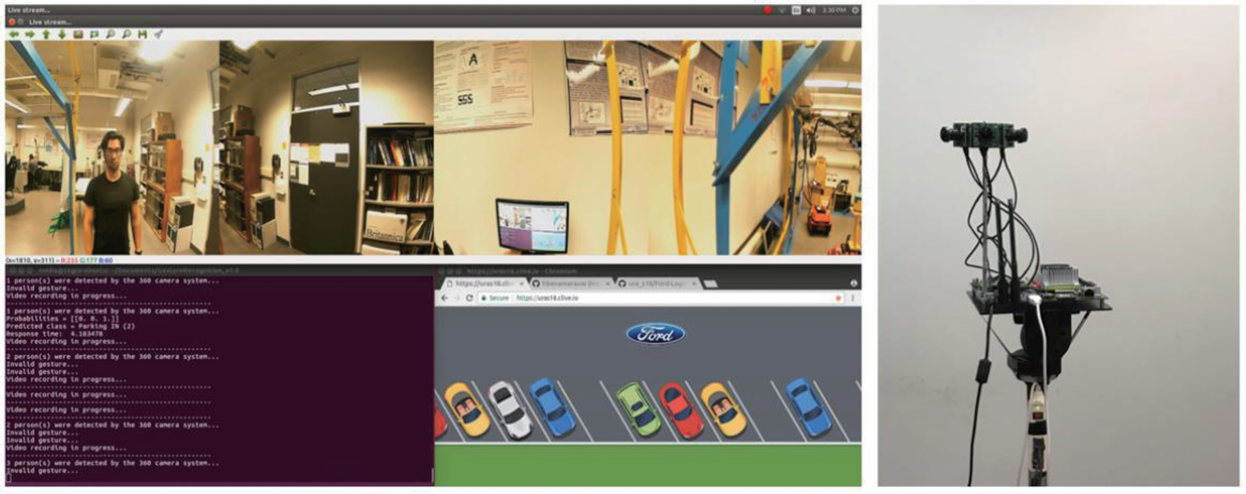

Fig. 19 A snapshot of the experimental results from a live demo.

5 Conclusion

The present paper proposed a vision-based multiview dynamic hand gesture recognition system and its application to vehicle self-parking. In an effort to develop the next generation of vehicle self-parking feature, we partnered with one of America's big three automakers to prototype a robust end-to-end system that operates in real world environment. Our main motivation for this paper was to eliminate the intermediate medium between the car and the driver to offer a friendly user interface for the self-park feature. To achieve the aforementioned attribute, we solely relied on a vision-based gesture recognition solution. The most comprehensive available database to train a dynamic hand gesture recognition classifier was 20BN-JESTER[13]. For simplicity, the developed feature had two commands, represented by two hand gestures "swiping hand left" and "swiping hand down". Hence, we filter the comprehensive "20BN-Jester" dataset to only two classes with an additional "doing other things" class, which was represented by a randomly selected non-command hand gestures.

As a first step to recognizing the hand gesture, the solution required a person detection algorithm at the front of the pipeline. We researched multiple alternatives and selected the best performing model, in this case, YOLO. YOLO, paired with a six-camera-

based sensor offered a pre-processing module that detected and identified all instances of "persons" in the 360 view of the vehicle. Each of the identified objects (person present within the vehicle field of view), was then processed by a 3D-CNN classifier. The latter was a multi-class classification, which mapped the first hand gesture (swiping hand left) to the part-in command and the second hand gesture (swiping hand down) to the park-out command. A third class (garbage model) was necessary to capture any gesture that did not fall in the aforementioned buckets. We achieved an accuracy of 95.02% with the selected 3D-CNN, while alternatives such as LRCN, performed at a maximum accuracy of 85.44%. The reported results were based on experiments ran in a laboratory environment. Once tested in a real world setting, we noticed a significant drop in accuracy, due to the varying distance and height of the user with respect to the camera. This was expected, as all of the training dataset consisted of laptop/webcam collected videos which implied a limited range of distance and height of the person performing the gesture. To overcome this limitation of our training data, we decided to leverage transfer learning and collect custom made data that would generalize our model on different backgrounds, distances and heights of the classified subject.

This end-to-end solution was developed as the vehicle for a host. Hence, multiple optimization techniques were applied to ensure the resulting model would operate in real-time on an embedded platform (NVIDIA Jetson TX2): less than 2 sec of processing for one hand gesture command, which was considered as a successful real-time implementation.

Future work will address some of the shortcomings of the proposed solution and how it can be expanded to cover more use cases. In order to overcome the limitation of training data with regard to varying distances and heights, one approach would be to add a pre-processing step that uses the body skeleton to locate the hand and then zoom in/out the camera or the 3D object according to how far the subject performing the hand gesture is. This would enhance the gesture classifier robustness while rendering the data augmentation step an additional enhancement.

The LRCN classifier performed poorly on the training dataset due to the different hand position in each of the 30 frames from sample to sample, which led to feeding the LSTM with different positions of the hand in the respective indices of the 30 frames. We proposed to normalize the full hand gesture over the 30 frames instead of normalizing the sequence (video). This would require an additional module to detect the start and end of the gesture in the video sequence.

As the vehicle for a host, a cost-effective alternative to our embedded gesture recognizer could make use of cloud deployed centralized gesture classifier. In this case, the vehicle would make an online prediction request, assuming that the car is Internet-connected. The next step towards a market-ready solution, would be the deployment of our system on a vehicle. Using technologies like Bluetooth proximity and key fob smartphone app, a future optimization could reduce the person detection overhead, especially in crowded environments, by orienting the camera focus towards the car owner's direction, thereby significantly reducing the input size of the gesture classifier.

References

[1] Grobel, K. & Assan, M. (1997), Isolated sign language recognition using hidden Markov models, in ´ Systems, Man, and Cybernetics, 1997.

[2] Computational Cybernetics and Simulation., 1997 IEEE International Conference on´, pp. 162--167.

[3] Starner, T.; Weaver, J. & Pentland, A. (1998), ´ Real-time american sign language recognition using desk and wearable computer-based video ´, IEEE Transactions on pattern analysis and machine intelligence 20(12), 1371--1375.

[4] Xu, D. (2006), A neural network approach for hand gesture recognition in virtual reality driving training system of SPG, in 'Pattern Recognition, 2006. ICPR 2006. 18th International Conference on´, pp. 519--522.

[5] Reifinger, S.; Wallhoff, F.; Ablassmeier, M.; Poitschke, T. & Rigoll, G. (2007), Static and dynamic hand-gesture recognition for augmented reality applications, in 'International Conference on Human-Computer Interaction', pp. 728--737.

[6] Wabel, A. & Vo, M. (1993), A multi-modal human-computer interface: Combination of gesture and speech recognition, in 'Proc. of the Int. Conf. on Human Factors in Computing Systems (CHI)'.

[7] Kumar, A. & Zhang, D. (2006), 'Personal recognition using hand shape and texture', IEEE Transactions on image processing 15(8), 2454--2461.

[8] Insights, G. M. (2019), 'Automotive Gesture Recognition Market to exceed 13billion by 2024', https://www.gminsights.com/pressrelease/automotive-gesture-recognition-market.

[9] Web (2018), 'BMW customer report', https://www. autonews. com/article/20180702/RETAIL01/180709977/car-sales-on-pace-to-hit-60-year-low.

[10] Web (2016), 'How Tesla and Nissan's Self-Parking Cars work', http://fortune. com/2016/01/12/tesla-nissan-self-parking/.

[11] Information, B. M. W. M. (2016), 'BMW at the Consumer Electronics Show (CES) 2016 in Las Vegas.', https://www.bimmerpost.com/goodiesforyou/autoshows/ces2016/bmw-ces-2016.pdf.

[12] Ren, S.; He, K.; Girshick, R. & Sun, J. (2015), Faster r-cnn: Towards real-time object detection with region proposal networks, in 'Advances in neural information processing systems', pp. 91--99.

[13] Redmon, J.; Divvala, S.; Girshick, R. & Farhadi, A. (2016), You only look once: Unified, real-time object detection, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 779--788.

[14] TwentyBN (2018), 'The 20BN-jester Dataset V1.', https://20bn.com/datasets/jester.

[15] Molchanov, P.; Gupta, S.; Kim, K. & Kautz, J. (2015), Hand Gesture Recognition With 3D Convolutional Neural Networks, in 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops'.

[16] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', The Journal of Machine Learning Research 15(1), 1929--1958.

[17] Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K. & Darrell, T. (2015), Long-term recurrent convolutional networks for visual recognition and description, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2625--2634.

[18] Chen, X.; Liu, X.; Gales, M. J. F. & Woodland, P. C. (2015), Improving the training and evaluation efficiency of recurrent neural network language models, in 'Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on', pp. 5401--5405.

Authors' Biographies

Hassene Ben AMARA received his Bachelor's degree from National Engineering School of Carthage. Now he is a Master's degree candidate at CPAMI in the University of Waterloo. His main research interest lies in the field of machine learning with focus on deep learning and computer vision.

E-mail: hassene.benamara@gmail.com

Fakhri KARRAY, PhD, P. Eng., FIEEE, FCAE, FEIC is the University Research Chair Professor in Electrical and Computer Engineering and the founding co-director of the Institute of Artificial Intelligence at the University of Waterloo. He holds the Loblaw's Research Chair in Artificial Intelligence at the University of Waterloo and is the Director of the University's Center for Pattern Analysis and Machine Intelligence (CPAMI). Karray's current research interests are in the areas of intelligent systems design, artificial intelligence, concept mining, big data analytics, machine learning, soft computing, sensor fusion, and context aware machines with application to intelligent transportation systems, Internet of things, cognitive robotics, healthcare enhancement for the elderly and the disabled. He is the Chair of the *IEEE Computational Intelligence Society* Chapter in Kitchener-Waterloo, Canada and chaired various committees of the IEEE Computational Intelligence Society. He is a Fellow of: IEEE, the Canadian Academy of Engineering, and the Engineering Institute of Canada.

E-mail: karray@uwaterloo.ca