

Perceptual Attention-based Predictive Control

Keuntaek Lee

Georgia Institute of Technology
keuntaek.lee@gatech.edu

Gabriel Nakajima An

Georgia Institute of Technology
gabriel.nakajima.an@gmail.com

Viacheslav Zakharov

Georgia Institute of Technology
vzakharov3@gatech.edu

Evangelos A. Theodorou

Georgia Institute of Technology
evangelos.theodorou@gatech.edu

Abstract: In this paper, we present a novel information processing architecture for *safe* deep learning-based visual navigation of autonomous systems. The proposed information processing architecture is used to support a perceptual *attention-based* predictive control algorithm that leverages model predictive control (MPC), convolutional neural networks (CNNs), and *uncertainty quantification* methods. The novelty of our approach lies in using MPC to learn how to place attention on relevant areas of the visual input, which ultimately allows the system to more rapidly detect unsafe conditions. We accomplish this by using MPC to learn to select *regions of interest* in the input image, which are used to output control actions as well as estimates of *epistemic* and *aleatoric uncertainty* in the attention-aware visual input. We use these uncertainty estimates to quantify the safety of our network controller under the current navigation condition. The proposed architecture and algorithm is tested on a 1:5 scale terrestrial vehicle. Experimental results show that the proposed algorithm outperforms previous approaches on early detection of unsafe conditions, such as when novel obstacles are present in the navigation environment. The proposed architecture is the first step towards using deep learning-based perceptual control policies in *safety-critical* domains.

Keywords: Safe Imitation Learning, Anomaly Detection, Uncertainty Quantification, Bayesian Neural Networks, Perceptual Control, MPC, Autonomous Driving
Supplementary video: <https://youtu.be/-Zmi0HCvM9I>

1 Introduction

For autonomous systems to be able to operate in uncertain environments, they have to be equipped with robust decision-making capabilities using a variety of perceptual modalities including vision. Recent advancements in Artificial Intelligence and Deep Learning, have facilitated the development of algorithms that integrate perception and control in a holistic fashion. The resulting *perceptual control policies* offer unique capabilities with respect to generalization, representation, and performance in tasks such as vision-based navigation.

Prior work on vision-based navigation mostly relies on object detection and segmentation. Recently, Shelhamer et al. [1], Ren et al. [2], He et al. [3] showed improved performance in instance segmentation where both object detection and semantic segmentation is handled. Proceeding this vision step, path planning/trajjectory optimization is performed followed by control.

Alternative methodologies for performing vision-based navigation are via Imitation Learning (IL), also referred to as learning from demonstration. In the IL framework, the learning algorithm has access to an expert policy to take advantage from. This expert policy can come, for instance, from human demonstrations or Model Predictive Control (MPC). For the autonomous driving task, Bojarski et al. [4] proposed an approach for learning to drive a full-size car autonomously directly from vision data. Moreover, Pan et al. [5] accomplished high-speed autonomous driving via an end-to-end IL approach using DAGger [6] algorithm.

Nonetheless, the tremendous success of these methods cannot diminish the importance of *safety*. In particular, safety is a crucial topic in domains with long-tailed input distributions, such as in navigation where novel objects can appear in the visual input. Our previous work [7, 8] addressed the problem of incorporating safety into end-to-end trained controllers via Bayesian Neural Networks (BNNs). By using BNN’s ability to output distributions of its predictions, we were able to quantify the *uncertainty* in the control output of our network policy. When encountering unseen data in the long tails, our BNN policy would output high-variance (high uncertainty) distributions. This, in turn, enabled the system to pass the control authority to a safe expert before the end-to-end trained controller failed.

Despite the success of this approach, the system was appreciably slow at detecting such an increase in uncertainty in the control output. In particular, when encountering new obstacles, the system was only able to detect high uncertainty when reaching too close to the object. This consequently impeded the control authority to be passed to a safe expert in sufficient time, which resulted in aggressive maneuvers and ultimately accidents. This limitation was a consequence of the model’s inability to detect high uncertainty when such anomaly obstacles were distant and occupying limited space in the visual input.

In this work, we address this major limitation by viewing perceptual control policies as Information Processing Architectures (IPAs) and proposing a novel algorithm for safe vision-based control with quick anomaly detection. We accomplish this by introducing the Perceptual Attention-based Predictive Control (PAPC) algorithm, which is comprised of the following ingredients: (1) it uses IL to learn a perceptual controller; (2) it employs BNNs to estimate the uncertainty of the learned controller; and (3) it incorporates a novel *attention mechanism* that enhances the uncertainty quantification method to be able to promptly assess unsafe navigation conditions.

In particular, our attention mechanism takes advantage that our MPC expert policy outputs control and *state trajectories*. We process these trajectories to train a model to predict the vehicle’s future trajectory in the pixel space of the visual input. Finally, these pixel trajectories are then used to place attention on certain parts of the visual input, ultimately enabling our algorithm to detect unsafe conditions significantly quicker. In this work, we simulate safety hazards with the presence of unseen obstacles in the navigation path.

In summary, the contributions of this work are provided as follows:

- We introduce the Model Prediction-Network (MP-Net) for learning the vehicle’s trajectories represented as splines (*spline learning*) in pixel space. The MP-Net is trained using state trajectories generated by MPC. We use MP-Net’s output splines to select Region of Interests (ROIs) in the visual input, where attention is to be placed on.
- We present the eye-inspired Macula-Net, a 3D Convolutional Neural Network (CNN) that uses the ROIs mentioned above as input and generates controls as well as estimates of aleatoric and epistemic uncertainty.
- We integrate all the aforementioned blocks into the PAPC algorithm. PAPC outperforms the prior state-of-the-art solutions on assessing unsafe navigation conditions via early detection of novel obstacles in the vehicle’s way. In contrast to traditional methods, detection of these obstacles is performed without any image classification or object detection.

2 Preliminaries

In this section, we detail the building blocks of the proposed Information Processing Architecture (IPA) for perceptual control.

2.1 Model Predictive Optimal Control

MPC-based optimal controllers (e.g. iterative Linear Quadratic Gaussian/Model Predictive Control Differential Dynamic Programming (iLQG/MPC-DDP) [9], Model Predictive Path Integral [10]) provide planned control trajectories by solving the optimal control problem within a time horizon. An optimal control problem whose objective is to minimize a task-specific cost function can be

formulated as follows:

$$V(x(t_0), t_0) = \min_{u(t)} \left[\phi(x(t_f), t_f) + \int_{t_0}^{t_f} \ell(x(t), u(t), t) dt \right] \quad (1)$$

subject to dynamics $\frac{dx}{dt} = f(x(t), u(t), t)$, where $x \in \mathbb{R}^n$ represents the system states, $u \in \mathcal{U} \subset \mathbb{R}^m$ represents the control, ϕ is the state cost at the final time t_f , ℓ is the running cost, and V is the value function. By solving this optimization problem, we get the future optimal state trajectories from the optimal control trajectories.

In this paper, we take advantage of the optimal state and control trajectories provided by MPC to train perceptual control policies in an IL fashion and design an attention mechanism. As will be explained later, the state trajectories will be used to train CNNs to predict ROIs using raw images while control action will be used to train another CNN to predict the control action using the ROIs as input.

2.2 Perceptual Control via Imitation Learning

One of the deep learning-based perceptual control approaches for a navigation task is training agents to output optimal control actions given image data from cameras.

Reinforcement Learning (RL) is one way to train agents to maximize some notion of task-specific rewards. One of the major problems in RL is the sample-inefficiency problem: the agents have to randomly explore the action-state space without any prior knowledge of the environment or task.

However, IL uses supervised learning to train a control policy and bypasses the sample-inefficiency problem in RL. In IL, a policy is trained to accomplish a specific task by mimicking an expert’s control policy, which in most cases, is assumed to be optimal. Accordingly, IL provides a safer training process. In this work, we train our control policy in an IL fashion and we use MPC as the expert policy.

One of the major problems in IL is that the training data collected from an optimal expert does not usually include demonstrations of failure cases in unsafe situations. Ross et al. [6] introduced an online dataset aggregation algorithm, DAgger, which mixes the expert’s policy and the learner’s policy to explore various situations. However, even with the online scheme of collecting datasets, it is impossible to experience all kinds of unexpected scenarios. Therefore, in order to deploy the trained policy into safety-critical systems, we need a strategy to detect when our trained model is going to fail or provide unsafe results.

2.3 Bayesian Neural Networks

The quantification of uncertainty in the model output is a crucial component for the deployment of DNNs to safety-critical applications. To incorporate this capability to deep learning-based perceptual control policies, we use Bayesian Neural Networks. Currently, Bayes by backpropagation [11], Monte Carlo (MC) dropout [12], and Deep Ensembles [13] are the most common methods to instantiate BNNs. In our work, we use the MC-dropout method, which uses the dropout technique to build a probability distribution over network weights. This allows us to obtain the distribution of the network prediction at test time.

Kendall and Gal [14] introduced the *heteroscedastic* loss function which provides two different notions of uncertainty: aleatoric and epistemic. Aleatoric uncertainty originates from incomplete knowledge of the environment whereas epistemic uncertainty arises from the lack of sufficient data.

In this work, we use this *heteroscedastic* loss function to train our Bayesian network and we use the network’s output variance for the early assessment of unsafe conditions simulated by novel obstacles in the vehicle’s way.

2.4 B-spline

The B-spline is a collection of Bezier splines that are defined by a set of knot coordinates around which each spline is centered [15]. This set of splines has the following continuity requirements: i) The end of the previous curve must have the same value as the start of the next. ii) The first and second derivatives must be conserved between the intersecting points. Therefore, a high-degree

B-spline can smoothly approximate a curve. The equation for a k -degree B-Spline is formulated as $S(t) = \sum_{i=0}^n N_{i,k}(t)P_i$, where (P_0, P_1, \dots, P_n) are control points and $N_{i,k}(t)$ are the basis functions defined using the recursive Cox-de Boor formula [16]. In this work, the B-spline coefficients were used to train the Model Prediction Network described in the next section.

3 Model Prediction Network

Our attention mechanism revolves around the idea that the model should focus on areas (ROIs) of the visual input that the vehicle will navigate towards. Fortunately, the expert MPC’s future state trajectory output provides us with exactly this vehicle course information.

We first transform the MPC’s state trajectory in the original state space to a corresponding trajectory in pixel coordinates represented by a spline (as in Fig. 1 (C)). Next, we use these spline trajectories as targets to train a CNN model we refer to as Model Prediction Network (MP-Net). At test time, when there is no access to the MPC expert, the MP-Net will output a spline trajectory in pixel space, given the vehicle’s visual input. Subsequently, we select a specific number of *focal points* (as in Fig. 1 (D)) along the outputted pixel trajectory, and use these points to create the ROI windows. This mechanism allows the system to put attention on the image regions corresponding to the vehicle’s trajectory, therefore increasing sensitivity to safety hazards along this trajectory, even if such hazards are located far away and rendered small in the visual input.

In Section 3.1, we describe how we obtain the target spline trajectories to train the MP-Net, and in Section 3.2, we detail how the MP-Net is trained on those targets.

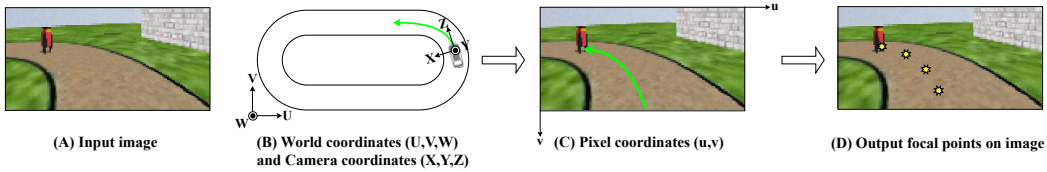


Figure 1: An overview of the Model Prediction Network (MP-Net). (B) From IMU/GPS data, the system states are estimated and the model predictive controller generates the model’s future path/position in the world coordinates. From (A) the input image, the MP-Net is trained to predict (C) the B-spline coefficients of the predicted state trajectory in the pixel coordinates. From the spline coefficients predicted by the MP-Net, we reconstruct the spline and (D) choose focal points from the reconstructed spline.

3.1 Targets for MP-Net using Coordinate Transformation

As seen in Fig. 1, MP-Net projects the vehicle’s future state trajectory described in the world coordinates onto a 2D image in a moving frame of reference. This coordinate transformation technique is widely used in 3D computer graphics [17]. The coordinate transformation consists of 4 steps: World \rightarrow Robot \rightarrow Camera \rightarrow Film \rightarrow Pixel. We follow the convention in the computer graphics community and set: the \mathbf{Z} (optic)-axis as the vehicle’s longitudinal (roll) axis; the \mathbf{Y} -axis as the axis normal to the road, where the positive direction points upwards; and the \mathbf{X} -axis as one perpendicular on the vehicle’s longitudinal axis, where the positive direction points to the left of the vehicle.

Let us define roll, pitch, yaw angles as ϕ, θ, ψ , respectively, the camera (vehicle) position as $U_{\text{cam}}, V_{\text{cam}}, W_{\text{cam}}$ in world coordinates, and the camera focal length as f . Then, we construct the rotation matrices around the $\mathbf{U}, \mathbf{V}, \mathbf{W}$ -axis R_U, R_V, R_W , the translation matrix T_{tl} , the robot-to-camera coordinate transformation matrix $T_{r \rightarrow c}$, and the projection matrix $T_{c \rightarrow f \rightarrow p}$ as:

$$R_U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, R_V = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, R_W = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$T_{\text{tl}} = \begin{bmatrix} 1 & 0 & 0 & -U_{\text{cam}} \\ 0 & 1 & 0 & -V_{\text{cam}} \\ 0 & 0 & 1 & -W_{\text{cam}} \end{bmatrix}, T_{r \rightarrow c} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, T_{c \rightarrow f \rightarrow p} = \begin{bmatrix} f/Z & 0 & o_X/Z \\ 0 & f/Z & o_Y/Z \end{bmatrix},$$

where the projection matrix $T_{c \rightarrow f \rightarrow p}$ projects the point (X, Y, Z) in the camera coordinates into the film coordinates using the perspective projection equations [17] and the offsets o_X and o_Y transform the film coordinates to the pixel coordinates by shifting the origin.

In addition, the total rotation matrix R , the world-to-robot coordinate transformation matrix $T_{w \rightarrow r}$, and the world-to-pixel transformation matrix T are calculated as

$$R = R_W R_V R_U \in \mathbb{R}^{3 \times 3}, \quad T_{w \rightarrow r} = R T_{il} \in \mathbb{R}^{3 \times 4}, \quad T = T_{c \rightarrow f \rightarrow p} T_{r \rightarrow c} T_{w \rightarrow r} \in \mathbb{R}^{2 \times 4}.$$

After converting the \mathbf{X} , \mathbf{Y} , \mathbf{Z} -axes to follow the convention in the computer vision community through $T_{r \rightarrow c}$, the projection matrix $T_{c \rightarrow f \rightarrow p}$ converts the camera coordinates to the pixel coordinates.

We obtain the vehicle (camera) position in pixel coordinates (u, v) with $[u', v']^T = T[U_{cam}, V_{cam}, W_{cam}, 1]^T$. However, this coordinate-transformed point $[u', v']$ in the pixel coordinates has the origin at the top left corner of the image. In our work, as we deal with the state trajectory of the vehicle, we rotate the axes by switching u' and v' , and we define the new origin at the bottom center of the image $[\frac{w}{2}, h]$, where h and w represent the height and width of the image, respectively. Finally, we subtract $[v', u']$ from $[\frac{w}{2}, h]$ and get the final expressions $[u, v] = [\frac{w}{2}, h] - [v', u']$.

3.2 Training MP-Net

Instead of training the MP-Net to predict the entire trajectory in the pixel coordinates, we train it to learn the spline coefficients of the trajectory. This is possible because the MPC trajectories are simple and smooth enough to be represented with splines. This greatly simplifies the regression problem, without jeopardizing performance. To train for spline coefficients, we first fit a spline through the vehicle’s pixel trajectory and we regress on the spline coefficients. At test time, when we don’t have access to the MPC expert, we predict spline coefficients and sample a fixed number of *focal points* along this spline to create the ROIs.

Another way to generate the focal points is by *directly* regressing them in pixel space. However, this is not flexible to changes in the number of focal points, as the network would have to be re-trained for a different number of points. Our *spline-learning* approach allows us to generate any number of focal points, which proved to be very useful during experimentation.

We compared the prediction error of the *spline-learning* and the *direct* focal points learning method. For a fair comparison, we used the same CNN architecture for both methods. For *spline-learning*, we trained the MP-Net to predict the eight B-spline coefficients and for the *direct* focal points learning, we trained the same model to predict the four focal points $[u, v]$ in the pixel coordinates. Our experiments showed that the *spline-learning* method required much fewer training epochs and it clearly outperformed the *direct* focal points learning approach. The average testing error (MSE) in pixel space was 0.4 for the *spline-learning* method and 25.2 for the *direct* focal points learning method. Here, we argue that even with the same number of values, the spline coefficients carry much more information than pixel position values do.

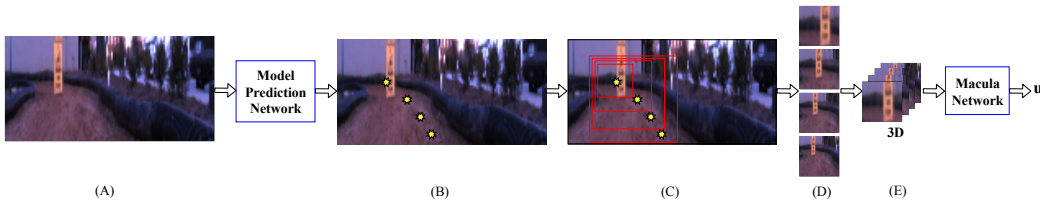


Figure 2: An overview of the Perceptual Attention-based Predictive Control (PAPC) algorithm. (A) Input RGB image with size (64, 128, 3). (B) Focal points (yellow) from the MP-Net-predicted spline. (C) Constructed ROIs from the focal points. (D) Resized ROIs with the same size (32, 32, 3). Bigger ROI loses more resolution by downsampling. (E) Stacked 2D images into 3D data (4, 32, 32, 3). This multi-resolution 3D data resembles the input to the macula in human eyes.

4 Perceptual Attention-based Predictive Control

As described in Fig. 2, once the focal points are obtained from the MP-Net-predicted splines, we (1) construct ROI windows according to these focal points, (2) perform downsampling to lose some resolution, which results in putting more attention on the less-downsampled regions, and (3) feed these processed ROIs into the Macula-Net. The Macula-Net is named after the central part of our eyes’ retinas, where we get the clearest vision with most resolution. The Macula-Net will take these

multi-resolution ROIs and output a control mean and variance via the Bayesian MC-dropout method [12]. For the architecture of the Macula-Net (Fig. 3), we adopted the 3D version of VGG 16 [18]. In addition, to run the PAPC algorithm in real-time (20Hz), 25 Monte Carlo samples were used in the Macula-Net.

The Macula-Net is trained using the *heteroscedastic* loss function [14] to produce a distribution over control actions as an output. This loss function is defined as

$$\mathcal{L}_h(y, \mu, \sigma) = \sum_i \frac{\|y_i - \mu\|_2^2}{\sigma^2} + \log \sigma^2, \quad (2)$$

where y is the target data, and μ and σ represent the Gaussian distribution of the prediction.

The ROIs are constructed in 3 steps: First, define the fovea focal point p_{fovea} as the farthest focal point along the spline. Second, construct the smallest ROI, referred to as fovea, as a window of size 32×32 with p_{fovea} as its center. Third, for each of the other focal points p_i , construct an ROI with center $p_{\text{mid}} = (p_i + p_{\text{fovea}})/2$ and a window size of (w_u, w_v) , where $w_u = 2(p_{\text{mid},u} - p_{\text{fovea},u}) + \text{margin}$, $w_v = 2(p_{\text{mid},v} - p_{\text{fovea},v}) + \text{margin}$. In this way, each ROI can cover the corresponding focal point and the fovea with some *margin*, defined manually as a hyperparameter.

One of the main advantages of our method is that the network can focus on the important/task-related regions of the image, while also eliminating irrelevant parts of the image. This effect can be seen in Fig. 2, in which unimportant features (e.g. buildings, sky, trees, etc.) are filtered out with the ROIs.

We resize all ROIs into the same size as the smallest ROI, which is constructed from the farthest focal point generated by the MP-Net. The resizing step is inspired by the Glimpse Sensor [19], where multiple resolution patches were used to improve classification performance. Unlike the Glimpse Sensor, we do not use a simple fully connected layer after the concatenated multi-resolution 2D images. Rather, we process 3D convolutions to extract 3D information among the stacked images.

Through the resizing step, the smallest ROI from the farthest focal point maintains its resolution, while the bigger ROIs downsample to the fixed size, thus obtaining lower resolution. In this manner, the network resembles the parafovea/perifovea area of the macula. In particular, this resemblance emerges from the mechanism that our eyes focus on a specific region with high resolution (fovea), while other surrounding regions are blurred out with lower resolution (parafovea/perifovea).

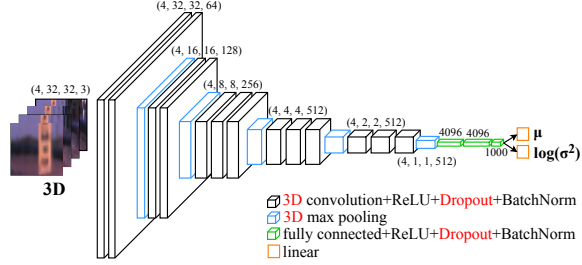


Figure 3: The Macula-Network structure having 3D image data as an input and control action (mean and variance) as an output. The network is the 3D version of VGG16 Network [18] with a Bayesian scheme by dropout.

Algorithm 1 Perceptual Attention-based Predictive Control (PAPC)

Input:

- $\mathbf{o}_{\text{img},t}$: Image from a camera at timestep t ,
- $\mathbf{o}_{\text{world},t}$: States in world coordinates at timestep t ,
- \mathbf{N} : Number of data points for training, \mathbf{T} : MPC timesteps

- 1: **for** $t = 0 : \mathbf{N}$ **do**
- 2: $u_{t,\dots,t+T}, [U, V, W, \phi, \theta, \psi]_{t,\dots,t+T} \leftarrow \text{MPC}(\mathbf{o}_{\text{world},t})$
- 3: **end for**
- 4: $[\mathbf{u}, \mathbf{v}] \leftarrow \text{CoordinateTransform}(U, V, W, \phi, \theta, \psi)$; Section 3.1
- 5: $C_{\text{pixel}} \leftarrow \text{Spline}(\mathbf{u}, \mathbf{v})$; Section 2.4
- 6: $\text{ROIs} \leftarrow \text{GenerateROIs}(\mathbf{o}_{\text{img}}, C_{\text{pixel}})$; Section 4
- 7: **while** Training MP-Net **do**
- 8: $\hat{C}_{\text{pixel}} \leftarrow \text{MP-Net}(\mathbf{o}_{\text{img}})$
- 9: $\text{Loss} = \text{MSE}(C_{\text{pixel}}, \hat{C}_{\text{pixel}})$
- 10: Update MP-Net
- 11: **end while**
- 12: **while** Training Macula-Net **do**
- 13: $\text{ROIs} \leftarrow \text{GenerateROIs}(\mathbf{o}_{\text{img}}, \text{MP-Net}(\mathbf{o}_{\text{img}}))$
- 14: $\hat{u}_{\text{mean}}, \hat{u}_{\text{var}} \leftarrow \text{Macula-Net}(\text{ROIs})$
- 15: $\text{Loss} = \mathcal{L}_h(u, \hat{u}_{\text{mean}}, \hat{u}_{\text{var}})$; Eq. (2)
- 16: Update Macula-Net
- 17: **end while**
- 18: **while** Testing **do**
- 19: $\hat{u}_{\text{mean}}, \hat{u}_{\text{var}} \leftarrow \text{Macula}(\text{GenerateROIs}(\mathbf{o}_{\text{img}}, \text{MP-Net}(\mathbf{o}_{\text{img}})))$
- 20: **end while**

Output: $\hat{u}_{\text{mean}}, \hat{u}_{\text{var}}$: control distribution

We stacked images in 3D, resulting in another dimension, z . Because the number of stacked images is relatively small, we do not want this dimension to be reduced and lose information by a pooling layer (Fig. 3). Therefore, the 3D max-pooling layers in the network act as 2D max-pooling layers, as they do not pool the z -dimension. For 3D filters, (3, 3, 3) kernels are used.

Finally, we combine MPC and the attention-based image processing, using the MP-Net (Fig. 1) and the Macula-Net (Fig. 3), into the P APC algorithm, described in Algorithm 1.

5 Experimental Results

For our experiments, we tested our algorithm on autonomous driving tasks in a simulated environment using the ROS Gazebo simulator [20] as well as with real hardware with a 1/5 scale AutoRally vehicle [21]. In particular, we show successful results of our attention mechanism for evaluating safety conditions. We obtain these results by examining our model’s behavior when encountering safety hazard obstacles in the vehicle’s path.

All of the simulation and real-world experiments including the failure cases can be found in the supplementary material and the video.

5.1 Setup

We conducted 100 test runs with 5 different obstacles in ROS Gazebo to evaluate P APC’s performance compared to the state of the art. In the real hardware experiments, we conducted 10 trials per obstacle and per network for comparison. All of the experiments were done with NVIDIA GeForce GTX 1050 Ti GPU for the real hardware experiments and 1060 GPU for simulation experiments.

MP-Net and Macula-Net were both trained with Adam [22] optimizer in TensorFlow [23]. Additionally, we used the Concrete Dropout method [24] to find the optimal dropout probability per layer in our Bayesian Network. After every convolution and fully connected layer, we performed batch normalization [25] to speed up the training and there was no data aggregation involved except for the 3D stack part.

We set a threshold for the output variance signal to 3-10 times larger than the maximum value of the usual variance in the normal scenario (without any novel obstacles). The specific threshold we set depends on the number of samples we choose for the MC-dropout. The usual output variance in the normal situation without any novel object in the scene was between 10^{-5} and 10^{-4} , depending on the number of MC samples as well. We used this threshold to classify an unsafe driving condition and execute an emergency stop.

In addition, we observed that out of the two uncertainties (epistemic and aleatoric) we get from our Bayesian network trained with the *heteroscedastic* loss function \mathcal{L}_h , the value of the epistemic uncertainty showed a drastic change reacting to the novel obstacles in the vehicle’s trajectory whereas the value of the aleatoric uncertainty does not show such a big difference. This is reasonable because the epistemic uncertainty originates from the lack of data and is therefore expected to provide large variance given input data from the tail of the distribution. Therefore, we used the epistemic uncertainty to quantify the safety of the navigation condition under our network controller.



Figure 4: *Top:* The autonomous driving simulation environment. *Bottom:* The 1/5 scale AutoRally vehicle with onboard cameras used for the experiments.









Obstacles	DropoutVGG[7]	P APC[Ours]
	Min: 0.37 m Avg: 0.39 m Max: 0.42 m	4.28 m 6.87 m 9.22 m
	Min: 2.20 m Avg: 2.54 m Max: 2.86 m	4.81 m 5.48 m 6.25 m
	Min: 0.00 m Avg: 0.00 m Max: 0.00 m	6.80 m 7.25 m 7.83 m
	Min: 2.12 m Avg: 2.25 m Max: 2.44 m	7.62 m 6.87 m 8.33 m
	Min: 1.28 m Avg: 2.06 m Max: 2.44 m	6.55 m 7.51 m 8.17 m
	Min: 0.00 m Avg: 0.63 m Max: 2.51 m	10.58 m 11.28 m 14.96 m
	Min: 0.00 m Avg: 0.26 m Max: 1.29 m	6.91 m 12.55 m 14.63 m
	Min: 0.00 m Avg: 0.67 m Max: 4.11 m	6.17 m 10.09 m 13.25 m

Table 1: Distance left from the object when the network detects it. Top 5 objects are tested in the ROS Gazebo simulator and the bottom 3 objects are tested with our real hardware.

5.2 Result analysis

Our policy not only performs the original driving task, but it can also assess the safety of the current navigation conditions by quantifying uncertainty in its control policy. In particular, this safety mechanism worked successfully with our vehicle promptly detecting safety hazard obstacles in sufficient time.

To quantify the quality of our safety method enhanced with our attention mechanism, we place novel obstacles in the track and record the distance of the vehicle from the obstacle right when the system declared unsafe driving conditions.

We showed in our previous work [7] the increased variance signal from the BNN when a vehicle saw a novel object on the road. However, as previously mentioned, the increase of the variance signal was not fast enough to avoid new obstacles at a high speed.

With our results summarized in Table 1, we show that our PAPC algorithm has a significantly faster mechanism for identifying unsafe driving conditions. Our method enables the vehicle to stop significantly farther away from the safety hazard obstacles, therefore providing the vehicle with enough time to avoid any collisions.

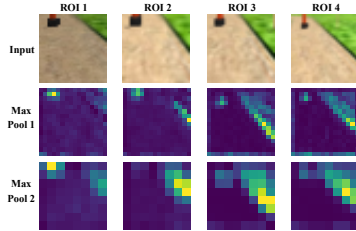


Figure 5: Input stacked 3D ROIs and the averaged feature activation maps after the first two max-pooling layers. Lighter color represents more activation. The smallest ROI, ROI 1, is more activated by the new object than the track/road boundary. As the ROI becomes larger, it tends to be activated from the track boundary than from the new object.

5.3 Analysis of attention mechanism

We analyze the MP-Net by plotting the averaged feature maps after the first two max-pooling layers per ROI (Fig. 5). ROI 1 (fovea) is activated to the feature of a new obstacle, more than the track boundary. However, interestingly, as the ROI becomes larger and downsampled, we can see that the ROI is more activated by the track boundary. From this combination of multi-resolution 3D inputs, the PAPC algorithm is able to detect distant obstacles while focusing on the task-related features.

Moreover, we can see that in Fig. 5 the deeper layers (closer to the output layer) tend to focus more on a single feature. For example in ROI 2, after the first max-pooling layer, the neurons are activated almost equally from both the new object and the track boundary. However, after two convolutional layers and a max-pooling layer, the neurons are activated only from the feature of the track boundary. This pattern is also seen in other ROIs.

Finally, in Fig. 5, we can also see that the resolution of the ROIs becomes lower as the ROI becomes larger since it needs more downsampling. From this downsampling, we can observe the loss of information on the bigger ROIs, where the brightness also has been changed.

6 Conclusion

In this work, we have presented a novel IPA, namely the PAPC algorithm, which performs vision-based navigation and incorporates a highly proficient attention mechanism to assess the safety of the navigation conditions under the network controller. We emphasize here that the PAPC can be used in any autonomous system that performs navigation using visual sensors for safe path planning and control (e.g. visuomotor for manipulation [26]).

In addition, while our initial goal is to use the PAPC architecture as the main system for navigation, its operational role can also be used as a secondary safety controller with the sole purpose of evaluating the safety of the navigation conditions. PAPC’s safety evaluation method can be integrated into decision-making modules to build other robust/adaptive controllers.

The proposed algorithm is validated in both simulation and real-world by outperforming the state-of-the-art approach in a safety-aware autonomous driving task.

Acknowledgments

This work was supported by Amazon Web Services (AWS) and Komatsu Ltd.

References

- [1] E. Shelhamer, J. Long, and T. Darrell. [Fully Convolutional Networks for Semantic Segmentation](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, Apr. 2017. ISSN 0162-8828.
- [2] S. Ren, K. He, R. Girshick, and J. Sun. [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](#). In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.
- [3] K. He, G. Gkioxari, P. Dollar, and R. Girshick. [Mask R-CNN](#). In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [4] M. Bojarski, D. Del 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*, Apr 2016.
- [5] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots. [Agile Autonomous Driving using End-to-End Deep Imitation Learning](#). *Robotics: Science and Systems*, 2018.
- [6] S. Ross, G. J. Gordon, and J. A. Bagnell. [A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning](#). In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR*, Fort Lauderdale, FL, USA, 2011.
- [7] K. Lee, K. Saigol, and E. A. Theodorou. [Early Failure Detection of Deep End-to-End Control Policy by Reinforcement Learning](#). In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8543–8549, May 2019.
- [8] K. Lee, Z. Wang, B. I. Vlahov, H. K. Brar, and E. A. Theodorou. [Ensemble Bayesian Decision Making with Redundant Deep Perceptual Control Policies](#). *The 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2019.
- [9] Y. Tassa, T. Erez, and W. D. Smart. [Receding Horizon Differential Dynamic Programming](#). *Advances in Neural Information Processing Systems 20*, pages 1465–1472, 2008.
- [10] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. [Aggressive driving with model predictive path integral control](#). *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.
- [11] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. [Weight Uncertainty in Neural Network](#). In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR.
- [12] Y. Gal and Z. Ghahramani. [Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning](#). In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.
- [13] B. Lakshminarayanan, A. Pritzel, and C. Blundell. [Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.
- [14] A. Kendall and Y. Gal. [What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?](#) In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.
- [15] C. D. Boor. [On calculating with B-splines](#). *Journal of Approximation Theory*, 6, 1970.

- [16] C. D. Boor. [Package for calculating with B-splines](#). *SIAM Journal on Numerical Analysis*, 14: 441–472, 1977.
- [17] E. Trucco and A. Verri. [Introductory Techniques for 3-D Computer Vision](#). Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998. ISBN 0132611082.
- [18] K. Simonyan and A. Zisserman. [Very Deep Convolutional Networks for Large-Scale Image Recognition](#). In [International Conference on Learning Representations](#), 2015.
- [19] V. Mnih, N. Heess, A. Graves, and k. kavukcuoglu. [Recurrent Models of Visual Attention](#). In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, [Advances in Neural Information Processing Systems 27](#), pages 2204–2212. Curran Associates, Inc., 2014.
- [20] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. [ROS: an open-source Robot Operating System](#). In [ICRA Workshop on Open Source Software](#), 2009.
- [21] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg. [AutoRally: An Open Platform for Aggressive Autonomous Driving](#). *IEEE Control Systems Magazine*, 39 (1):26–55, Feb 2019.
- [22] D. P. Kingma and J. Ba. [Adam: A Method for Stochastic Optimization](#). [Proceedings of the 3rd International Conference on Learning Representations \(ICLR\)](#), abs/1412.6980, 2014.
- [23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. [TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems](#), 2015. Software available from tensorflow.org.
- [24] Yarin Gal and Jiri Hron and Alex Kendall. [Concrete Dropout](#). In [Advances in Neural Information Processing Systems 30](#), 2017.
- [25] S. Ioffe and C. Szegedy. [Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift](#). [Proceedings of the 32nd International Conference on Machine Learning](#), 37:448–456, 2015.
- [26] S. Levine, C. Finn, T. Darrell, and P. Abbeel. [End-to-End Training of Deep Visuomotor Policies](#). [Journal of Machine Learning Research](#), 17(39):1–40, 2016.

Appendix

A1 Comparison Plot

Fig. 6 shows the comparison between our proposed approach, P_{APC}, and the state-of-the-art method DropoutVGG [7] for detecting a new obstacle.

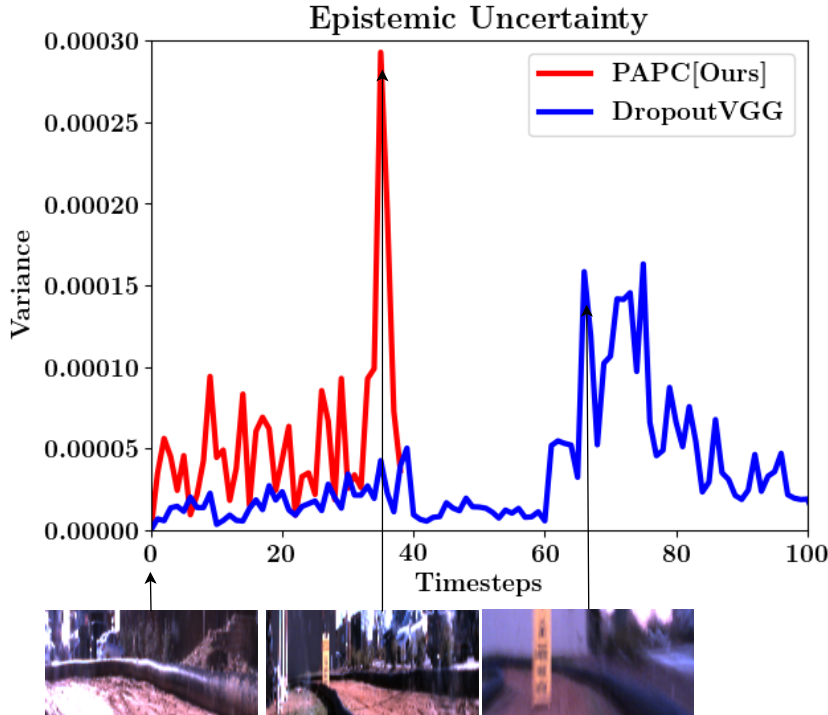


Figure 6: The Output variance plot from Bayesian networks. It shows the epistemic uncertainty from the MC-dropout. P_{APC} shows an earlier detection of the novel obstacle compared to the DropoutVGG [7].

A2 Failure Cases

As shown in Fig. 7, our network sometimes fails, depending on the object size or color. P_{APC} was not able to detect a can in the simulation environment and a detergent container in the real world. We argue that their distributions were not different enough from the training data, even though the fovea ROI caught the object correctly. However, when the vehicle gets closer to the objects, the fovea ROI has already passed it, so no more strong attentions exist at that time. We believe these kinds of smaller objects or those having a similar distribution to the training data without obstacles can be detected by increasing the number of focal points and ROIs, with the fovea having a smaller window. Having more ROIs will require faster GPUs and smaller network structures to run the network in real time, however.

A3 Data Distribution Visualization with t-SNE

Because the Macula-Network uses the Bayesian dropout [12] approach to determine whether ROI contains a new obstacle, it is useful to analyze how the distribution of the ROIs with and without obstacle differ from one another. We use the t-SNE technique for dimensionality reduction in order to visualize the high dimensional ROI images that the Macula-Network takes in as input (Fig. 8). We run t-SNE with perplexity values around 60 with simulated and real-world ROI data. In addition, we not only run t-SNE with the ROI images but also with the output of the first fully-connected layer in the Macula-Net. Using a middle layer allows us to visualize how the Macula-Net itself interprets

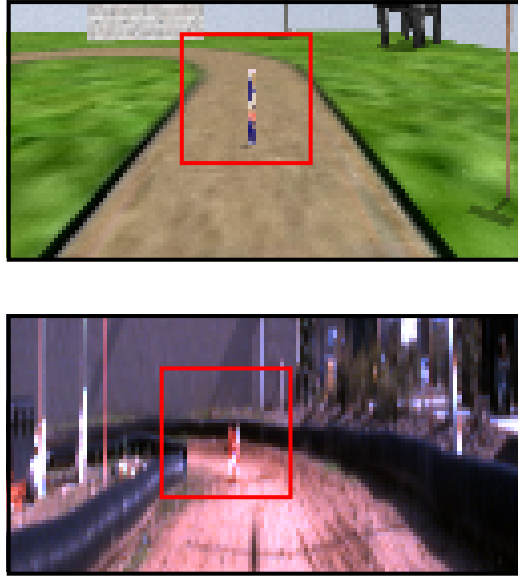


Figure 7: The failure cases of detecting a novel beer can in the ROS Gazebo simulator and the detergent container at the real track. Even though the fovea (red box) caught the objects, the PAPC did not output any meaningful signal.

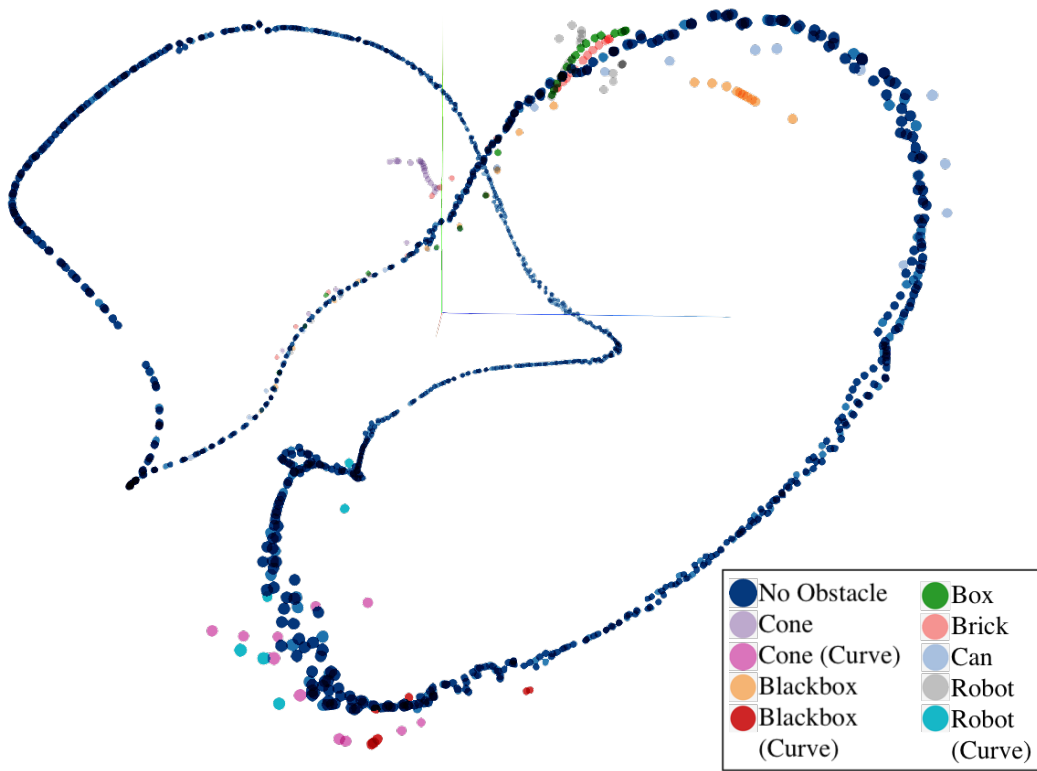


Figure 8: A screenshot of the 3D t-SNE plot of the 3rd max-pooling layer in the Macula-Net. The colorful cluster shows how the Macula-Net is able to abstract the presence of an obstacle in its way. The 3D plot is best viewed in the video: <https://youtu.be/-Zmi0HCvM9I>.

the ROI images. In Fig. 5, we show that the Macula-Net features of the images with obstacles lie in a noticeably different distribution (colorful cluster) as the images with no obstacles. This shows that the Macula-Net was able to capture a change in the distribution of the ROIs when an obstacle is put in the track. Because these image samples with obstacles lie in a different distribution as compared to the training data, the Bayesian network inside the Macula-Net will output control values with high variance, thus indicating the policy that it is no longer safe to drive. In this manner, the algorithm will be able to tell when to concede control to a safer controller.

A4 Expert Model Predictive Controller used to train the model

In this work, iterative Linear Quadratic Gaussian/Model Predictive Control Differential Dynamic Programming [9] was used as an expert controller to drive the vehicle on the track. Using an on-board GPS and a IMU sensor, the state estimation was performed and the particle filter was used for filtering it. With the estimated 12 states $x = [U, V, W, \phi, \theta, \psi, \dot{U}, \dot{V}, \dot{W}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$, the task-dependent cost function Eq. (1) was optimized with the nonlinear vehicle dynamics model in the DDP algorithm. In our experiments, the nonlinear vehicle dynamics are learned from collected data using deep feedforward neural networks. DDP optimization results then provided planned future path of the model within a time horizon along with control trajectories. DDP was performed iteratively with a receding-horizon fashion in 50Hz.

For autonomous driving task on a track, the running cost and the terminal cost was designed to stay at the center of the track depending on U and V . Before running DDP, the center line data (U, V) of the track/map were collected with GPS, and used as some notion of way points which our robot could follow. Additionally, to maintain a certain speed, a quadratic speed cost depending on \dot{U} and \dot{V} was added to the running cost and the terminal cost.