UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Gabriel Andrade Nunes de Moraes

Image-Based Real-Time Path Generation Using Deep Neural Networks

Vitória, ES
May 26, 2022

Gabriel Andrade Nunes de Moraes

Image-Based Real-Time Path Generation Using Deep Neural Networks

Thesis submitted to the *Programa de Pós-Graduação em Informática* of the *Centro Tecnológ*ico at the *Universidade Federal do Espírito Santo*, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Advisor: Prof. Dr. Claudine Santos Badue
Co-advisor: Prof. Dr. Alberto Ferreira De Souza

Vitória, ES
May 26, 2022

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Moraes, Gabriel Andrade Nunes de, 1995-
M827i Image-based real-time path generation using deep neural networks / Gabriel Andrade Nunes de Moraes. - 2022.
44 f. : il.

Orientadora: Claudine Santos Badue.
Coorientador: Alberto Ferreira de Souza.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Veículos autônomos. 2. Redes neurais (Computação). I. Badue, Claudine Santos. II. Souza, Alberto Ferreira de. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Image-Based Real-Time Path Generation Using Deep Neural Networks

Gabriel Andrade Nunes de Moraes

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 26 de maio de 2022.

Prof. Dr. Claudine Santos Badue
Orientador, participação remota

Prof. Dr. Alberto Ferreira De Souza
Coorientador, participação remota

Prof. Dr. Thiago Oliveira dos Santos
Membro Interno, participação remota

Prof. Dr. Fernando Santos Osório
Membro Externo, participação remota

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória/ES, 26 de maio de 2022

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
CLAUDINE SANTOS BADUE - SIAPE 1729561
Departamento de Informática - DI/CT
Em 27/05/2022 às 10:26

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/483618?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
THIAGO OLIVEIRA DOS SANTOS - SIAPE 2023810
Departamento de Informática - DI/CT
Em 27/05/2022 às 11:07

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/483661?tipoArquivo=O

AGRADECIMENTO

À minha família, pelo apoio incondicional e sempre presente.

À minha orientadora, Claudine, por não poupar esforços em me ajudar ao longo de toda a jornada.

Ao meu coorientador, Alberto, pela inspiração e conhecimento que transmite com tanta facilidade.

À minha namorada, por toda a tranquilidade que me trouxe e traz, mesmo que ao custo da sua própria.

Aos outros professores e alunos do LCAD, pelo companheirismo em níveis que eu não esperava encontrar.

Aos meus amigos, por serem quem são, essas pessoas sem as quais não sei imaginar minha vida.

À CAPES pelo financiamento, imprescindível ao meu trabalho e a tanto do que esteve ao meu entorno na pesquisa.

RESUMO

No presente trabalho, foi proposto para o carro autônomo *Intelligent Autonomous Robotic Automobile* (IARA) um planejador de caminho em tempo real, baseado em imagens, chamado de *DeepPath*. O DeepPath usa uma rede neural convolucional (*convolutional neural network* - CNN) para inferir caminhos a partir de imagens. Durante a operação do carro autônomo, o *DeepPath* recebe uma imagem do ambiente imediatamente à frente do veículo e a pose atual do carro. Em seguida, ele envia a referida imagem para uma CNN treinada, de forma a inferir um modelo do caminho que deverá ser percorrido pela IARA. Depois disso, o *DeepPath* gera o caminho no sistema de coordenadas da IARA usando o modelo de caminho inferido. Posteriormente, dada a pose atual da IARA, o *DeepPath* transforma cada pose do caminho, originalmente no sistema de coordenadas da IARA, em outra pose no sistema de coordenadas mundial. Finalmente, ele envia o caminho para o subsistema Seletor de Comportamento, o próximo subsistema no sistema de tomada de decisão da IARA. Avaliamos o desempenho do *DeepPath* em cenários do mundo real. Nossos resultados mostraram que o *DeepPath* é capaz de gerar corretamente caminhos para a IARA que diferem apenas ligeiramente daqueles definidos por humanos.

ABSTRACT

We propose an image-based real-time path planner for the self-driving car Intelligent Autonomous Robotic Automobile (IARA), named DeepPath. DeepPath uses a convolutional neural network (CNN) for inferring paths from images. During the self-driving car operation, DeepPath receives an image and the current car pose. Then, it sends the image to a CNN trained to infer a model of the path. After that, DeepPath generates the path in the IARA's coordinate system using the path model. Subsequently, given the current IARA's pose, DeepPath transforms each pose of the path in the IARA's coordinate system into another pose in the world coordinate system. Finally, it sends the path to the IARA's Behavior Selector subsystem, the next subsystem in the IARA's Decision-Making system. We evaluated the performance of DeepPath in real world scenarios. Our results showed that DeepPath is able to correctly generate paths for IARA that differ only slightly from those defined by humans.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVATIONS AND ACRONYMS

CARMEN	Carnegie Mellon Robot Navigation Toolkit
CNN	Convolutional Neural Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
IARA	Intelligent Autonomous Robotic Automobile
IMU	Inertial Measuring Unit
IP	Internet Protocol
IPC	Inter Process Communication
LCAD	*Laboratório De Computação de Alto Desempenho* (High Performance Computing Laboratory)
LIDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MOT	Moving Objects Tracking
MSE	Mean Squared Error
RMS	Root Mean Square
RTK	Real Time Kinematic
TCP	Transmission Control Protocol
TSD	Traffic Signalization Detection
UFES	*Universidade Federal do Espírito Santo* (Federal University of Espírito Santo)

1 INTRODUCTION

The architecture of the autonomy system of self-driving cars is typically organized into two main parts: the perception system and the decision-making system [PAD16]. Fig. 1 shows a block diagram of the typical architecture of self-driving cars, where the perception and decision-making systems are shown as a collection of subsystems of different colors.

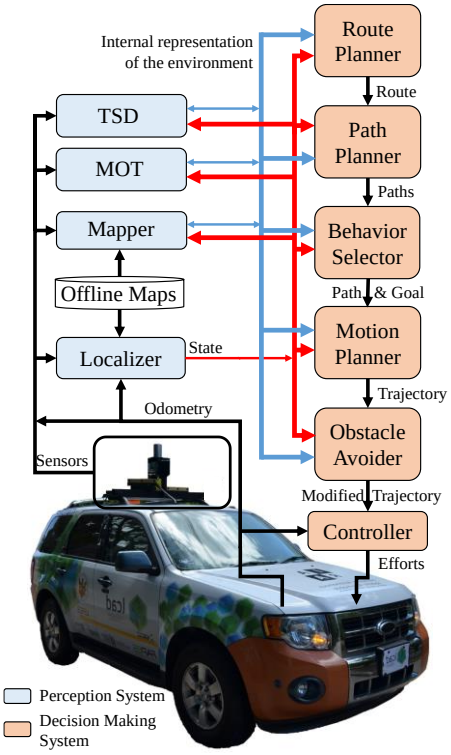

Fig. 1. Overview of the typical architecture of self-driving cars [BAD19]. TSD denotes Traffic Signalization Detection and MOT Moving Objects Tracking. [Fonte: Moraes et al. (2020)]

The perception system is responsible for creating an internal representation of the world and is generally divided into many subsystems responsible for tasks such as: self-driving car localization in a set of previously built offline maps, online static obstacles mapping, online road mapping, moving obstacles detection and tracking, and traffic signalization detection and recognition, among others. The decision-making system is responsible for navigating the car from its initial pose to a final goal pose and is commonly partitioned as well into many subsystems responsible for tasks such as: route planning in offline maps, path planning, behavior selection, motion planning, obstacle avoidance and control, though this partitioning is somewhat blurred and there are several different variations in the literature [PAD16].

Given the initial pose of the self-driving car and a final goal pose defined by a user operator, the route planner subsystem generates a route, W, in offline maps through a road network from the initial car's pose to the final goal pose [BAS15]. A route is a sequence of waypoints, $W = \{w_1, w_2, \dots, w_i, \dots, w_{|W|}\}$, where each waypoint, w_i, is a coordinate pair, $w_i = (x_i, y_i)$, in the offline maps. Given the route, the path planner subsystem generates a path, P, considering the current car's state and the internal representation of the environment, as well as traffic rules [PAD16] [GON16]. A path is a sequence of poses, $P = \{p_1, p_2, \dots p_i, \dots, p_{|P|}\}$, where each pose, p_i, is a coordinate pair, (x_i, y_i), in offline maps plus the desired car orientation, θ_i, at the position defined by the coordinate pair, i.e., $p_i = (x_i, y_i, \theta_i)$.

We have developed a self-driving car, named Intelligent Autonomous Robotic Automobile (IARA, Fig. 1), whose autonomy system follows the typical architecture of self-driving cars [BAD19]. IARA is based on a Ford Escape Hybrid adapted with a variety of sensors and processing units. Its autonomy system is composed of many subsystems, which includes a Mapper [MUT16], a Localizer [VER16], a Moving Obstacle Tracker [SAR19], a Traffic Signalization Detector [POS19] [TOR19], a Route Planner, a Path Planner, a Behavior Selector, a Motion Planner [CAR17], an Obstacle Avoider [GUI16] and a Controller [GUI17], among others.

In this work, we present a new image-based real-time path planner for the self-driving car IARA, named DeepPath (Fig. 2). DeepPath uses a deep Convolutional Neural Network (CNN) for inferring paths from images. While the self-driving car is in operation, DeepPath receives an image, captured from the IARA's front camera, and the current pose of IARA, computed by the IARA's Localizer subsystem [VER16]. Then, it sends the image to a CNN trained to infer a model of the path, which consists of a displacement on the y-axis of the IARA's coordinate system plus a set of three knots that specify a cubic spline. After that, DeepPath generates the path in the IARA's coordinate system using the path model. Subsequently, given the current IARA's pose in the world coordinate system, DeepPath transforms each pose of the path in the IARA's coordinate system into another pose in the world coordinate system. Finally, it sends the path to the IARA's Behavior Selector subsystem (Fig. 1), the next subsystem in the IARA's Decision-Making system.

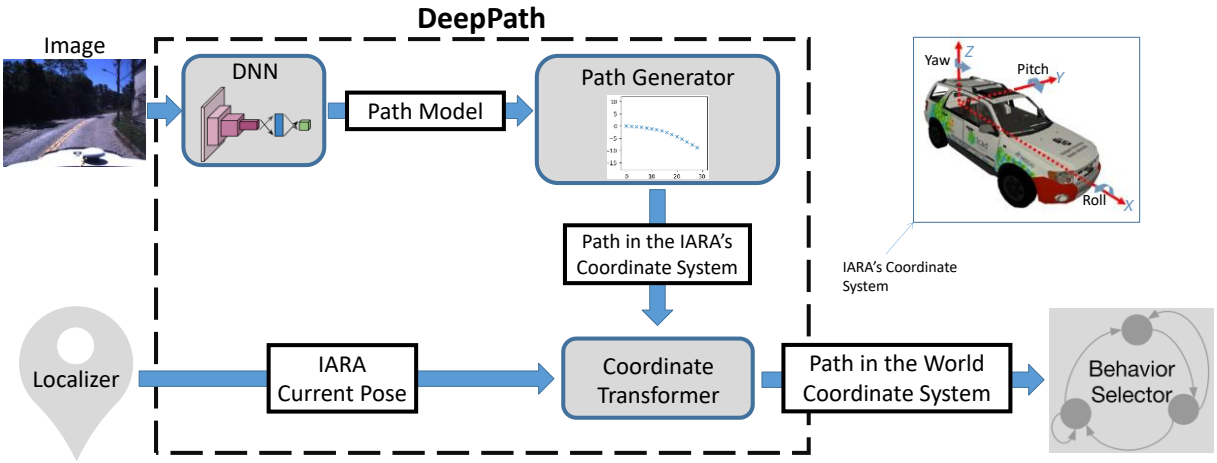

Fig. 2. Overview of DeepPath

We evaluated the performance of DeepPath in real world scenarios. For that, we used DeepPath for path planning along the ring road of the *Universidade Federal do Espírito Santo* (UFES) campus, which has 3.7 km of extension. Our experimental results showed that DeepPath is able to correctly generate paths for IARA that differ only slightly from those defined by humans – the Root Mean Square (RMS) of the differences between the poses of the paths estimated from images by DeepPath and those of the desired paths (paths followed by human drivers) is 0.37 m, on average.

1.1 Motivation

The current IARA path planning subsystem works as follows. In an offline phase, a human driver conventionally drives IARA through a course of interest from start to finish. During this travel, the IARA localization subsystem records the poses of the car. In the autonomous operation phase along the route of interest, the path planning subsystem produces a short-term path corresponding to the lane segment immediately ahead of the car at any given time, which is composed of a set of car's poses that are regularly spaced by 0.5 m and extracted from the poses recorded in the offline phase.

A limitation of the current IARA path planner is the need for a human driver to travel a course before that course can be travelled autonomously. In this work, we propose a path planner, called DeepPath, that tries and solves this limitation as follows. Using a neural network trained for the task of planning a path, DeepPath determines a short-term path (associated with the lane segment immediately ahead of the car at a given time) using real-time camera images

as input. Although DeepPath requires a map of the surroundings the car's localization, such a map can be built by driving on any trajectory along the road of interest, for example using the opposite lane.

1.2 Objective

The main objective of this work is to propose and develop a path planning subsystem based on CNNs for self-driving cars, which is compatible with the IARA autonomy system and operates in real-time. It is also the objective of this work to evaluate the performance of the proposed path planner through tests in real world scenarios.

To fulfill the objectives of this work, we first chose and adapted a neural network architecture to the task of planning a path. Next, we built a dataset to train the neural network, using sensor data collected by IARA. In addition, we implemented the interface between the neural network and the other modules of the IARA autonomy system. Finally, we carried out experiments in real world scenarios to evaluate the performance of the proposed path planner already integrated into the IARA autonomy system.

1.3 Contribution

The main contributions of this work are:

1) Proposal of a path planning subsystem based on CNNs for self-driving cars, named DeepPath, which is compatible with IARA's autonomy system and operates in real-time;
2) Development of DeepPath and its integration into the IARA's autonomy system;
3) Construction of a new dataset composed of camera images of the road and associated pose sequences for the road segment immediately ahead of the car;
4) Evaluation of the performance of DeepPath through a series of experiments using sensor data collected from IARA;
5) Evaluation of DeepPath on the autonomous operation of IARA.

1.4 Organization

This thesis is structured as follows. After this introduction, in Chapter 2, we present related works. In Chapter 3, we detail DeepPath. In Chapter 4, we describe the experimental methodology adopted to evaluate DeepPath and, in Chapter 5, we discuss experimental results. Finally, in Chapter 6, we close with conclusions and directions for future work. Chapter 7 references the scientific paper published during the development of this work.

2 RELATED WORK

There are various methods proposed in the literature to address the problem of path planning for self-driving cars. For reviews on these methods, readers are referred to González et al. [GON16], Paden et al. [PAD16] and Badue et al. [BAD19].

Among those based on CNNs, some methods infer paths from images. Guo et al. [GUO17] proposed a path planning method that uses a behavior-induction potential map to generate a path from a front-view camera image. The proposed method is intended to imitate the following mechanism: when a human driver sees a vehicle ahead on the road, he does not consider the distance he must keep from it, but what path he must take to interact with it. In the proposed method, a CNN is adopted to detect vehicle candidates in an image. The bounding box candidates are then merged based on both the image evidence and the statistical support of candidates. After that, the detected vehicles are classified into six categories by a Bayesian Network. Finally, a behavior-induced potential map is constructed in which a mass-spring-damper system with a cubic spline is employed to generate a path for following a predefined route. Simulations with videos collected in urban environments were conducted to evaluate the performance of the behavior-induced potential map on path generation. Simulation results showed that paths generated by the proposed method are close to those by human drivers.

Reh et al. [REH17] proposed a path planning technique that employs a CNN to infer a path from an aerial image. To build training and test datasets, a set of aerial images was collected from Google Maps and annotated with paths followed by human drivers, which were observed from the road side of an intersection. Experimental results showed that the proposed method generates paths similar to those by human drivers.

In comparison to DeepPath, the methods mentioned above [GUO17] [REH17] involve statistical models or aerial images, while DeepPath requires only front-view camera images and car poses.

Other methods infer motion commands (such as steering wheel angle, speed and lateral/longitudinal positions) from front-view camera images. In fact, these methods address the tasks of path planning and motion planning [BAD19] at the same time. Bojarski et al. [BOJ16] proposed a motion planning method that uses a convolutional neural network (CNN) to infer a steering wheel angle from a single front-view camera image. Most of the training and test dataset was collected by driving on a variety of roads in central New Jersey. Experimental results

showed that the proposed method was able to drive on local roads with or without lane markings, and on highways.

Chen et al. [CHE17a] presented a motion planning technique similar to that of Bojarski et al. [BOJ16]. Its CNN was trained and tested using the comma.ai dataset [SAN16], which was collected by driving on roads. Test results showed that the presented technique can produce relatively accurate vehicle steering.

Yan et al. [YAN18] proposed a multi-modal multi-task CNN to predict a steering wheel angle and a speed simultaneously. The proposed model receives as input a stream of front-view images and a sequence of speeds. It was evaluated on the public Udacity dataset (https://github.com/udacity/self-driving-car) and on the newly collected SAIC dataset. Experimental results showed that the proposed model provides an accurate speed prediction, in addition to further improving the state-of-the-art steering wheel angle estimate.

Cai et al. [CAI19] presented a CNN Long Short-Term Memory (LSTM) to infer a trajectory composed of speeds and lateral/longitudinal positions, 3.0 seconds in the future. The presented model takes as input a stream of front-view images and a sequence of car states in the past 1.5 seconds. It was evaluated using training and test datasets extracted from the Robotcar dataset [MAD17]. Experimental results suggest that the presented model generates trajectories similar to the ground truth when turning at various intersections or keeping straight.

Compared to DeepPath, the methods mentioned above [BOJ16] [CHE17a] [YAN18] [CAI19] address the tasks of path planning and motion planning simultaneously, while DeepPath handles path planning only. The reason is that DeepPath was designed for the self-driving car IARA, whose autonomy system deals with path planning and motion planning separately in different subsystems [BAD19].

3 CNN PATH GENERATION SYSTEM (DEEPPATH)

During the self-driving car operation, DeepPath receives as input an image from the IARA's front camera and the current IARA's pose from the IARA's Localizer subsystem, and generates as output a path (Fig. 2). It does so by using a CNN to infer a model of the path, which consists of four parameters: a displacement on the y-axis of the IARA's coordinate system plus a set of three knots that specifies a cubic spline. The model of the path is then transformed into an actual path by: (i) generating the path in the IARA's coordinate system and then (ii) transforming each pose of the path in the IARA's coordinate system into another pose in the world coordinate system.

In the following subsections, we describe this process in more details.

3.1 CNN Architecture

To infer a model of the path, DeepPath employs a modified version of the CNN for image segmentation proposed by Bulo et al. [WU19], pre-trained on the Mapillary Vistas dataset. The CNN architecture they proposed is composed of three sections. The first section (or body) of the network is the WideResNet38 architecture [CHE17b], which contains 17 residual blocks. The first 15 blocks contain 2 convolution operations each, while the last 2 blocks contain 3. The second section of the network is the DeepLabV3 architecture [MAA13]. It performs 4 parallel convolutions, with different dilation rates for exploring multiple scale contexts, and combines the results with a Spatial Pyramid Pooling operation [MAA13]. The innovation proposed by Bulo et al. [BUL18] was the replacement of all the Batch Normalization layers in those architectures by a new structure made of LeakyReLU neurons [THR01] with negative slope of 0.01, which performs an operation called In-Place Activated Batch Normalization. This modification allows the release of a significant amount of Graphics Processing Unit (GPU) memory, which can be exploited to scale up the batch size. Finally, the third (or final) section of the network performs bilinear up-sampling so that the output has the same dimensions as the input image of the network.

To repurpose this CNN to the regression task required by DeepPath (of inferring path model parameters from images) instead of image segmentation, we modified only the final

section of the network. We replaced this bilinear up-sampling layer by a fully connected layer with four linear neurons that output: the displacement on the y-axis of the IARA's coordinate system, and the set of three knots that specify a cubic spline.

3.2 Path Model

Fig. 3 shows a path and its parameters in the IARA's coordinate system. The displacement, dy, is the distance that IARA has to move on its y-axis to be exactly over the desired path, i.e., the path that the operator wants IARA to go through. The set of three knots, $K = \{k_1, k_2, k_3\}$, defines the cubic spline that best fits the desired path, $spline(., K)$, in a coordinate system that is identical to that of IARA, except by the fact that it is translated in the y-axis by the displacement, dy. The spline starts in the origin of this coordinate system and crosses the three knots at coordinates $(10, k_1)$, $(20, k_2)$ and $(30, k_3)$. The values 10, 20 and 30 are in meters and were chosen so that the paths generated by DeepPath would be long enough to allow safe autonomous operation at speeds of up to 5.56 m/s (or 20 km/h). These values can be altered if faster speeds are desired.

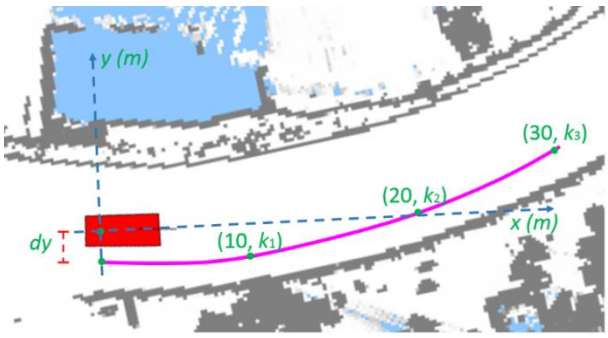

Fig. 3. Desired path and its parameters in the IARA's coordinate system (in meters) with an occupancy grid map in the background. The red rectangle indicates the current IARA's pose; the small dashed red line indicates the displacement, dy; the three green dots represent the three knots, k_1, k_2, k_3, of the cubic spline; and the purple curve represents the cubic spline that best fits the desired path.

3.3 Path Generation

DeepPath generates paths in the IARA's coordinate system using the path model. An estimated path in the IARA's coordinate system, P^e, consists of a sequence of poses, $P^e = \{p_1^e, p_2^e, \ldots, p_i^e, \ldots, p_{|P^e|}^e\}$, in which $p_i^e = (x_i^e, y_i^e, \theta_i^e)$. The car position at each pose p_i^e, (x_i^e, y_i^e),

is defined by the value of x_i^e, that can be any value between 0 and 30 m along the spline, and of $y_i^e = spline(x_i^e, K) + dy$, where dy is the displacement. The car orientation at each pose p_i^e, θ_i^e, is defined by the slope of the line that tangents the spline (or the derivative of the spline) at the position (x_i^e, y_i^e). It is approximated by the equation

$$\theta_i^e \cong \arctan\left(\frac{spline(x_i^e+\Delta x,K)-spline(x_i^e,K)}{\Delta x}\right), \quad (1)$$

where Δx is small.

3.4 Coordinate Transformation

Given the current IARA's pose in the world coordinate system, DeepPath transforms each pose of a path in the IARA's coordinate system, $p_i^e = (x_i^e, y_i^e, \theta_i^e)$, into another pose in the world coordinate system, $p_i^w = (x_i^w, y_i^w, \theta_i^w)$, using the equation

$$\begin{pmatrix} x_i^w \\ y_i^w \\ \theta_i^w \end{pmatrix} = \begin{pmatrix} \cos\theta_I & -\sin\theta_I & 0 \\ \sin\theta_I & \cos\theta_I & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i^e \\ y_i^e \\ \theta_i^e \end{pmatrix} + \begin{pmatrix} x_I \\ y_I \\ \theta_I \end{pmatrix}, \quad (2)$$

where $p_I = (x_I, y_I, \theta_I)$ is the IARA's pose in the world coordinate system at the time of the coordinate transformation, i.e., the time when the input image is captured.

4 METHODOLOGY

To examine the performance of DeepPath, we used the hardware and software infrastructure of the self-driving car IARA (Fig. 1). To train the CNN of DeepPath and test its performance, we built a series of training and test datasets, using data logged from IARA's sensors and computed by IARA's subsystems. We then assessed the quality of the paths generated by DeepPath by comparing them with the desired paths using proper metrics.

4.1 IARA

IARA (Fig. 1) is a self-driving car developed by the computational intelligence research group of the *Laboratório de Computação de Alto Desempenho* (LCAD) (http://www.lcad.inf.ufes.br) at the *Universidade Federal do Espírito Santo* (UFES), in Brazil. Details about its hardware and software are given below.

4.1.1 Hardware of the IARA`s Autonomy System

IARA is based on a Ford Escape Hybrid, which was modified to allow electronic control of steering, throttle, brakes, gears and several signalization items; and to provide the car odometry for the IARA's autonomy system, and power supply for computers and sensors. Its main computer is a Dell Precision R5500 with two Xeon X5690 six-core 3.4 GHz processors and one NVIDIA TITAN Xp. Its sensors include one Velodyne HDL 32-E LIDAR (Light Detection and Ranging), one Trimble RTK (Real Time Kinematic) GPS (Global Positioning System), one Xsens MTi IMU and one Bumblebee XB3 stereo camera.

4.1.2 Software of the IARA's Autonomy System

The IARA's autonomy system follows the typical architecture of self-driving cars [BAD19]. The subsystems of the IARA autonomy system (Fig. 1) are implemented as one or more software modules using the Carnegie Mellon Robot Navigation Toolkit (CARMEN).

Since 2011, CARMEN has been extended and maintained by LCAD at UFES, and made available to the public at https://github.com/LCAD-UFES/carmen_lcad. CARMEN-LCAD can be integrated into any type of vehicle platform. For that, it is necessary to adapt the platform to enable electronic vehicle control; provide the vehicle odometry to the autonomy system, and power for computers and sensors; and install sensors, e.g., LIDAR, GPS, IMU and cameras. CARMEN-LCAD has now been tested in cars (electric and combustion-powered), trucks and even in planes. For those tests, sensors were encapsulated in a box (called the sensor box) that was fixed to the roof of the vehicles.

4.1.3 CARMEN

CARMEN is a modular collection of software for controlling mobile robot. It was designed to provide basic navigation primitives, and to support several robots and sensors. It was created by Carnegie Mellon University (http://carmen.sourceforge.net/).

In CARMEN, communication between modules is done through the Inter Process Communication (IPC), where each module can publish messages and/or subscribe to messages of interest to the module and receive them. This exchange of messages can even happen between different computers using TCP/IP (Transmission Control Protocol/Internet Protocol). The messages are sent through a publication to the Central server, which is responsible for receiving the messages and passing them on to the modules that signed the specific message. Modules that publish messages are called Publishers, while modules that receive messages are called Subscribers. A module can be both at the same time.

4.2 Datasets

To train the DeepPath's CNN and test its performance, we built a series of training and test datasets, whose characteristics are summarized in Table 1. To build these datasets, we logged data from all IARA's sensors while IARA was being conducted by a human driver along the ring road of the UFES main campus. This ring road has 3.7 km of extension.

Table 1. Characteristics of the training and test datasets

Datasets	Sources of Datasets	Number of Images	Image Size
Training Dataset	Logs 1 to 4	31,911	640×480-pixel
Validation Dataset	Log 5	7,276	
Test Dataset 1	Log 6	6,260	
Test Dataset 2	Log 7	8,199	

We recorded seven sensor data logs, whose characteristics are summarized in Table 2. In the Logs 1 and 2, the driver tried and kept IARA centralized on the lane along the ring road (see Fig. 4) – in the Log 1, the driver travelled clockwise, while, in the Log 2, counterclockwise. In the Logs 3 and 4, the driver conducted IARA on a zigzag course, swinging to the left and to the right side of the center of the lane (see Fig. 5) – in the Log 3, the driver travelled clockwise, while, in the Log 4, counterclockwise. From the Logs 5 to 7, the driver tried and kept IARA centralized on the lane along the ring road–in the Logs 5 and 6, the driver travelled clockwise, while, in the Log 7, counterclockwise.

Fig. 4. Sample images from the logs in which IARA was driven centralized on the lane

Fig. 5. Sample images from the logs in which IARA was driven on a zig zag course along the road

Table 2. Characteristics of the sensor data logs

Logs	**Path Shape**	**Travel Direction**	**Desired Path Source**
Log 1	Centralized	Clockwise	Log 1
Log 2	Zigzag	Counterclockwise	Log 2
Log 3	Centralized	Clockwise	Log 1
Log 4	Zigzag	Counterclockwise	Log 2
Log 5	Centralized	Clockwise	Log 1
Log 6	Centralized	Clockwise	Log 1
Log 7	Centralized	Counterclockwise	Log 2

In all the seven sensor data logs, we computed and logged as well IARA's poses on the paths along the ring road using the IARA's Localizer subsystem. Finally, for each camera image in the seven logs, we estimated and logged the parameters of the model of a segment of the *desired ring road path* extending 30 m in front of the IARA's pose, at the time when each

camera image was captured, using the logged IARA's poses. In Section 4.3, we describe this process in detail.

We have two *desired ring road path* paths, one clockwise and one counterclockwise. They are the paths followed by the driver while he kept IARA centralized on the lane along the ring road in the Logs 1 and 2 mentioned above. So, desired paths for images of the Logs 3 and 4 were captured from data of the Logs 1 and 2, respectively. Desired paths for images from the Logs 5 and 6 were captured from data of the Log 1, while, for images from the Log 7, from data of the Log 2. The Logs 3 and 4 were necessary for teaching the CNN the cases in which the displacement, dy, is far from zero.

The dataset used to train the CNN (Training Dataset) was extracted from the Logs 1 to 4 (Table 2). It is composed of 31,911 images of 640×480-pixel and the parameters of its associated path models. The Validation Dataset was extracted from the Log 5. It is composed of 7,276 images of 640×480-pixel and the parameters of the associated path models. We built two test datasets, which were extracted from the Logs 6 and 7. The Test Dataset 1 is composed of 6,260 images of 640×480-pixel and the parameters of the associated path models, and the Test Dataset 2 is composed of 8,199 images of 640×480-pixel.

4.3 Generation of the Parameters of the Desired Path Model

As mentioned before, for each camera image in the seven logs, we estimated and logged the parameters of the model of a segment of the *desired ring road path* extending 30 m in front of the IARA's pose at the time when each camera image was captured.

To estimate the displacement, dy, we found the point in the *desired ring road path* closest to the current IARA's position, and computed the difference between the ordinates of this nearest point and of the current IARA's position; when the point is to the left of IARA, dy is positive and when it is to the right, negative.

To estimate the set of three knots, k_1, k_2 and k_3, we found the cubic spline with three knots that best approximates the segment of the *desired ring road path* closest to the current IARA's position. For that, we (i) transformed each point of the relevant segment of the *desired ring road path* in the world coordinate system into another point in the IARA's coordinate system, (ii) discounted the displacement, dy, of each one of its points and (iii) used the conjugate

gradient optimization algorithm to gradually change initial guesses for the set of three knots, so that the positions estimated using the spline were as close as possible to the points of the *desired ring road path* segment with the same abscissa. Fig. 6 shows an example of a relevant segment of the *desired ring road path*, the current IARA's pose and the estimated cubic spline.

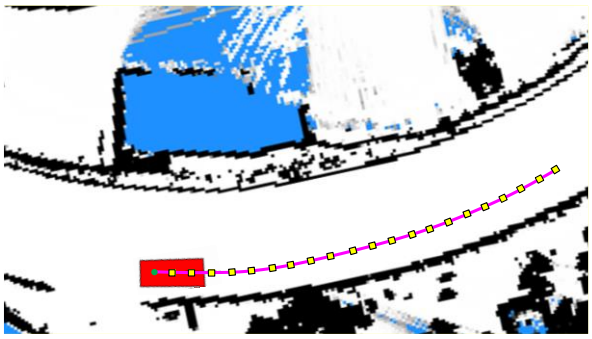

Fig. 6. Example of a relevant segment of the desired ring road path, the current IARA's pose and the estimated cubic spline. The red rectangle indicates the current IARA's pose, yellow points represent the relevant segment of the desired ring road path and the purple curve represents the estimated cubic spline.

More precisely, to estimate the three knots of the path model, we solved the minimization problem given by the equation

$$\arg\min_{K} f(K, P^d), \tag{3}$$

where $P^d = \{p_1^d, p_2^d, \dots, p_i^d, \dots, p_{|P^d|}^d\}$ is the sequence of poses of the *desired ring road path*, in which $p_i^d = (x_i^d, y_i^d, \theta_i^d)$, and

$$f(K, P^d) = \sqrt{\frac{1}{|P^d|}\sum_{i=1}^{|P^d|} d(l_i^e, l_i^d)^2} \tag{4}$$

is the squared root of the average of the summation of the squares of the Euclidian distance, $d(l_i^e, l_i^d)$, between each position estimated using the spline, $l_i^e = (x_i^d, spline(K, x_i^d))$, and the position of the desired path with the same abscissa, $l_i^d = (x_i^d, y_i^d)$.

Initial guesses for the knots k_1, k_2 and k_3 are taken from the ordinates associated to the abscissas closest to 10, 20 and 30 m, respectively, of poses of the *desired ring road path*.

4.4 CNN Loss and DeepPath Evaluation Metrics

To train the DeepPath's CNN, we used as loss function the Mean Squared Error (MSE) of the differences between the elements of the vector of path model parameters estimated from an image by DeepPath, $V^e = \{dy^e, k_1^e, k_2^e, k_3^e\}$, and that inferred from the *desired ring road path* for the image, $V^d = \{dy^d, k_1^d, k_2^d, k_3^d\}$, using the equation

$$\text{MSE} = \frac{1}{|V^e|} \sum_{i=1}^{|V^e|} (v_i^e - v_i^d)^2, \tag{5}$$

where v_i^e is i-th element of the vector estimated by DeepPath (i.e., $v_1^e = dy$, $v_2^e = k_1$, $v_3^e = k_2$ and $v_4^e = k_3$), v_i^d is the i-th element of the vector inferred from the *desired ring road path* and $|V^e| = 4$ is the number of parameters of the path model.

To analyze the performance of DeepPath, for each path model parameter, $g \in V = \{dy, k_1, k_2, k_3\}$, we used the Root Mean Square (RMS) of the differences between the values estimated from an image i by DeepPath, g_i^e, and that inferred from the *desired ring road path* for the image i, g_i^d, using the equation

$$RMS_g = \sqrt{\frac{1}{|Te|} \sum_{i=1}^{|Te|} (g_i^e - g_i^d)^2}, \tag{6}$$

where $|Te|$ is the number of images in the test dataset, Te. In the experiments, we examined the value of RMS_g for each path model parameter, i.e., RMS_{dy}, RMS_{k_1}, RMS_{k_2}, and RMS_{k_3}.

We also used the Root Mean Square (RMS) of the differences between the positions of the path estimated by DeepPath for an image j, $P_j^e = \{p_{j,1}^e, p_{j,2}^e, \dots, p_{j,i}^e, \dots, p_{j,|P_j^e|}^e\}$ and those of the *desired ring road path* for the image j, $P_j^d = \{p_{j,1}^d, p_{j,2}^d, \dots, p_{j,i}^d, \dots, p_{j,|P_j^d|}^d\}$, using the equation

$$\text{RMS}_{p,j} = \sqrt{\frac{1}{|P_j^e|} \sum_{i=1}^{|P_j^e|} (l_{j,i}^e - l_{j,i}^d)^2}, \tag{7}$$

where $l_{j,i}^e = (x_{j,i}^e, y_{j,i}^e) \in p_{j,i}^e$ is i-th position of the path estimated for the image j and $l_{j,i}^d = (x_{j,i}^d, y_{j,i}^d) \in p_{j,i}^d$ is the i-th position of the *desired ring road path* for the image j. In the experiments, we examined the value of $\text{RMS}_{p,j}$ on average for all images in each test dataset, Te, i.e.,

$$\mathrm{RMS}_p = \frac{1}{|Te|}\sum_{j=1}^{|Te|}\sqrt{\frac{1}{|P_j^e|}\sum_{i=1}^{|P_j^e|}(l_{j,i}^e - l_{j,i}^d)^2}. \quad (8)$$

5 EXPERIMENTAL RESULTS

To evaluate the performance of DeepPath, we trained the CNN for inferring path models from images using the training and validation datasets. We then examined the quality of the paths generated by DeepPath by comparing them with the associated path segments of the *desired ring road path*. Finally, we assessed the performance of DeepPath on the autonomous operation of IARA.

5.1 CNN Training

To train the CNN of DeepPath, we initialized all but its last layer parameters with the set of values of the parameters and hyperparameters provided by Bulo et al. [BUL18], who trained their CNN for image segmentation. The last fully connected layer of our CNN (that replaced the final up-sampling section of the original CNN – Section 3.1) was initialized with random weights. The weights of all remaining layers were kept unfrozen for fine-tuning.

The CNN was trained for 11 epochs, with batch size of 4, using the Adam optimizer. As loss function, we used the MSE of the differences between the elements of the vector of path model parameters estimated from an image by DeepPath and those inferred from the *desired ring road path* for the image (see Section 4.3). We performed 9 different training procedures to find a good combination between the value of the learning rate and the size of the final section of the network (fully connected layer with four output neurons). The learning rates used were: 10^{-6}, 10^{-7} and 10^{-8}; and the final section configurations used were: only one layer with 4 output neurons; addition of an intermediate layer with 20 neurons; and addition of an intermediate layer with 100 neurons.

The different values of learning rate were what caused the main difference in the final loss of the training process, but the other changes also enabled slightly better results with each learning rate. All that can be seen in more detail in the following figures.

Fig. 7 shows the evolution of the mean loss of the validation dataset after each train epoch for combinations of a learning rate of 10^{-8} and the three different configurations of the final layers. Fig. 8 and Fig. 9 are the equivalent plots for learning rates of 10^{-7} and 10^{-6}, respectively.

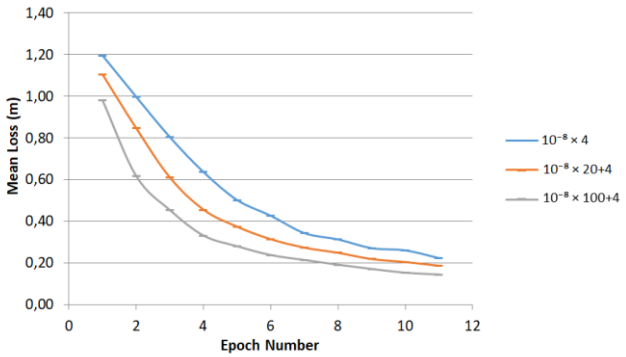

Fig. 7. Mean loss on the validation dataset after each train epoch. In line captions, $10^{-8} \times 4$ denotes the combination of learning rate of 10^{-8} with only one layer with 4 output neurons (as in the original network); $10^{-8} \times 20 + 4$ denotes the combination of learning rate of 10^{-8} with addition of an intermediate layer with 20 neurons; $10^{-8} \times 100 + 4$ denotes learning rate of 10^{-8} with addition of an intermediate layer with 100 neurons.

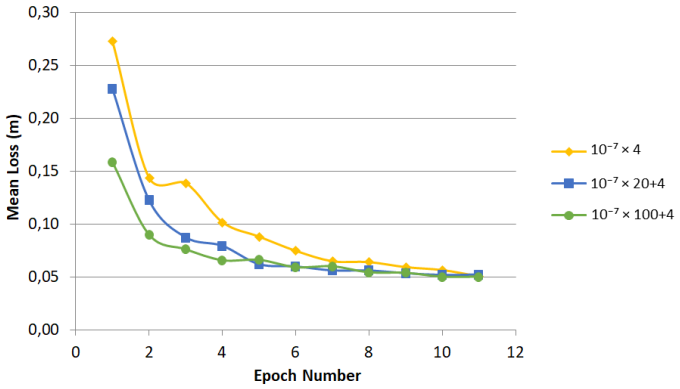

Fig. 8. Mean loss on the validation dataset after each train epoch. In line captions, $10^{-7} \times 4$ denotes the combination of learning rate of 10^{-7} with only one layer with 4 output neurons (as in the original network); $10^{-7} \times 20 + 4$ denotes the combination of learning rate of 10^{-7} with addition of an intermediate layer with 20 neurons; $10^{-7} \times 100 + 4$ denotes learning rate of 10^{-7} with addition of an intermediate layer with 100 neurons.

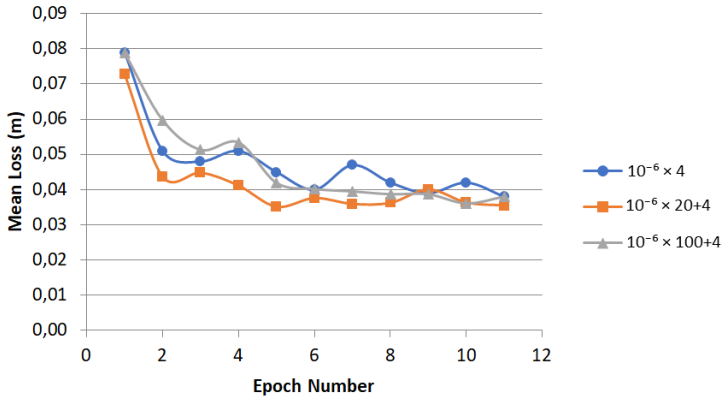

Fig. 9. Mean loss on the validation dataset after each train epoch, for each hyperparameter combination. In line captions, $10^{-6} \times 4$ denotes the combination of learning rate of 10^{-6} with only one layer with 4 output neurons (as in the original network); $10^{-6} \times 20 + 4$ denotes the combination of learning rate of 10^{-6} with addition of an intermediate layer with 20 neurons; $10^{-6} \times 100 + 4$ denotes learning rate of 10^{-6} with addition of an intermediate layer with 100 neurons.

Fig. 10 combines previous results in a single plot for a more general view. As shown in Fig. 10, the learning rate of 10^{-6} provided the best (lower) mean loss among the rates used. Since changes to the final section configuration (addition of an intermediate layer with 20 or 100 neurons) did not significantly reduce the loss, we chose the simplest configuration with only one layer with 4 output neurons.

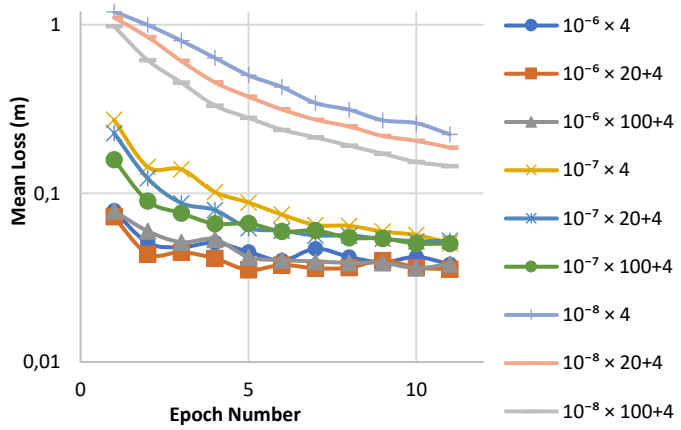

Fig. 10. Mean loss on the validation dataset after each train epoch, for each hyperparameter combination. In line captions, $10^{-6} \times 4$ denotes the combination of learning rate of 10^{-6} with only one layer with 4 output neurons (as in the original network); $10^{-6} \times 20 + 4$ denotes the combination of learning rate of 10^{-6} with addition of an intermediate layer with 20 neurons; $10^{-6} \times 100 + 4$ denotes learning rate of 10^{-6} with addition of an intermediate layer with 100 neurons; and so on.

5.2 DeepPath Test

To analyze the performance of DeepPath on the test datasets, we used the IARA's software infrastructure (Section 4.1.2) to play the same logs of sensor data used for building the test datasets (Section 4.2), so that DeepPath would run as if it was operating connected to IARA in real world scenarios. While the sensor data logs were being played, we saved the path model parameters estimated from images by DeepPath (Chapter 3). We then compared the saved path model parameters with the ground truth (parameters inferred from the *desired ring road path* paths) in the test datasets.

We executed two test experiments with the two test datasets and, for each path model parameter, we measured the RMS of the differences between the values estimated from images by DeepPath and the ground truth in each of the two test datasets. Table 3 shows the RMS parameter errors (in meters). In Table 3, the last line shows, for each path model parameter, the RMS on average for all images in each of the two test datasets. As Table 3 shows, the errors are small–in the order of 1.00 m, at most.

Table 3. RMS of the differences between the parameters estimated by DeepPath and the ground truth

Test Dataset	RMS_{dy}**(m)**	RMS_{k_1}**(m)**	RMS_{k_2}**(m)**	RMS_{k_3}**(m)**
1	0.64	0.36	0.48	0.77
2	0.38	0.27	0.37	0.64
Average	**0.51**	**0.32**	**0.43**	**0.70**

We also examined the RMS of the differences between the positions of the paths estimated by DeepPath for images and those of the *desired ring road path* in each of the two datasets. Table 4 shows the RMS position errors (in meters). As Table 4 shows, the errors are also small – in the order of 0.5 m, at most.

Table 4. RMS of the differences between the positions estimated by DeepPath and the ground truth

Test Dataset	RMS_p**(m)**
1	0.42
2	0.31
Average	**0.37**

Finally, we evaluated the performance of DeepPath on the autonomous operation of IARA. A video that shows the real-time operation of DeepPath in some relevant situations in these experiments is available at https://youtu.be/ZiE7T3I2KDI.

The real world experiments demonstrated that IARA's autonomy system is capable of following a road while using DeepPath as a path planner. DeepPath was able to generate paths that kept IARA in the right lane on the curved sections of the road, as shown in Fig. 11. DeepPath was also able to keep IARA in the right lane on the straight sections of the road, as shown in Fig. 12.

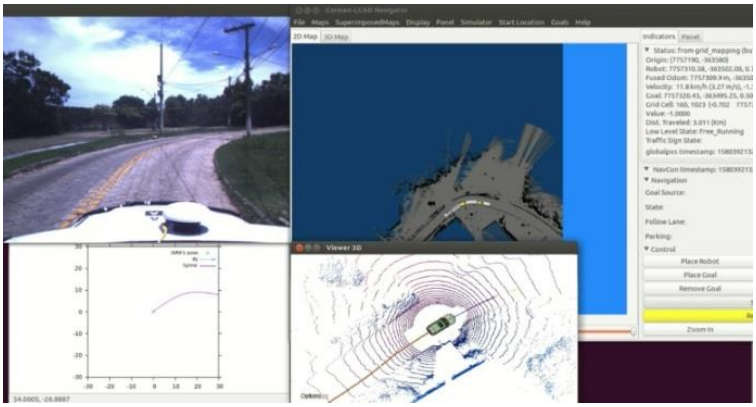

Fig. 11. Screenshot of IARA's autonomy system interface captured during autonomous operation at a time DeepPath generated a path that led IARA safely along a curved stretch of the road. The screenshot shows, in the left upper corner, the camera image; in the left bottom corner, a graph with the current IARA's pose indicated by a green ×, the displacement represented by a blue vector and the cubic spline in the world coordinate system represented by a purple curve (we discounted the current IARA's position of the cubic spline, of the displacement, and of the IARA's position itself, for visualization purposes); in the right upper corner, the online occupancy grid map, which is used by the IARA's Localizer subsystem to generate the current IARA's pose; and, in the right bottom corner, the point cloud computed from the Velodyne LiDAR sensor data, which is used by the IARA's Mapper subsystem to generate the online occupancy grid map.

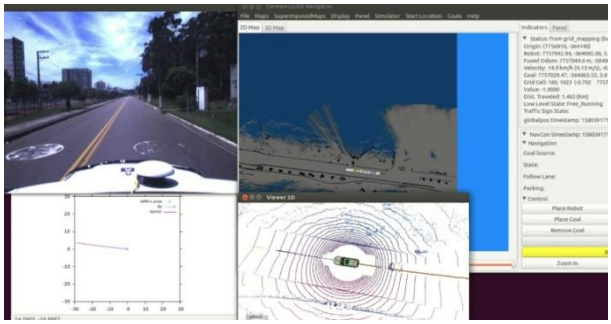

Fig. 12. Screenshot of IARA's autonomy system interface captured during autonomous operation at a time DeepPath generated a path that led IARA safely along a straight stretch of the road

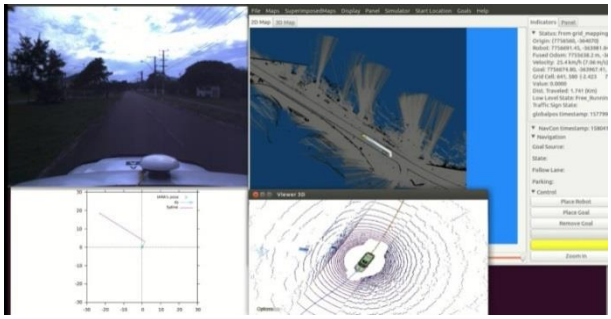

Fig. 13. Screenshot of IARA's autonomy system interface captured during a test experiment at a time DeepPath generated a path that took IARA to the right lane, even with a large displacement, dy

In addition, DeepPath was capable of taking IARA to the right lane, even when the displacement, dy, was large, as shown in Fig. 13. This figure shows an example of this behavior observed during a test experiment with an image (and associated IARA's pose) of the Training Dataset captured while the human driver conducted IARA on a zigzag course.

However, DeepPath generally makes mistakes when the road has forks. When DeepPath encounters a fork in the road, it may not be able to choose a suitable path and IARA is stopped by the Motion Planner or the Obstacle Avoider subsystems. This is expected, because DeepPath does not know the route to follow, as the route is not one of the inputs of its CNN.

5.3 Discussion of Test Results

The related works that we discussed in Chapter 2 evaluated the performance of their methods on path generation quantitatively and/or qualitatively. A quantitative analysis allows for the exposition of the results as plots, tables and calculations, enabling the comparison with other works and deeper insights about the experiments. A qualitative analysis, though not as precise, gives an idea of the perception people have about the behavior of the system regarding aspects that can't be properly measured.

In the qualitative analyses, some works [GUO17] [REH17] [CHE17a] [YAN18] [CAI19] plotted estimated and desired paths on graphs, compared them visually and argued that paths generated by their methods are close to those by human drivers; another work [BOJ16] showed videos of their test cars driving in diverse conditions. In quantitative assessments, a work [CHE17a] computed the errors of steering wheel angle estimates and constructed a histogram of these errors; another work [YAN18] computed the Mean Absolute Error (MAE) of steering

wheel angle and speed predictions; and another work [CAI19] computed the MAE of path position estimates, which can be compared with our results, as we have also computed the errors of path position estimates. Cai et al. [CAI19] reported the MAE of path position predictions of their method for three different behaviors: 0.77 m for keep straight, 0.61 m for turn left and 0.63 m for turn right. For all behaviors, the MAE obtained by their method was larger than the RMS reached by DeepPath, which was 0.37 m, on average. As the RMS is always larger or equal to the MAE, DeepPath outperformed the method by Cai et al. [CAI19].

5.4 Limitations of DeepPath

Because of our choice for a system which infers paths using camera images as the sole input, DeepPath does not apply to situations that require some higher level knowledge of a route to be followed. The main example of such a situation is the occurrence of forks on the road. Since the neural network was trained to recognize plausible paths in the road ahead, returning one, and only one, path as output, images with two distinct possible choices could result in any of those paths being chosen, or it could result in even more unpredictable outputs, because we did not conduct tests for this kind of situation.

Treatment of this case is a very appealing way of adding usefulness to the DeepPath system, so it is also mentioned in the Future Works section, along with some other identified investigation directions.

6 CONCLUSIONS AND FUTURE WORK

This chapter presents the conclusions and future work. In Section 6.1, we present the conclusions based on the experimental results of this research work. In Section 6.2, we present a critical analysis of this research work with direction for future work.

6.1 Conclusions

We proposed DeepPath, an image-based real-time path planner for the self-driving car IARA. DeepPath uses a CNN to infer paths from images. While the self-driving car is in operation, DeepPath periodically receives an image and the current IARA's pose as an input, and generates a path as an output. For this, DeepPath uses a CNN to infer a model of the path, which consists of four parameters: a displacement on the y-axis of the IARA's coordinate system plus a set of three knots of a cubic spline. DeepPath then uses the path model to generate a path in the IARA's coordinate system. Finally, it moves the path to the world coordinate system using the current IARA's pose.

We evaluated the performance of DeepPath in real world scenarios. For that, we used DeepPath for path planning along the ring road of the UFES campus, which has 3.7 km of extension. Our experimental results showed that DeepPath is capable of correctly generating paths for IARA that differ only slightly from desired paths (paths followed by human drivers)– the RMS of the differences between the path model parameters estimated from images by DeepPath and those inferred from desired paths are 0.51 m, 0.32 m, 0.43 m and 0.71 m (for the displacement and the three spline knots, respectively), on average for the test datasets; and the RMS of the differences between the poses of the paths estimated from images by DeepPath and those of the desired paths is 0.37 m, on average for all images in the test datasets.

6.2 Future Work

A direction for future work is to investigate a CNN based approach that infers control commands (i.e., acceleration, brake and steering efforts) from camera images. Another direction for future research is to provide a route as an input to the CNN, which will indicate it the

path forward when DeepPath finds a fork in the road. Finally, other direction for further research is to examine the benefits of using other types of pre-trained CNNs for inferring path models from images.

7 PUBLICATION

During the development of this work, a scientific paper was published, which is referenced to as:

- G. Moraes, A. Mozart, P. Azevedo, M. Piumbini, V. B. Cardoso, T. Oliveira-Santos, A. F. De Souza and C. Badue, “Image-Based Real-Time Path Generation Using Deep Neural Networks”, 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020, pp. 1-8.

8 REFERENCES

[BAD19] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. F. R. Jesus, R. F. Berriel, T. M. Paixão, F. Mutz, T. Oliveira-Santos and A. F. De Souza, "Self-Driving Cars: A Survey", arXiv:1901.04407v2, 2019.

[BAS15] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner and R. F. Werneck, "Route Planning in Transportation Networks", arXiv:1504.05140, 2015.

[BOJ16] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao and K. Zieba, "End to End Learning for Self-Driving Cars", arXiv:1604.07316v1, 2016.

[BUL18] S. R. Bulo, L. Porzi and P. Kontschieder, "In-Place Activated BatchNorm for Memory-Optimized Training of DNNs", IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, pp. 5639–5647, 2018.

[CAI19] P. Cai, Y. Sun, Y. Chen and M. Liu, "Vision-Based Trajectory Planning via Imitation Learning for Autonomous Vehicles", IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, pp. 2736–2742, 2019.

[CAR17] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese and A. F. De Souza, "A Model-Predictive Motion Planner for the IARA Autonomous Car", IEEE International Conference on Robotics and Automation (ICRA), Singapore, pp. 225– 230, 2017.

[CHE17a] Z. Chen and X. Huang, "End-To-End Learning for Lane Keeping of Self-Driving Cars", IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, pp. 1856–1860, 2017.

[CHE17b] L.-C. Chen, G. Papandreou, F. Schroff and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation", Computer Vision and Pattern Recognition, vol. 22, no. 7, pp. 1182-1189, 2017.

[GON16] D. Gonzalez, J. Perez, V. Milanes and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles", IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135–1145, 2016.

[GUI16] R. Guidolini, C. Badue, M. Berger and A. F. De Souza, "A Simple Yet Effective Obstacle Avoider for the IARA Autonomous Car", IEEE 19th International

Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 2016.

[GUI17] R. Guidolini, A. F. De Souza, F. Mutz and C. Badue, “Neural-Based Model Predictive Control for Tackling Steering Delays of Autonomous Cars”, International Joint Conference on Neural Networks (IJCNN), Anchorage, Alaska, pp. 4324–4331, 2017.

[GUO17] C. Guo, T. Owaki, K. Kidono, T. Machida, R. Terashima and Y. Kojima, “Toward Human-Like Lane Following Behavior in Urban Environment with a Learning-Based Behavior-Induction Potential Map”, IEEE International Conference on Robotics and Automation (ICRA), Singapore, pp. 1409–1416, 2017.

[MAA13] A. L. Maas, A. Y. Hannun and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models", 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.

[MAD17] W. Maddern, G. Pascoe, C. Linegar and P. Newman, “1 year, 1000 km: The Oxford Robotcar Dataset”, The International Journal of Robotics Research, vol. 36, no. 1, pp. 3–15, 2017.

[MUT16] F. Mutz, L. P. Veronese, T. Oliveira-Santos, E. de Aguiar, F. A. Auat Cheein and A. Ferreira De Souza, “Large-Scale Mapping in Complex Field Scenarios Using an Autonomous Car”, Expert Systems with Applications, vol. 46, pp. 439–462, 2016.

[PAD16] B. Paden, M. Cap, S.Z. Yong, D. Yershov and E. Frazzoli, “A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles”, IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33-55, 2016.

[POS19] L. C. Possatti, R. Guidolini, V. B. Cardoso, R. F. Berriel, T. M. Paixão, C. Badue, A. F. De Souza and T. Oliveira-Santos, “Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars”, IEEE International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019.

[REH17] E. Rehder, J. Quehl and C. Stiller, “Driving Like a Human: Imitation Learning for Path Planning Using Convolutional Neural Networks”, International Conference on Robotics and Automation Workshops, Marina Bay, Singapore, pp. 1–5, 2017.

[SAN16] E. Santana and G. Hotz, “Learning a Driving Simulator”, arXiv:1608.01230v1, 2016.

[SAR19] R. Sarcinelli, R. Guidolini, V. B. Cardoso, T. M. Paixão, R. F. Berriel, P. Azevedo, A. F. De Souza, C. Badue and T. Oliveira-Santos, “Handling Pedestrians in Self-Driving Cars Using Image Tracking and Alternative Path Generation with Frenét Frames”, Computers & Graphics, vol. 84, pp. 173-184, 2019.

[THR01] S. Thrun, D. Fox, W. Burgard and F. Dellaert, “Robust Monte Carlo Localization for Mobile Robots”, Artificial Intelligence, vol. 128, no. 1–2, pp. 99–141, 2001.

[TOR19] L. T. Torres, T. M. Paixão, R. F. Berriel, A. F. De Souza, C. Badue, N. Sebe and T. Oliveira-Santos, “Effortless Deep Training for Traffic Sign Detection Using Templates and Arbitrary Natural Images”, IEEE International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019.

[VER16] L. de P. Veronese, J. Guivant, F. A. A. Cheein, T. Oliveira-Santos, F. Mutz, E. de Aguiar, C. Badue and A. F. De Souza, “A Light-Weight Yet Accurate Localization System for Autonomous Cars in Large-Scale and Complex Environments”, IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, pp. 520–525, 2016.

[WU19] Z. Wu, C. Shen and A. V. D. Hengel, “Wider or Deeper: Revisiting the ResNet Model for Visual Recognition”, Pattern Recognition, vol. 90, pp. 119–133, 2019.

[YAN18] Z. Yang, Y. Zhang, J. Yu, J. Cai and J. Luo, “End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perceptions”, 24th International Conference on Pattern Recognition (ICPR), Beijing, China, pp. 2289–2294, 2018.