Closed-loop Approach to Perception in Autonomous System

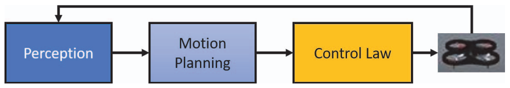
Kruttidipta Samal
School of ECE
Georgia Institute of Technology
Atlanta, USA
ksamal3@gatech.edu

Marilyn Wolf
Department of CSE
University of Nebraska - Lincoln
Lincoln, USA
mwolf@unl.edu

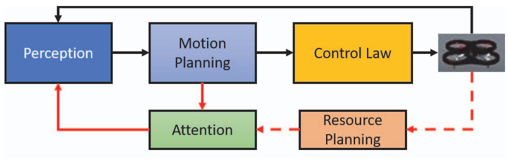
Saibal Mukhopadhyay
School of ECE
Georgia Institute of Technology
Atlanta, USA
saibal.mukhopadhyay@ece.gatech.edu

***Abstract*—Currently, functional tasks within Autonomous Systems are balkanized into several sub-systems such as object detection, tracking, motion planning, multi-sensor fusion etc. which are developed and tested in isolation. In recent times, deep learning is used in the perception systems for improved accuracy, but such algorithms are not adaptive to the transient real-world requirements of an Autonomous System such as latency and energy. These limitations are critical for resource constrained systems such as autonomous drones. Therefore, a holistic closed-loop system design is required for building reliable and efficient perception systems for autonomous drones. The closed-loop perception system creates a focus-of-attention based feedback from end-task such as motion planning to control computation within the deep neural networks (DNNs) used in early perception tasks such as object detection. We observe that this closed-loop perception system improves resource utilization of resource hungry DNNs within perception system with minimal impact on motion planning.**

***Index Terms*—Autonomous Systems, Planning, Deep Neural Networks**



(a) Traditional Autonomy Pipeline



(b) Autonomy Pipeline with closed-loop perception

Fig. 1: High-Level Architecture of an Autonomous Drone System. Top Figure shows the architecture of present systems. Bottom Figure shows the architecture of the Proposed system. The focus-of-attention feedback for closed loop perception is depicted in red. The Resource Planning arm in dotted red line will be explored in the future work.

I. Introduction

In recent years there has been significant interest in real-time autonomous systems such as autonomous drones and vehicles that can navigate in real world environments. Developments in autonomous navigation systems has been possible partly due to innovations in deep neural network (DNN) based perception systems which can be used by upstream tasks such as motion planning for safe navigation. In addition to accurate perception of the state of the world, an autonomous system also needs to operate within real-time constraints such as latency and energy consumption. Within the compute engine of modern autonomous system, deep learning based perception sub-modules consume the highest latency and energy [1]. Therefore, it is critical that any autonomous system that operates in real world should take both accuracy and computational complexity of the perception module into account in order to make safe decisions.

While latency of perception system is important for making faster decisions, energy consumption is critical as well. For systems such as autonomous drones that have a limited energy supply, power consumption of perception systems directly affects flight time and hence their ability to complete the mission. Increasing capacity of power supply is not ideal either as it will increase overall weight of the drone and hence reduce its airworthiness [2]. Furthermore, due to small form factor and limited cooling resources, computing hardware of such systems cannot include powerful GPUs that are widely used for processing DNNs. Therefore, reducing computational complexity of perception systems is paramount for real-time autonomous drone systems.
There have been significant work on reducing computational complexity of deep learning based perception tasks such as object detection using different network architectures [3] [4] but they often lead to reduction in accuracy, and hence, in general can degrade safety. Techniques such as weight and activation quantization, pruning [5], encoding [6] etc. also reduce complexity without changing network architecture, but with degradation of accuracy. This loss in accuracy cannot discriminate between regions of the scene. However, from a navigation system's perspective, some regions in the scene are more important than rest. For example, Samal et. al. [7] have shown that objects that are closer are more important for autonomous driving than objects that are farther away. This priority information associated with different regions of the scene is not privy to low level tasks such as object detection. Instead high level task such as motion planning is cognizant of this information. But present systems with only

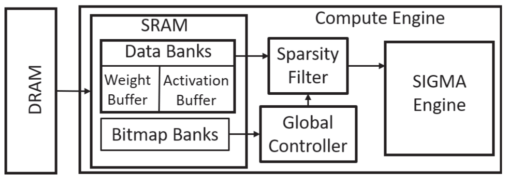


Fig. 2: Architecture of Sigma Accelerator [7]

feedforward data paths (see Figure 1a) cannot capture this focus-of-attention concept.
In this paper we present a closed-loop perception system for autonomous drones that creates a focus-of-attention based feedback from planning to reduce computation within perception module (see Figure 1b). The associated reduction in perception accuracy is localized to regions of the input that are less important for navigation. A real-world simulation environment has been created to implement and test the effect of this closed-loop perception on motion planning in an autonomous drone. We observe that the closed-loop perception can lead to significant reduction in latency and energy consumption with minimal impact on operational safety of the drone.

II. Closed-Loop Design for Resource-Efficient Perception

A. Focus-of-attention as feedback-driven perception

Closed-loop approach to perception introduces the idea of task-level attention to focus on certain regions of the input. The objective of this approach is to reduce utilization of system resources such as latency and energy consumed with graceful degradation in perception accuracy such that there is minimal impact on the end-task. Similar to other approaches to reduce computation, this approach also leads to a reduction in accuracy. But by focusing on regions that are important for the task, closed-loop perception system limits the reduction in perception accuracy at these regions. This is similar to the psychological phenomenon of 'focus-of-attention' [8].

B. How to determine focus of attention

Figure 1b shows the architecture of an autonomous drone system with closed-loop perception system. The perception system is responsible for creating obstacle maps from input images acquired from drone sensors. It uses an object detector DNN to detect objects in 2D RGB image and a localization module to map these objects in 3D using depth map. This object map or obstacle map is used by motion planning module to create a path for moving the drone. The proposed system has a feedback connection between motion planning and perception module (red line in Figure 1b) to support closed-loop perception. Depending on attention strategy, the Attention module creates Regions of Interest (RoI) within the RGB image using input from the motion planning module such as depth of interest, direction of motion etc. This RoI is used to "suppress" activations within the object detector DNN.

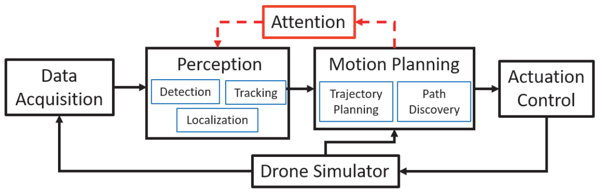


Fig. 3: High-level System Architecture of Closed Loop Simulation System. Functional modules are shown in black boxes and submodules in blue. Red dotted line represents feedback data path.

C. How to use focus of attention regions to control the perception network

In an object detector DNN, early layers have high spatial resolution and are responsible for detecting low-level features such as edges and corners [9]. On the other hand, late layers of the DNN have lower spatial resolution and are responsible for detecting high-level semantic information such as faces etc. Thus by using an RoI to direct suppression, activations in early layers can be reduced based on their spatial location. This ensures the loss of information, and hence of perception accuracy, is localized to regions that are not of interest. The reduction in activations can be utilized by sparsity-aware hardware accelerators to reduce latency and energy utilized by the DNN [10] [11].
Computations within DNNs are implemented as 2D General Matrix Multiplication (GEMM) operations between layer weights and intermediate layer output. This operation is carried out recursively with one layer's output serving as next layer's input. As computational load of the network is directly dependent on the number of these multiplication and accumulation (MAC) operations, there have been many works to increase sparsity within the weights and intermediate layer activations using quantization [5], ReLU activations etc. Sparsity-aware hardware accelerators [10] [11] have been developed that can utilize sparsity in either weight or activation to avoid computation.
As an example, Figure 2 shows the high level architecture of Sigma accelerator [11] that can utilize activation sparsity to reduce memory access and MAC operations. Sparsity information of activation maps are stored in bitmap banks in SRAM. Each element in the bitmap encodes whether the corresponding element in layer input is zero or Non-Zero. Global controller and sparsity filter utilizes this bitmap to access only Non-Zero elements of layer inputs. Weights and activations are stored in "data backs" in SRAM. The Sigma Engine supports flexible datapaths to perform computations at Non-Zero locations only. While supporting sparsity requires an overhead in terms of hardware footprint and FLOPs, they are more than compensated by online resource savings.
Samal et. al. [7] created an analytical model of the Sigma accelerator [11] to estimate the latency and energy consumed by the convolutional backbone of an object detector. Following is the equation for calculating the total amount of memory

accesses and MACs for a given layer.

$$DR^l = \begin{cases} W^l + i_x^l.i_y^l.C_i^l & \text{if } l = 0 \\ W^l + DW^{l-1} & \text{if } l > 0 \end{cases}$$

$$DW^l = s.o_x^l.o_y^l.C_f^l - ac$$

$$MAC^l = \frac{s^l.(f_x^l.f_y^l.C_i^l).(o_x^l.o_y^l.C_f^l)}{PE}$$

$$s^l = \frac{NZ_{input}^l}{N_{input}^l} \tag{1}$$

In the above equation, DW^l and DR^l represent the amount of data written to and read from DRAM respectively, MAC^l represents the number of MAC accesses for layer l. i_x^l, i_y^l represent the input spatial dimension, C_i^l represents the number of input channels of l, o_x^l, o_y^l represent the output spatial dimensions, f_x^l, f_y^l, represent the filter spatial dimensions and C_f^l represents the number of filters. PE is the total number of processing elements in Sigma Engine, ac is the size of activation buffer. W^l represents the size of Weights, NZ_{input}^l and N_{input}^l represent the size of Non-Zero inputs and total number of inputs for layer l. It can be observed that the number of DRAM memory accesses and MAC operations is directly dependent on sparsity of the layer inputs. Similarly, SRAM memory access is also dependent on sparsity and input and weight memory layout, please refer to Qin et. al. [11] and Samal et. al. [7] for details. Thus, by increasing sparsity, both volume of memory access and MAC operations can be reduced.
Samal et. al. [7] observed that the computational latency in the DNN was dominated by DRAM access during processing of early layers. On the other hand energy consumption was dominated by MAC operations. Therefore increasing sparsity of activations in early layers can lead to significant reduction in both latency and energy.

III. Software Environment for Simulating Closed-Loop Perception

Since motion planning is an active task, the closed-loop perception cannot be tested in a pre-recorded dataset, or an open-loop simulation set-up. Therefore, we developed a drone simulation set-up that utilizes an autonomous system with the proposed closed-loop perception to control a quad-rotor. The environment of this simulation set-up consists of moving vehicles and pedestrians in a busy cross-section in an urban environment. We manually defined a global path for the drone that overlaps with traffic and pedestrian movement, thus creating a challenging task for motion planning.
Figure 3 shows the software architecture of the system. This entire system was implemented in Python and has a modular architecture such that underlying algorithms of different modules such as object detector, Attention feedback etc. can be switched with minimal effort. We use Airsim [12] real-world simulator for the drone simulation. Data Acquisition module acquires RGB and Depth images from

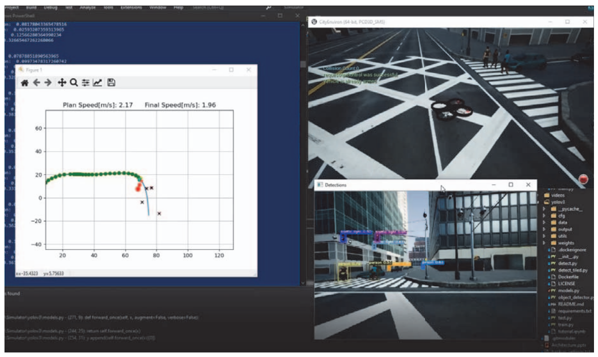


Fig. 4: Simulator Set-Up. Top right window shows the drone that is controlled by our autonomous system software inside the AirSim simulator environment. Motion planning output is shown in left window. Object detection output is shown in right bottom window.

the drone using Airsim APIs. The perception system uses the images to detect and localize objects. This module can support various perception tasks such as object detection, tracking etc. Currently we use an uncertainty aware Yolov3 [13] for object detection, but the architecture can also support other detection algorithms as well. This module is also capable of converting uncertainty of object location from object detector in image co-ordinate system to N-E world co-ordinate system using the depth image, 3D perspective transformation and yaw of the drone.
The motion planning module executes path planning tasks such as path discovery, trajectory planning etc. The trajectory planning algorithm uses current obstacle map, kinematics (position, velocity, yaw etc.) and generates a future path for the drone according to pre-set global waypoints and kinematic limits. We use a modified Frenet frame [14] planning algorithm that takes collision risk into account in addition to motion Jerk. Finally actuation control is responsible for converting future path to angular velocity. This is the actuation signal that is sent to the simulator to move the drone. Feedback data path is implemented between planning and perception module to support closed-loop perception. All the modules are executed on a single thread in a producer-consumer manner to avoid synchronization issues. Cuurently, the system can run at 5fps on a system with 4 core intel i5 processor running at 3.8GHz, 16GB RAM and one Nvidia 1080 GPU with 16 GB memory. Figure 4 shows the output of the system at a time-step while running in open-loop configuration.

IV. Case Studies on Attention Strategies

A. Case Study 1: Distance-Based Attention for Closed-loop Perception

In autonomous navigation systems, local motion planning is a major end-task that uses output of perception module to make motion decisions. In Samal et. al. [7], authors observed that ignoring objects that are located beyond a safe distance

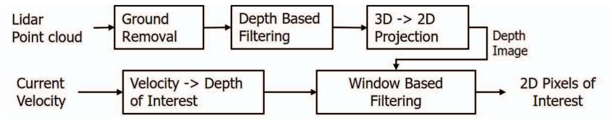


Fig. 5: Distance-based RoI [7].

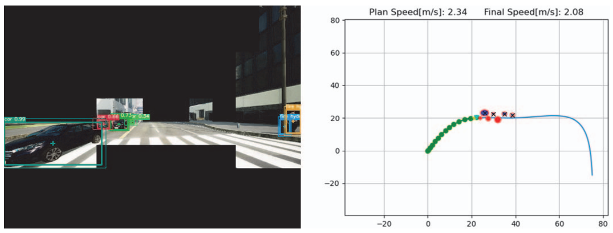


Fig. 6: Visualization of Distance based closed-loop perception.

has minimal impact on motion planning. This is similar to overt attention in biological vision. Using this observation, an RoI extraction algorithm was created that uses Lidar image to determine the RGB pixels that are within a "Depth of Interest" (DoI). The DoI was calculated using the current velocity and a pre-set forecast time. Additionally, ground pixels are determined using RANSAC plane fitting during Lidar pre-processing and corresponding RGB pixels are removed. Figure 5 shows the window based filtering algorithm for pruning an RGB image [7]. Following equation is used to determine the DoI and the pruned RGB image.

$$DoI(t) = Forecast\ Time \times Velocity(t)$$
$$out(x,y) = \begin{cases} in(x,y), \\ \qquad \text{iff } min(dm(w_x,w_y)) < DoI \\ 0, \ \text{else} \end{cases} \tag{2}$$

where, $(w_x,w_y) = \{(x+i, y+i)\}_{-W/2 < i < W/2}, \forall x,y$; dm is the depth map; DoI is the "Depth of Interest"; W is the Window Size, in is the Input Image; and out is the output image i.e. the pruned RGB image. Values of pixels in depth map that belong to ground (as detected by "ground removal" pre-processing) to be very high, thus the above filtering processing deletes most of the ground points as well. We implemented the above approach from Samal et. al. [7] in our simulator set-up. We set a forecast time of 10 seconds and a minimum DoI of 20 meters. Figure 6 shows the output of the system at one time-step. Instead of Lidar point-cloud, depth map was used to detect ground points and depth of pixels in RGB image.

B. Case Study 2: Direction-Based Attention for Closed-loop Perception

Contrary to input pruning, covert attention is not blind to regions outside RoI and still retains ability to detect dominant objects in those regions. In this case, RoIs are determined using future direction of motion as planned by the motion planning module. In such an RoI prediciton scheme, input pruning is not appropriate since it will lead to ignoring

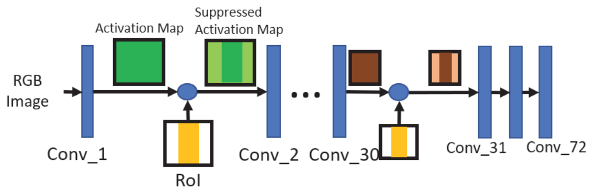


Fig. 7: Layer-wise "activation suppression". RoI is highlighted in yellow. For different layers, activation map is different, but relative position of RoI remains constant.

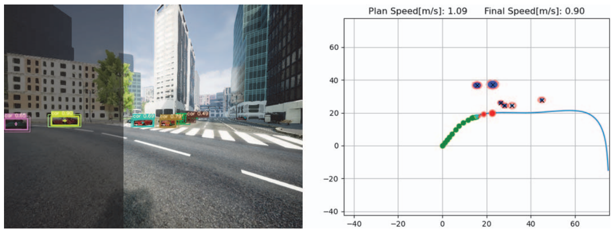


Fig. 8: Visualization of Direction based closed-loop perception. Dark regions are outside RoI.

objects that might be close to the drone but outside RoI. We implemented this attention using "activation suppression" (Section II) within early layers of the object detector DNN.
The Attention module (see Figure 1b) divides the entire RGB image into N_{tiles} number of horizontal tiles and activates these tiles according to future yaws of current planned path. The activated tiles form the RoI in current RGB image. Following is the equation for this conversion.

$$\begin{aligned} tile_{min} &= (FoV/2 + (yaw_{min} - yaw)/tile_{FoV}) \\ tile_{max} &= (FoV/2 + (yaw_{max} - yaw))/tile_{FoV} \end{aligned} \tag{3}$$

Here, $tile_{FoV}$ is the FoV of each tile, FoV is the total Field-of-View of the RGB camera, yaw_{min}, yaw_{max} represent the minimum and maximum yaw according to current path planned in Motion planning module, yaw is the current yaw of the drone. All the tiles between $tile_{min}$ and $tile_{max}$ are activated.
The RoI calculated above is used to suppress activations within the early layers of the object detector DNN. This suppression is carried out by thresholding of the layer-wise activations. Figure 7 shows the layer-wise activation suppression across the convolutional layers of an YOLOv3 [3] object detector with a given RoI. Each layer's threshold is different and is determined by the mean of all the activations outside the RoI. The suppressed output is determined using following equation.

$$\begin{aligned} act^l_{\overline{RoI}} &= [act^l(x,y)], \ \forall (x,y) \in \overline{RoI}^l \\ thr^l_{\overline{RoI}} &= K^l_{ACT}.mean(act^l_{\overline{RoI}}) \\ out^l(x,y) &= \begin{cases} 0, \ if \ act^l(x,y) < thr^l_{\overline{RoI}}, \\ \qquad\qquad \forall (x,y) \in \overline{RoI}^l \\ act^l(x,y), \ otherwise \end{cases} \end{aligned} \tag{4}$$

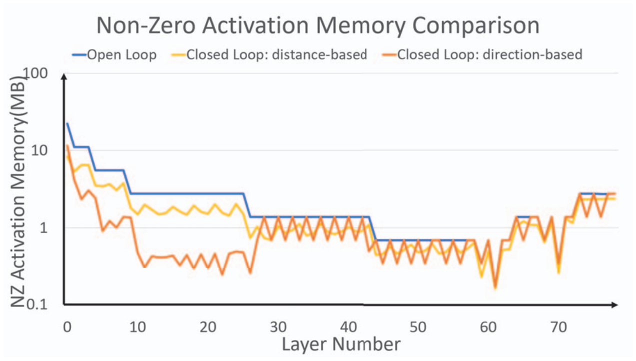


Fig. 9: Comparison of Layer-wise Non-Zero Activations Memory for different systems.

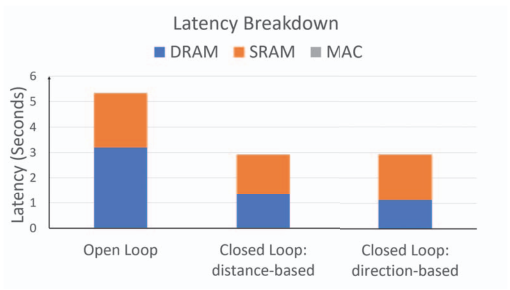


Fig. 10: Comparison of estimated latency by different systems.

Here, $act^l(x, y)$ are the activations at location (x, y) for all feature maps in layer l. $act^l_{\overline{RoI}}$ represent all the activations outside the RoI, $thr^l_{\overline{RoI}}$ is the activation threshold for the layer. $out^l(x, y)$ is the final output activation of the layer. K^l_{ACT} is a hyperparameter. As shown in the Figure 7, this "suppression" is limited to first 30 layers only. Empirically we observed that suppressing activations beyond layer 30 has a severe impact on accuracy of the detector due to increase in receptive field. This limit will be different for different network architectures.

Figure 8 shows the output of the this system at one time-step. It can be observed that regions of the image that are within the current yaws/headings of the planned trajectory, are considered RoI. While early layer activations outside this RoI are "suppressed", the system is still able to detect objects in this region.

C. *Simulation Results*

1) Resource Utilization: Figure 9 shows the layer-wise Non-Zero activation memory for each layer in the object detector DNN. It can be observed that the closed-loop perception systems have substantially lower Non-Zero activations in earlier layers than the open-loop perception system. Also, the direction-based "covert" attention has the least activation among all the systems. We do not observe any substantial difference in activation between all the systems at later layers as most of the neurons in these layers have high receptive field and carry semantically rich information.

Figure 10 shows the estimated per-frame object detection latency in the sparsity-aware computing model discussed in Section II-C. We can observe that both closed-loop perception systems have less than 45% latency compared to open-loop system due to reduction in DRAM accesses. This is due to increased sparsity in early layer activation maps.

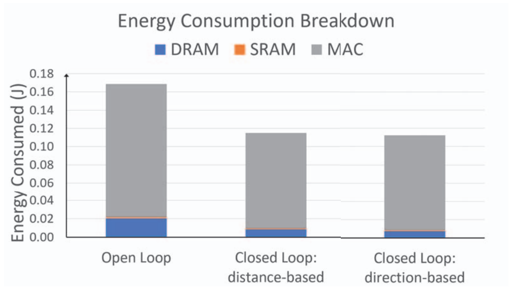


Fig. 11: Comparison of estimated energy consumed by different systems.

TABLE I: Motion Planning Results.

Name	Time Steps	Avg. Speed (m/s)	Avg. Obs.
Open Loop	42	2.07	5.12
Closed Loop			
distance − based	43	1.97	2.76
direction − based	44	1.96	4.06

Figure 11 shows the estimated energy consumed by object detector in the sparsity-aware computing model for processing a single frame. It can be observed that there is about 37% reduction in energy consumed. The reduction is due to increased activation sparsity within the network which leads to reduction in MAC operations within the network.

2) Motion Planning: Table I shows the motion planning results for the different systems averaged over 10 simulations for each configuration. We can observe that there is minimal effect on mission completion with closed-loop perception. There is also minimal effect on time to completion, while the average velocity for closed-loop system is slightly lower than closed loop, this decrease is less than 5%. We do observe that the number of obstacles detected by closed-loop systems is lower than the open-loop system, as closed-loop systems tend to ignore regions of the image that is unimportant for navigation. The decrease in distance-based input pruning is more severe than direction-based "activation suppression", this is because in case of "activation suppression", dominant objects outside RoI are still detected. We did not observe any collision with any of the perception systems. This shows that closed-loop perception does not affect mission safety.

Figure 12 shows the path cost at every time-step for a simulation for different methods. The path cost is a sum of cost functions associated with lateral and longitudinal motion [14]. We also added a collision cost that is inversely dependent on distance between the drone and obstacle positions. Thus, the path cost represents quality of motion, in terms of jerk and collison risk. We observe that the closed loop perception using distance-based attention has very high path cost. On the

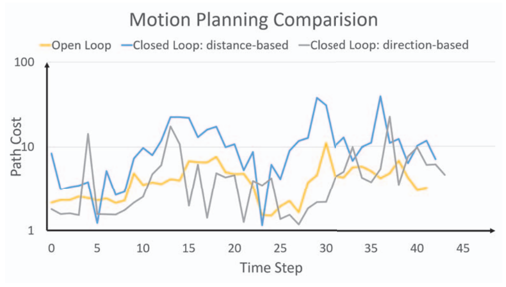


Fig. 12: Comparison of motion planning path cost at different time-steps using different systems.

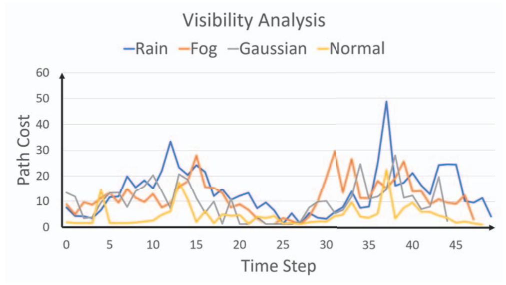


Fig. 13: Comparing effect of adverse visibility condition on the $Closed\ Loop$ system's motion planning.

other hand direction-based closed loop perception has similar path cost compared to open-loop perception. This implies decreasing computations within object detection DNN using closed loop perception reduces motion quality. But direction-based attention is an optimal compromise between motion quality and resource utilization.
We also conducted simulations to study effect of adverse visibility conditions on the closed-loop perception system with direction-based attention. Figure 13 shows the effect of adverse visibility conditions such as rain, fog and gaussian noise on path cost during motion planning at every time-step. While we did not observe any collisions in these simulations, the path cost for adverse conditions especially rain is higher, this indicates a jerkier motion compared to motion under normal visibility condition. We observed the same pattern for distance-based attention as well.

V. Conclusion

In this paper we presented a closed loop perception system that can reduce latency and energy budget of Object detector DNN using attention feedback from motion planning. This attention based "activation suppression" leads high activation sparsity in early layers of the object detector DNN which leads to reduced resource utilization in a sparsity-aware compute engine. The reduction in activation leads to almost 45% decrease in latency and 37% reduction in energy consumed. Experiments on controlling a quad-rotor drone in a real-world simulator shows that the direction-based closed-loop perception can strike a balance between motion planning performance and resource utilization. In the current framework, we have incorporated uncertainty in perception for both object detection and localization but it does not play any role in attention. Similarly, the transient resource requirements of the system such as energy and latency are not used for focus-of-attention. Therefore, in future we plan to integrate both perception uncertainty and resource planning into our attention mechanism. We also plan to explore other attention strategies such as focus driven by target speed, using foveated images instead of fixed resolution images to reduce activation etc.

Acknowledgment

The research reported here was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-17-2-0045. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA.

References

[1] S.-C. Lin *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *ASPLOS*, 2018, pp. 751–766.
[2] L. Pazmany, *Light airplane design*. L. Pazmany, 1963.
[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.
[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
[6] G. Georgiadis, "Accelerating convolutional neural networks via activation map compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7085–7095.
[7] K. Samal, M. Wolf, and S. Mukhopadhyay, "Attention-based activation pruning to reduce data movement in real-time ai: A case-study on local motion planning in autonomous vehicles," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 3, pp. 306–319, 2020.
[8] E. B. Goldstein, *Encyclopedia of perception*. Sage, 2010.
[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.
[10] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.
[11] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 58–70.
[12] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.
[13] J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 502–511.
[14] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.