COMPUTER VISION

Ajna: Generalized deep uncertainty for minimal perception on parsimonious robots

Nitin J. Sanket[1,2]†, Chahat Deep Singh[1]*†, Cornelia Fermüller[1], Yiannis Aloimonos[1]

Robots are active agents that operate in dynamic scenarios with noisy sensors. Predictions based on these noisy sensor measurements often lead to errors and can be unreliable. To this end, roboticists have used fusion methods using multiple observations. Lately, neural networks have dominated the accuracy charts for perception-driven predictions for robotic decision-making and often lack uncertainty metrics associated with the predictions. Here, we present a mathematical formulation to obtain the heteroscedastic aleatoric uncertainty of any arbitrary distribution without prior knowledge about the data. The approach has no prior assumptions about the prediction labels and is agnostic to network architecture. Furthermore, our class of networks, Ajna, adds minimal computation and requires only a small change to the loss function while training neural networks to obtain uncertainty of predictions, enabling real-time operation even on resource-constrained robots. In addition, we study the informational cues present in the uncertainties of predicted values and their utility in the unification of common robotics problems. In particular, we present an approach to dodge dynamic obstacles, navigate through a cluttered scene, fly through unknown gaps, and segment an object pile, without computing depth but rather using the uncertainties of optical flow obtained from a monocular camera with onboard sensing and computation. We successfully evaluate and demonstrate the proposed Ajna network on four aforementioned common robotics and computer vision tasks and show comparable results to methods directly using depth. Our work demonstrates a generalized deep uncertainty method and demonstrates its utilization in robotics applications.

INTRODUCTION

As an old saying goes, "If knowledge is power, knowing what you don't know is wisdom" (*1*). It is as important to know when the agent is unsure as much as the correctness of the prediction. Especially in the case of neural network predictions, estimating the uncertainty associated with these predictions aids in making better decisions rather than blindly relying on these predictions based on the assumption that they are correct. Roboticists have remarked on this observation, and this led to the approach of combining multiple measurements using uncertainties, which has become the gold-standard approach in robotics. Fundamentally, these measurements are combined using Bayesian formulations and propagating the distribution statistics.

Although uncertainties are very useful for combining multiple measurements, we believe that they are underused in robotics. This is because uncertainties also provide contextual cues/information. Before we provide examples of the previous statement, let us discuss two kinds of common uncertainties: (i) aleatoric or observational data uncertainty and (ii) epistemic or model uncertainty.

The aleatoric uncertainty models the inherent bias in the way a sensor collects data, and the epistemic uncertainty models the inherent bias in the scenarios used to collect the training data. For example, the aleatoric uncertainty would be high for transparent or dark regions for RGB-Depth data, and the epistemic uncertainty of a network trained indoors would be high when tested on outdoor data.

The contextual information that an epistemic uncertainty model provides is that the trained model requires more data to improve accuracy for the particular input sample. Such information is useful to know whether one is operating "out of domain" and whether online learning is required for a desirable operation. On the contrary, contextual information from aleatoric uncertainty when studied more carefully is more intriguing because it helps unravel information about the scene based on the sensor characteristics. For example, cameras cannot see through objects; hence, one would expect high aleatoric uncertainty at the object's depth boundaries, which can act as a powerful cue for performing various robotics tasks.

Furthermore, from a pragmatic viewpoint, estimating epistemic uncertainty requires variational inference and multiple runs of the neural network, making it ineffectual for real-time applications unless multiple neural network accelerators are used. On the contrary, aleatoric uncertainty is highly suited for real-time applications because it requires a minor increase in the number of parameters and requires a single pass of the network to predict the uncertainty. In this work, we focus on estimating the heteroscedastic aleatoric uncertainty or observational uncertainty with respect to the input data.

In particular, we propose a generalized loss function formulation to estimate the heteroscedastic aleatoric uncertainty that can be used to model various probability distributions and relate it to the works in the past decade. This demonstrates that previous works are special cases of our generalized formulation. Furthermore, we present a theoretical analysis of what information/cues this uncertainty formulation provides for various prediction modalities. Last, we apply our predicted uncertainty to perform various robotic tasks and demonstrate the unification such a methodology can bring to various classes of robotics problems. We call our class of networks

[1]Perception and Robotics Group (PRG), University of Maryland, College Park, MD, USA. [2]Perception and Autonomous Robotics (PeAR) Group, Worcester Polytechnic Institute, Worcester, MA, USA.
*Corresponding author. Email: chahat@umd.edu
†These authors contributed equally to this work.

Ajna, which is named after the third eye of Lord Shiva from Hindu mythology and refers to the eye of wisdom/consciousness/intuition because our networks can "see" (predict) where they might not work well. We denote the uncertainty of predicted values as Υ because it represents the Greek letter for u standing for uncertainty and resembles the shrug emoji. We formally define the problem statement and a list of our contributions next.

Problem formulation and contributions

We address the following questions: How do you estimate the heteroscedastic aleatoric uncertainty of a neural network? What informational cues does it provide for various robotic tasks? Given an input x, label $\hat{y}$, and prediction $\tilde{y}$, we predict the heteroscedastic aleatoric uncertainty Υ by minimizing the proposed generalized loss function. This loss function reduces to classical statistical properties of variance for common distributions, such as Gaussian or Laplacian. Furthermore, we use this loss function to learn the uncertainty of optical flow and apply it for four example robotic tasks—navigating through a scene with static obstacles, dodging unknown dynamic obstacles, detecting and flying through unknown shaped gaps, and segmenting an unknown object pile (see Fig. 1). A summary of our contributions is as follows: We propose a generalized heteroscedastic aleatoric uncertainty formulation for neural networks, provide the analysis of informational cues provided by heteroscedastic aleatoric uncertainty for robotic tasks, and perform extensive real-world experiments demonstrating how such uncertainty can be used for various robotic tasks (Movie 1).

Related work

Uncertainties and error statistics have been used for decades in robotics. For the related work, we will present works that estimate uncertainties in neural networks and applications of deep uncertainty in computer vision and robotics.

Estimating uncertainties in neural networks

As we mentioned before, there are two types of uncertainties: aleatoric or observational uncertainty and epistemic or model uncertainty. Earlier works estimated either the aleatoric or epistemic uncertainty alone. Previous works (*2*–*4*) estimated only epistemic uncertainty by assuming a Gaussian prior distribution over weights. Such a class of models is called Bayesian neural networks (BNNs). The mathematical formulations of BNNs are simple, but inference requires complex computing because one has to compute marginal distributions across all neurons. Furthermore, Gal and Ghahramani (*5*) introduced a dropout variational inference to make the epistemic uncertainty estimation tractable by using stochastic Monte Carlo dropout. Gal (*6*) introduced a method to perform aleatoric uncertainty estimation alone, which was later combined with epistemic uncertainty to obtain the total uncertainty in Kendall and Gal (*7*). These methods, however, were too slow to run on a robot or not accurate enough; to this end, Gast and Roth (*8*) introduced lightweight probabilistic deep networks by propagating uncertainties using assumed density filtering and an even faster variant by directly predicting the uncertainties only in the final layer. This work was further extended to be agnostic to the network architecture and loss function in Loquercio *et al.* (*9*). A review of more works can be found in (*10*), and we redirect the keen reader to this work for a more detailed summary of prior work.

Applications of deep uncertainty in robotics and computer vision

In robotics, uncertainties and their statistics have been commonly used to fuse multiple measurements from either a single sensor or multiple sensors. Recently, the domination of the accuracy charts by

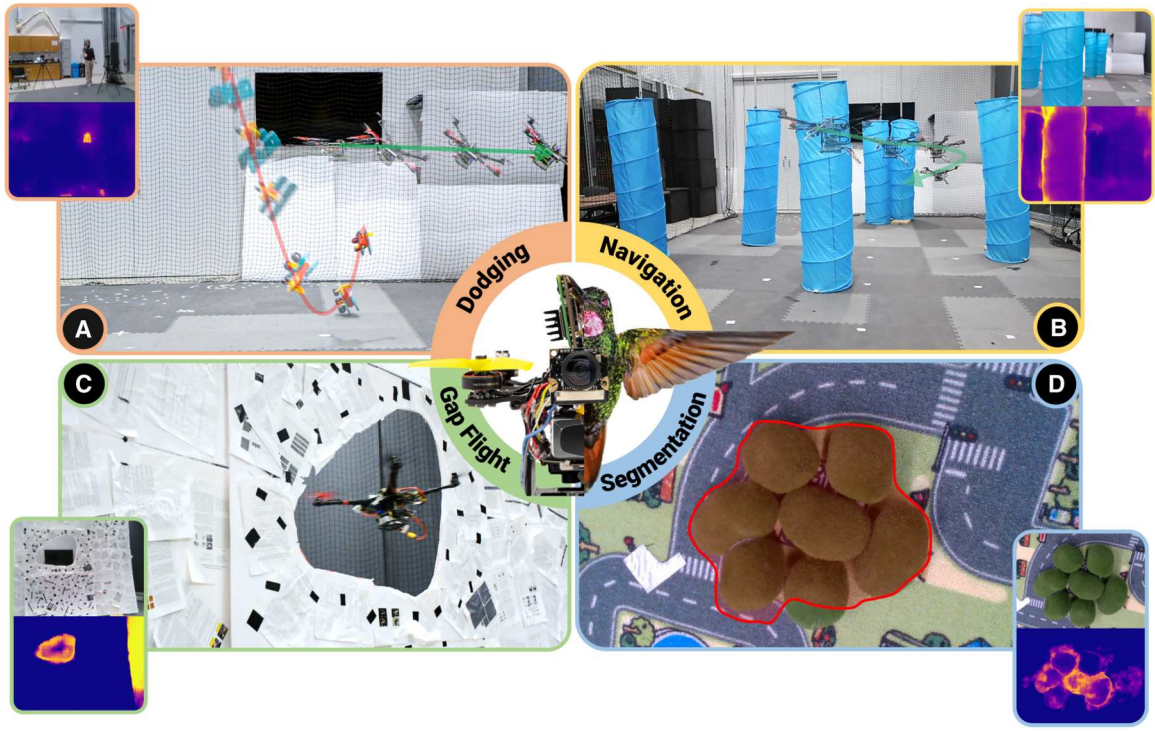

Fig. 1. Unification of common robotics problems using our generalized uncertainty formulation.

Movie 1. Demonstration of real-world quadrotor experiments using uncertainty estimation in optical flow.

deep learning approaches has shifted research focus toward the fusion of uncertainties in neural networks. TLIO (*11*) presented a method to fuse multiple inertial measurements to estimate odometry using the predicted uncertainties coupled to an extended Kalman filter. KFNet (*12*) introduced the concept of fusion of measurement and process model based on the classical Kalman filter formulation (*13*) in the form of a neural network and applied it to the problem of camera relocalization. IVOA (*14*) fused the predicted uncertainties into the navigation stack for robust performance. Furthermore, Loquercio *et al.* (*9*) presented a general framework for uncertainty estimation by combining both aleatoric and epistemic uncertainties and applied it to three tasks: end-to-end steering angle prediction, object motion prediction, and closed-loop control of a quadrotor.

In computer vision, using deep uncertainty predictions to improve performance has gained center stage in the past half decade. Various applications, such as object detection, estimation of optical flow, visual odometry monocular depth, stereo depth/disparity, and surface normal, have leveraged uncertainties as a regularizer for improving robustness. Feng *et al.* (*15*) presented a method to improve the robustness of three-dimensional (3D) object detection using light detection and ranging (LiDAR) data by learning to ignore being trained from noisy samples. The authors in (*8*, *16*–*19*) used either a generative adversarial model or aleatoric uncertainty model to estimate uncertainties that were used as regularizers to train the optical flow model, which has been proven empirically to improve performance. In our work, we present theoretical reasoning as to why this is occurring—due to loss attenuation at optical flow discontinuities. The authors in (*20*–*23*) presented methods to estimate dense depth from either stereo- or monocular views by improving accuracy at the boundaries using an uncertainty metric. Furthermore, Martin-Brualla *et al.* (*24*) used the same aleatoric uncertainty formulation to improve the results of volumetric color rendering in a NeRF model by rejecting dynamic objects using uncertainty. In (*25*), uncertainty was used to perform self-supervised depth completion and resulted in state-of-the-art performance. Similar to (*25*), the authors in (*26*) also used uncertainty obtained by image flipping to improve the results in monocular depth estimation. Costante and Mancini (*27*) presented a method to estimate and incorporate total uncertainty into a deep visual odometry pipeline. Furthermore, Kawashima *et al.* (*28*) presented an alternative method for aleatoric uncertainty estimation using virtual residuals to tackle the problem of overfitting and showed state-of-the-art results in age and monocular depth estimation. Alternatively, uncertainty has also been indirectly learned as the probability of outlier/inlier in SFMLearner (*29*). All in all, the utilization of generalized uncertainty has been widely used to improve various kinds of predictions.

RESULTS

Quadrotor platform

The quadrotor used in the experiments is a custom-built platform called PRGLabrador500 (*30*). The platform was built on an X-shaped 500-mm-sized (motor-to-motor dimension) frame. Each of the four T-Motor F80 Pro 2500KV motors was mated to 6042 × 3 propellers to provide thrust in the system. The position-hold and lower-level controllers were handled by the ArduPilot.1.4 firmware running on the Holybro Kakute F7 flight controller coupled with a GL9306 optical flow sensor and Benewake TFMini-S LiDAR as the altimeter source. The higher-level navigational commands were processed on and sent by the companion computer NVIDIA Jetson TX2 (*31*) using RC-Override to the flight controller running in Loiter mode using MAVROS. The TX2 runs the vision and planning algorithms on board at around 8 Hz on Python 3.6. The quadrotor take-off weight, including an 1800-mAh 3S LiPo battery, was 1110 g, and the quadrotor had a thrust-to-weight ratio of 4.9:1 and a flight time of about 10 min. All flight experiments were performed in the Brin Family Aerial Robotics Lab at the University of Maryland with a flying volume of 7.3 m by 5.5 m by 5 m.

Perception pipeline

In this section, we will describe the overall procedure for achieving the proposed tasks. It involves two key steps: perception and control. The perception pipeline was common for every task and is described next.

The perception pipeline ran on consecutive RGB color frames at an image size of 320 pixels by 240 pixels at a frame rate of 30 Hz. These consecutive image frames were fed into our neural network that was based on the EVPropNet architecture (*32*) with the only change being the number of output channels, which in our case was four rather than one (*32*). Our network Ajna has 2.72 million parameters, uses about 6.3 GFLOPs for a forward pass, and is of model size 10.40 MB, and each inference took about 49 ms (20.4 Hz) on a batch size of one. Ajna was trained in using the loss described in Eq. 11, which used self-supervision to learn the uncertainty and supervised labels to learn the target predictions. In particular, our network was trained to predict dense optical flow $\tilde{\hat{p}}_x$ and its dense heteroscedastic aleatoric uncertainty Υ_x. Ajna was trained on Flying Chairs 2 dataset (*33*, *34*) for 400 epochs at a learning rate of 10^{-4} and then trained further for 50 more epochs on the FlyingThings3D dataset (*35*) at a learning rate of 10^{-5}. The batch size used was 32. We used the following loss function for training our networks based on (*36*):

$$\arg\min_{\tilde{\hat{p}}_x \Upsilon_x} \mathbb{E}\left(\frac{\|\tilde{\hat{p}}_x - \tilde{\hat{p}}_x\|}{\log(1+e^{\Upsilon_x+\epsilon})} + \lambda \log(1+e^{\Upsilon_x})\right) \quad (1)$$

In our mathematical formulation in Eq. 11, this is equivalent to using $f(\tilde{y},\hat{y}) = \|\tilde{y}-\hat{y}\|_1, h(a) = 1/\log(1+e^{a+\epsilon}), g(a) = \log(1+e^{a})$, with $\epsilon = 10^{-3}$, $\lambda = 1.0$, and $\mathbb{E}$ is the expectation/averaging operator. All the hyperparameters are obtained through cross-validation.

To summarize, the input to our networks was consecutive image frames and the output was four channels: two channels for optical flow in the *x* and *y* direction and two channels for uncertainty in the *x* and *y* optical flow. We denoted per pixel the optical flow vector at location $\mathbf{x}$ as $\tilde{\hat{p}}_x \in \mathbb{R}^{2\times 1}$ and its aleatoric heteroscedastic uncertainty as $\Upsilon_x \in \mathbb{R}^{2\times 1}$. We then used the predicted optical flow uncertainty Υ to compute a point on the image using morphological operations, which was in turn used to compute the control strategy based on the task as explained in the following sections. The goal of this work is to showcase how uncertainty can be used for various robotics applications and how such a formulation can unify classes of robotics problems together. Hence, we did not use any other information, such as color, optical flow, or depth, in our experiments, which are merely shown for comparison purposes in this paper. Furthermore, no prior knowledge about placement or type of structure(s) was used in our experiments. Our control actions were based on Υ obtained from the current image pairs; no temporal smoothing or filtering was used. All our perception, planning, and control algorithms ran on board the NVIDIA Jetson TX2 and the flight controller on the aerial robot and hence could be ported to a palm-sized aerial robot (*37*, *38*) without any added effort. In the following sections, we describe the specific experiment and its environmental setup along with the control policies for four applications: dodging dynamic obstacles, navigating through unstructured environments, flying through an unknown gap, and finding the object pile.

Dodging dynamic obstacles

In this experiment, we present a method to detect and dodge unknown (zero-shot) dynamic obstacles using only a monocular camera. The procedure of dodging dynamic obstacles involves three key steps: first, detection of the obstacle or independently moving object(s); second, prediction of the obstacle trajectory on the image plane; and third, invoking a dodging maneuver to avoid getting hit by the obstacle(s). We used the fact that a dynamic obstacle will have the maximum amount of occlusions and accretions on the consecutive frames from a hovering quadrotor. This in turn leads to high uncertainty of optical flow. We detected the dynamic obstacle by performing simple morphological operations on the obtained uncertainty map. Further, we tracked the obstacle over three frames by detection (segmentation) to compute the direction the obstacle would hit on the image plane. Then, we computed a safe direction and executed a control command to move in that direction for "best-effort" dodging as proposed in (*39*). We describe the experimental setup and results next.

The experimental setup contained a hovering quadrotor at which the obstacles were thrown or flown into such that a collision would definitely occur if the quadrotor were to hold its position and not invoke a dodging maneuver. We experimented with four different obstacles, varying in shape, size, color, texture, and trajectory: a spherical ball of diameter 140 mm, a toy car 185 mm by 95 mm by 45 mm, a toy airplane 270 mm by 250 mm by 160 mm, and a PRGHusky360 quadrotor 440 mm by 370 mm by 160 mm. No prior information about the objects was used in any of the experiments. The objects—ball, car, and airplane—were thrown at the quadrotor and followed a parabolic trajectory under the influence of gravity, whereas the PRGHusky360 quadrotor followed a linear trajectory. The objects were thrown or flown at speeds of 4.5 to 8.0 m s^{-1} from a distance ranging from 4.8 to 6.0 m. We achieved an overall success rate (SR) of 83.3% over 60 trials. We compared our results with depth-based methods, event-based methods, and occlusion-based methods (Fig. 2 and Tables 1 and 2). In the depth-based methods, D435i gave true scale depth, whereas the MiDaS and MiDaS-S (*40*, *41*) only output relative scale depth. Here, MiDaS denotes the MiDaS v3.0 DPT-Large (*41*) pretrained model, and MiDaS-S denotes the MiDaS v2.1 small pretrained model directly obtained from the original work without any fine-tuning or retraining. For comparison with occlusion-based methods, we used the predicted occlusion mask from MaskFlowNet (*42*). We call this prediction OccMask. In all the depth-based methods, we threshold the depth value as an obstacle when it is closer than a particular depth value to dodge them. We observed that Intel RealSense D435i (one of the best depth sensors on the market) was not able to obtain depth on moving objects accurately when they were far, hence necessitating an alternative formulation for dynamic obstacle dodging like the one presented in this work. In the event-based method, the approach was adapted from (*39*), where the output is the probability of each pixel being an obstacle. Alternatively, a stereo pair of event cameras can provide the guarantee of dodging obstacles (*43*). Last, in the occlusion-based method, we threshold the large values as belonging to dynamic obstacles followed by morphological operations similar to our method, Ajna. We compared our results with the aforementioned methods on metrics such as detection rate (DR), run time, FLOPs, and number of parameters in Table 1. Here, we

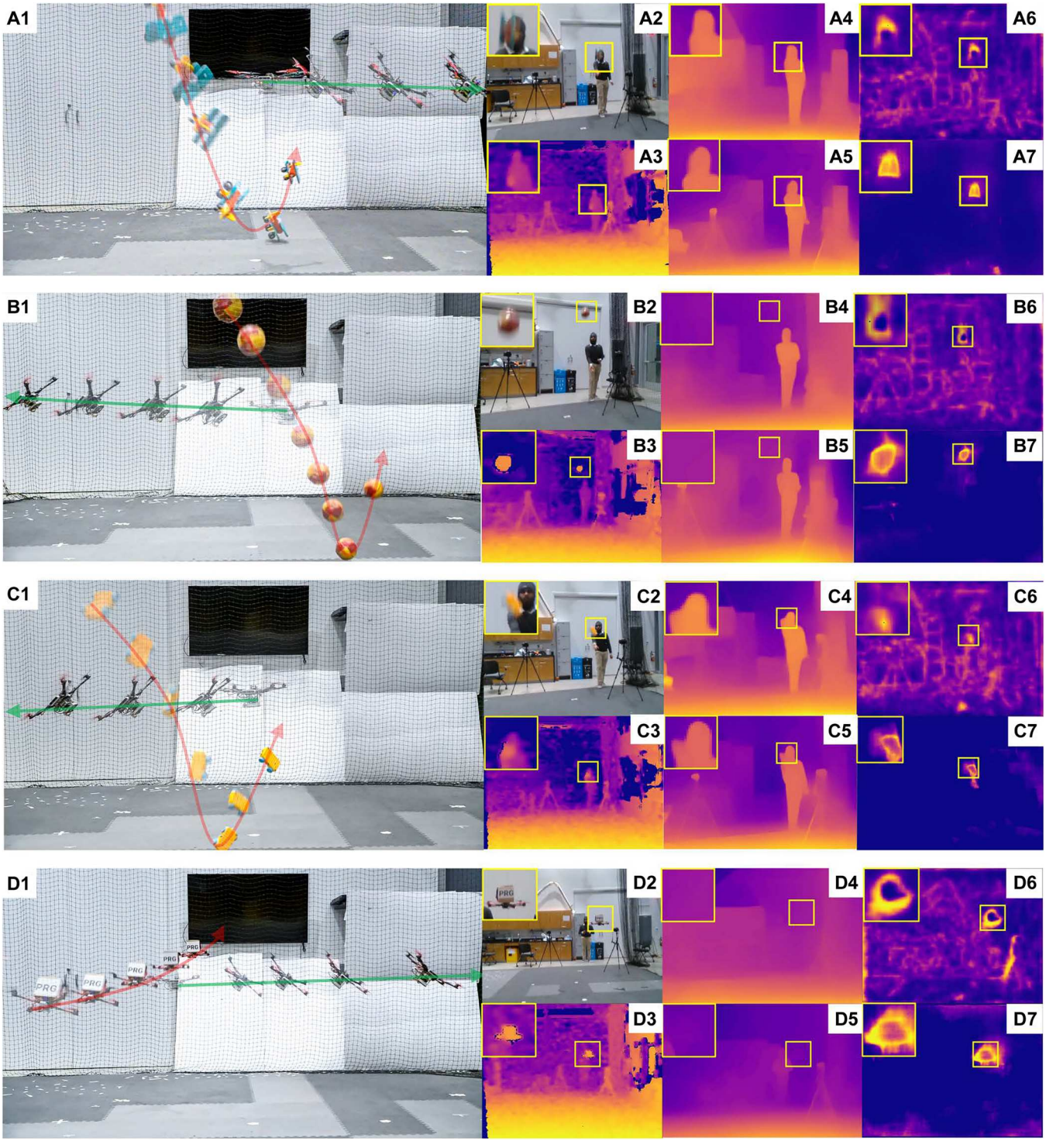

Fig. 2. A sequence of images of quadrotor dodging objects. Dodging (**A**) airplane, (**B**) ball, (**C**) cart, and (**D**) drone. Here, the object and quadrotor transparency shows the progression over time. Red and green arrows indicate object and quadrotor directions, respectively. In each subfigure, the outputs are shown in the following order [using subfigures of (A) as examples]: (A1) image sequence of dodging, (A2) RGB image as seen by the quadrotor, (A3) D435i depth image, (A4) MiDaS-S output, (A5) MiDaS output, (A6) OccMask, and (A7) Ajna. The color map used in all the depth images is plasma, where blue color represents far and yellow is close. The color map for occlusion and uncertainty map is inverse plasma, where blue color represents lower uncertainty/occlusions and yellow represents higher uncertainty/occlusions. The yellow boxes show the zoomed-in view of the object. The color map is consistent across all figures in this paper.

define DR as

$$\mathrm{DR} = \frac{\text{Num. success}}{\text{Num. trials}}; \text{Success} := IoU \geq 0.5 \quad (2)$$

where $IoU = (\mathscr{D} \cap \mathscr{G})/(\mathscr{D} \cup \mathscr{G})$. Here, $\mathscr{D}$ is the predicted mask and $\mathscr{G}$ is the ground truth mask.

Table 1. Quantitative evaluation for dodging obstacles and flying through gaps.

Method	DR (%) ↑	Run time (ms) ↓	FLOPs (G) ↓	Number of parameters (M) ↓
Dodging dynamic obstacles				
D435i* (*61*)	100.0	1.0	–	–
MiDaS-S (*40*)	0.0	12.0	43.7	21.1
MiDaS (*40*)	3.1	137.8	1052.9	344.6
EVDodgeNet† (*39*) (SegNet)	40.4	2.5	0.2	0.03
EVDodgeNet† (*39*) (DB + H + SegFlowNet)	90.7	11.0	5.2	3.6
OccMask (*42*)	74.8	31.4	62.7	20.6
Ajna (ours)	89.2	10.1	6.3	2.7
Flying through unknown shaped gaps				
GapFlyt (PWC-Net) (*45*, *62*)	94.2	91.0	90.8	8.75
GapFlyt (FlowNet2) (*45*, *63*)	93.0	124.0	24836.4	162.5
GapFlyt (SpyNet) (*45*, *64*)	74.0	70.0	149.8	1.2
D435i* (*61*)	100.0	1.0	–	–
MiDaS-S (*40*)	0.0	12.0	43.7	21.1
MiDaS (*40*)	30.1	137.8	1052.9	344.6
OccMask (*42*)	56.2	31.4	62.7	20.6
Ajna (ours)	91.0	10.1	6.3	2.7

*Uses depth images. †Uses event sensor images and results are taken from (*39*).

Navigating through unstructured environments

In this experiment, we present a method to navigate a quadrotor toward a goal direction through different unstructured environments: indoor forest, boxes, and a photorealistic simulated forest. We identified the safe region in the current image to avoid obstacles while also moving toward the goal by dynamically weighing the contributions of the local planner (avoiding obstacles) and global planner (going toward the goal). Once the weighted intermediate goal direction was obtained, a control policy was deployed on the quadrotor to change the current heading direction using a proportional-integrative-derivative (PID) controller to reach the goal while avoiding collisions. The current desired direction $\tilde{\mathbf{v}}_g$ was obtained as the weighted sum of the goal direction $\mathbf{v}_g$ and free path direction $\mathbf{v}_{\text{free}}$. $\tilde{\mathbf{v}}_g$ acts as the global planner, and $\mathbf{v}_{\text{free}}$ acts as the local planner. This policy was based on the policy from (*44*) with minor modifications explained next.

Consider a small neighborhood $\mathscr{N}$ on the image plane centered around the intersection of the goal direction vector and the image plane. Let $\mathbf{v}_{\text{free}}$ be the geometric center of the largest free space in the neighborhood $\mathscr{N}$. We consider higher values of uncertainty in optical flow to be closer to the camera because the rotation-compensated optical flow is inversely proportional to depth (*45*). Now, let Z_{close} be the closest depth value in $\mathscr{N}$. Let $Z_{\text{o,i}}$ denote the depth value in $\mathscr{N}$ in different directions i. We chose the second most "unsafe" region such that $Z_{\text{o}} = \min(Z_{\text{o,i}} > Z_{\text{close}} + \delta)$, where δ is a user-defined heuristic. This formulation is inspired by the classical receding horizon planner (*46*). The final control policy is given by

$$\tilde{\mathbf{v}}_g = (1 - w)\mathbf{v}_g + w\mathbf{v}_{\text{free}}; w \in [0, 1] \quad (3)$$

$$\mathbf{e}(t) = \tilde{\mathbf{v}}_g(t) - \tilde{\mathbf{v}}_{\text{curr}}(t) \quad (4)$$

$$\mathbf{u}(t) = K_p\mathbf{e}(t) + K_i\int_0^{\tau}\mathbf{e}(\tau) + K_d\frac{d\mathbf{e}(t)}{dt} \quad (5)$$

$$w = \frac{1}{1 + e^{(-Z_o/Z_{\text{close}})}}; Z_{\text{close}} = \min Z(x, y)\forall\{x, y\} \subset \mathscr{N} \quad (6)$$

where $\tilde{\mathbf{v}}_{\text{curr}}$ is the estimated current heading direction. Note that the goal vector is dominated by $\mathbf{v}_{\text{free}}$ when the foreground element is very close, which means $w \to 1$, and by global goal vector $\mathbf{v}_g$

Table 2. Quantitative evaluation for navigating through unstructured environments.

Method	SR (%) ↑	Average path length increase (%) ↓	Average safe point error (px.) ↓	Run time (ms) ↓	FLOPs (G) ↓	Number of parameters (M) ↓
MorphEyes* (*44*)	99.0	0.6	1.1	2.5	–	–
MiDaS-S (*40*)	32.0	7.8	6.8	13.4	43.7	21.1
MiDaS (*40*)	97.0	1.0	1.1	139.2	1052.9	344.6
OccMask (*42*)	0.0	–	40.0	35.3	62.7	20.6
Ajna (ours)	92.0	2.7	4.1	11.6	6.3	2.7

*Uses stereo camera images. All other methods above use RGB image(s) as their input.

when there are no obstacles nearby, which means $w \rightarrow 0$. We deployed this policy in all three aforementioned environments. We controlled yaw velocity and net thrust vector to achieve the desired direction. In addition to (*44*), where both "safe" and "unsafe" regions are used, our method also weighed in the second most unsafe region.

Indoor forest environment

An indoor forest environment was constructed using cylindrical play tunnels of diameter 0.48 m and height 1.83 m (Fig. 3A). These tunnels were scattered over the space of 7 m by 5 m in an unstructured manner. The quadrotor successfully navigated through the indoor forest environment at an average speed of 3.2 m s^{-1} with an SR of 86% over 50 trials in different configurations.

Boxes environment

We constructed cubes or "boxes" using yoga mats of dimensions 0.6 m by 0.6 m for our unstructured environment (Fig. 3B). The rectangular mat in the middle had dimensions 1.37 m by 0.6 m. This environment was constructed in the same area as mentioned in the previous section. The quadrotor successfully navigated through the box environment at an average speed of 1.6 m s^{-1} with an SR of 82% over 50 trials. Figure 3 shows a comparison of our results with Intel D435i, MiDaS-S, MiDaS, and OccMask.

Simulated forest environment

We rendered RGB images in Blender 3D software in a photorealistic forest scene (Fig. 4B). Figure 4A represents a simplified version of the forest from the top view. The camera was mounted on a quadrotor that followed a differential flatness model for its controller. Our simulator used a Cycles rendering engine that used a ray-tracing algorithm to produce images that the quadrotor saw. The images were rendered at 30 frames/s at a resolution of 640 pixels by 480 pixels. The scene consisted of static trees with varying sizes, shapes, and textures. We tested both our perception and control algorithm in simulation. We compared Ajna with MorphEyes (*44*), MiDaS-S, MiDaS, and Comas on SR, average safe point error, run time, and average path length increase from path length using ground truth depth in Table 2 and Fig. 4. We defined the SR as the ratio of the number of successful trials of drones navigating through the scene without any collisions to the total number of trials. The average safe point error was the difference between the ideal safe point obtained from the ground truth depth map to the safe point computed through the respective method. The safe point was defined as the safest point on the image plane where the drone should be headed. This was adapted from (*44*). Note that the safe point error was computed in pixels on the image plane. Average path length increase was defined as the percentage increase in the average distance taken by the particular method as compared with the distance traversed when using the ground truth depth map. Furthermore, we randomized our scene of size 40 m by 20 m with a uniform random distribution of various trees of diameter ranging from 0.6 to 0.9 m with a minimum separation distance of 3 m between two trees. We successfully tested our method to achieve an SR of 92% over 100 trials with an average speed of 4.8 m s^{-1}.

Flying through an unknown gap

In this experiment, we present a method to detect and navigate through a gap of unknown shape and location using only a monocular camera. The procedure of flying through a gap involves two key

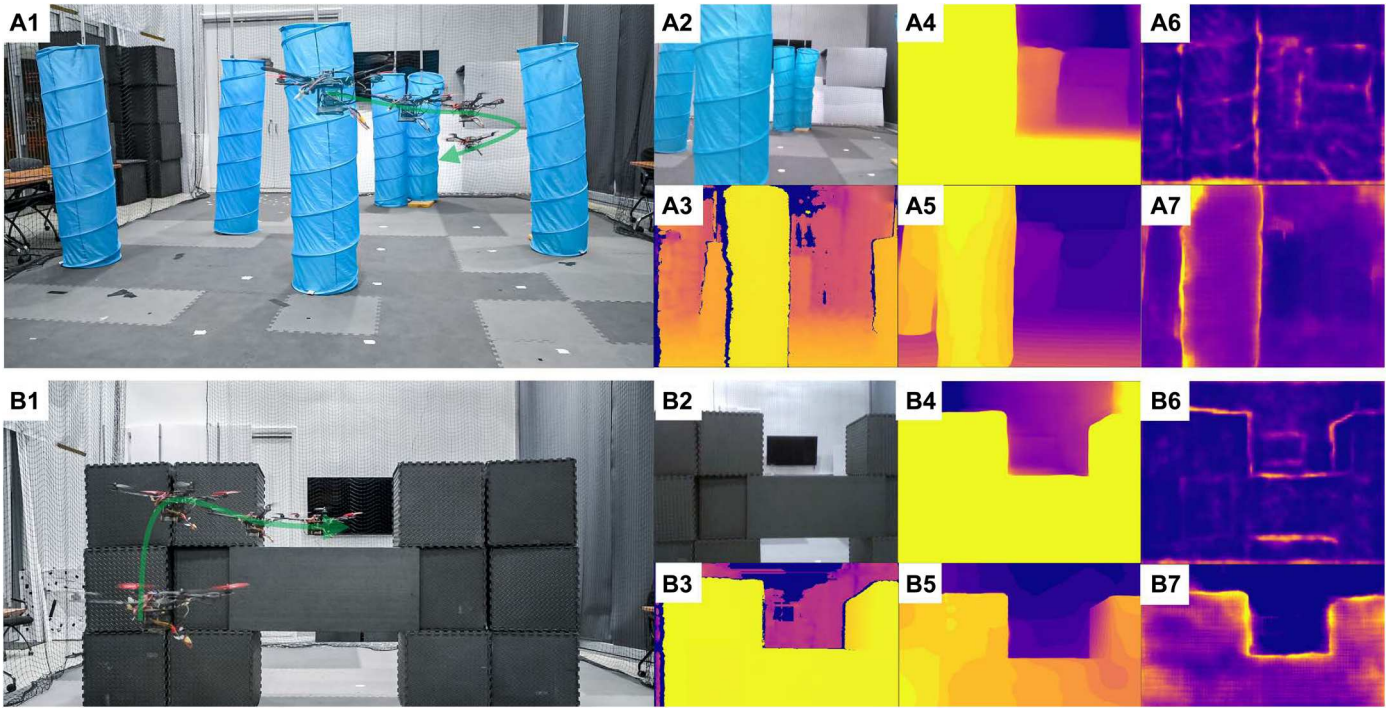

Fig. 3. A sequence of images of the quadrotor navigating through cluttered environments. (**A**) Indoor forest. (**B**) Boxes. Here, the object and quadrotor transparency shows the progression of time. The green arrow indicates the quadrotor direction. In each subfigure, the outputs are shown in the following order [using subfigures of (A)]: (A1) image sequence of navigation, (A2) RGB image as seen by the quadrotor, (A3) D435i depth image, (A4) MiDaS-S output, (A5) MiDaS output, (A6) OccMask, and (A7) Ajna.

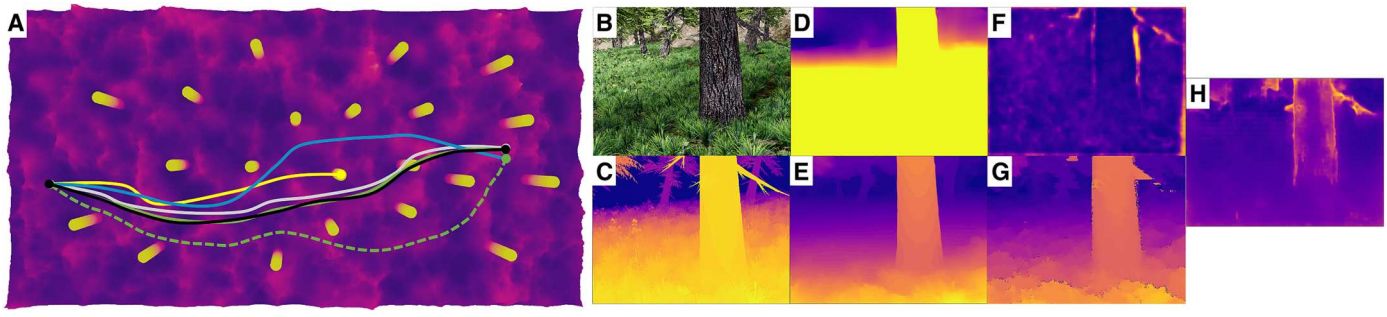

Fig. 4. Comparison of various methods to navigate through a simulated realistic forest scene. (**A**) The scene from the top view with paths overlaid (direction of travel is left to right). The legend is as follows: white, ground truth depth; green, MiDaS; dashed green, MiDaS-S; yellow, OccMask; black, MorphEyes; blue, Ajna (ours). (**B**) Sample RGB image as seen by the quadrotor, (**C**) ground truth depth, (**D**) MiDaS-S output, (**E**) MiDaS output, (**F**) OccMask output, (**G**) MorphEyes output, and (**H**) Ajna output.

steps: detection of the gap and aligning/flying through the gap. In the first step, we used the active vision (*47*, *48*) philosophy to perform an "exploratory" maneuver just like in (*45*). We then used the fact that the uncertainty of optical flow inside the gap would be much higher than outside. This is intuitively true because the number of occlusions and accretions caused inside the gap would be much larger due to the parallax effect. This disparity in uncertainty enabled us to detect the gap in an effortless manner using basic morphological operations. Once the gap was detected, we tracked the contour indirectly by tracking the foreground (the part outside the gap) and background regions (the part inside the gap) separately. We then propagated the gap contour shape using the focus of expansion (FOE) constraints as in (*45*) to fly through the safe point $\mathbf{x_s}$. Further, we tracked the gap and visually servoed toward the safe point by actively switching between background and foreground as necessary (*45*). We describe the experimental setup and results next.

The experimental setup contained a rigid scene with two near-planar "wall" surfaces, namely, the foreground and the background (Fig. 5). The foreground contained an unknown-shaped gap. The foreground wall had newspaper stuck on it to add texture to the scene. The background wall had real-life features and hence was not augmented with additional features. The average distance from the initial position of the quadrotor to the foreground and background was 3.0 and 7.2 m, respectively. For the detection of the gap, we bumped up the proportional gain of the position controller (*49*, *50*) momentarily to invoke random exploratory maneuvers. This could easily be replaced by a fixed diagonal line trajectory, such as presented by Sanket *et al.* (*45*). Over this maneuver, a number of images were captured. The uncertainty in optical flow between these sets of images was used to detect the gap. Similar to (*45*), once the gap was detected, the foreground and background regions were tracked using Kanade-Lucas-Tomasi tracker (*51*) and the quadrotor servoed toward the gap. We compared our results using optical flow methods, depth-based methods, and occlusion-based methods on metrics of DR, run time, FLOPs, and number of parameters in Table 1. The DR was the same as defined in the "Dodging dynamic obstacles" section. We obtained a DR accuracy of 91% over 100 trials across four different unknown-shaped gaps with a minimum tolerance of just 8 cm. The optical flow methods for detecting the gap were based on GapFlyt (*45*) with the input to the algorithm coming from different optical flow networks, such as PWC-Net, FlowNet2, and SpyNet. In the depth-based methods (D435i, MiDaS, and MiDaS-S), the gap was obtained by clustering depth values into foreground and background, followed by obtaining the region of largest disparity between the values. For the occlusion-based method OccMask, we used simple morphological operations after thresholding the values to obtain the gap.

Segmentation of object pile

We studied the possibility of using Ajna for foreground-background segmentation tasks. In this experiment, we assumed that the object set was placed on a planar surface. We used the activeness of the robot to obtain uncertainty due to occlusions. The camera moved actively to take two snapshots from different views of the same scene to compute the uncertainty Υ. The boundary between the foreground and background is correlated to the occlusion between two frames. These occluded regions give rise to high uncertainty. We conducted our experiments on GrassMoss (Fig. 6A), Rocks (Fig. 6B), and YCB (Fig. 6C) datasets as mentioned in (*36*) on 30 unique configurations per dataset. We qualitatively compared Ajna's output with Intel D435i, Mask-RCNN (*52*), PointRend (*53*), MiDaS-S, MiDaS, and OccMask. Note that Ajna and OccMask are active approaches and rely on two images to segment the foreground and background. For segmenting in the D435i image, we used a plane-fitting algorithm on the depth map and removed it for the background subtraction. Object segmentation methods like Mask-RCNN and PointRend rely on features and texture maps. They segment the object cluster but also segment some objects on the background texture (see traffic lights and trees being segmented in Fig. 6B3). In MiDaS-S and MiDaS, we segmented the background by thresholding the depth value on the predicted output image. In OccMask, the occlusions were visible on the objects, but OccMask also results in artifacts in the background textures as well (see Fig. 6A7). In Fig. 6A8, Ajna gives an estimate of where the object pile is located. Just using Υ directly does not provide segmentation of the object pile. It acts as an attention mechanism to "show" where the object pile is but is seldom sufficient to provide segmentation of the pile. This can, however, be used as an initialization step for interactive segmentation as proposed in (*36*) for segmenting out zero-shot or unknown objects/samples. If one is segmenting known objects, one can use instance segmentation–based methods, such as Mask-RCNN or PointRend, that identify objects from a learned database, and one can extract the required objects directly. An alternative approach is to use a method that provides depth like the D435i or infers depth using a neural network

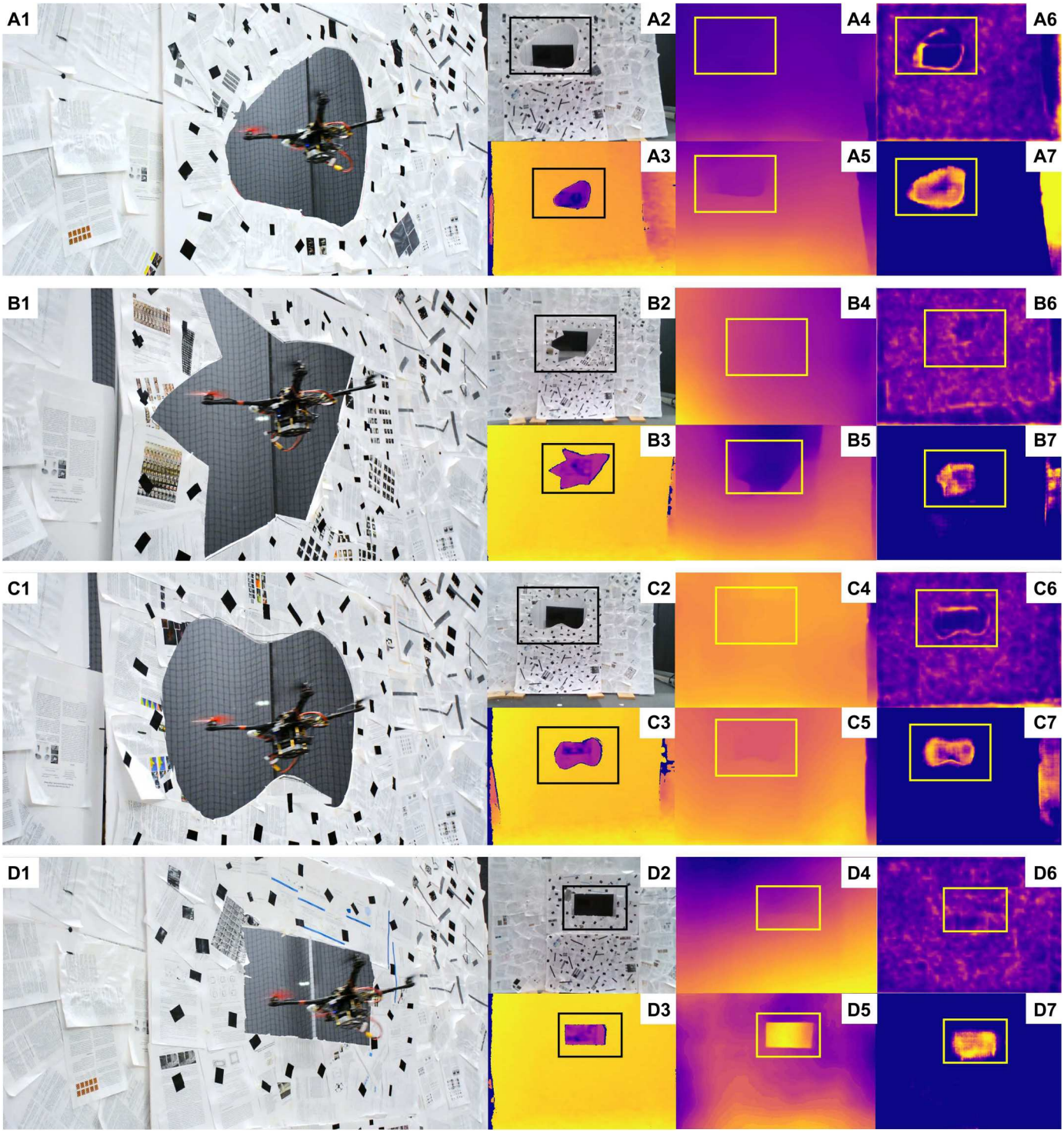

Fig. 5. Image of quadrotor flying through unknown gaps. (**A**) Egg. (**B**) Goku. (**C**) Infinity. (**D**) Rectangle. In each subfigure, the outputs are shown in the following order [using subfigures of (A)]: (A1) image of flight through the gap, (A2) RGB image as seen by the quadrotor, (A3) D435i depth image, (A4) MiDaS-S output, (A5) MiDaS output, (A6) OccMask, and (A7) Ajna. The black or yellow boxes in the images show the window location.

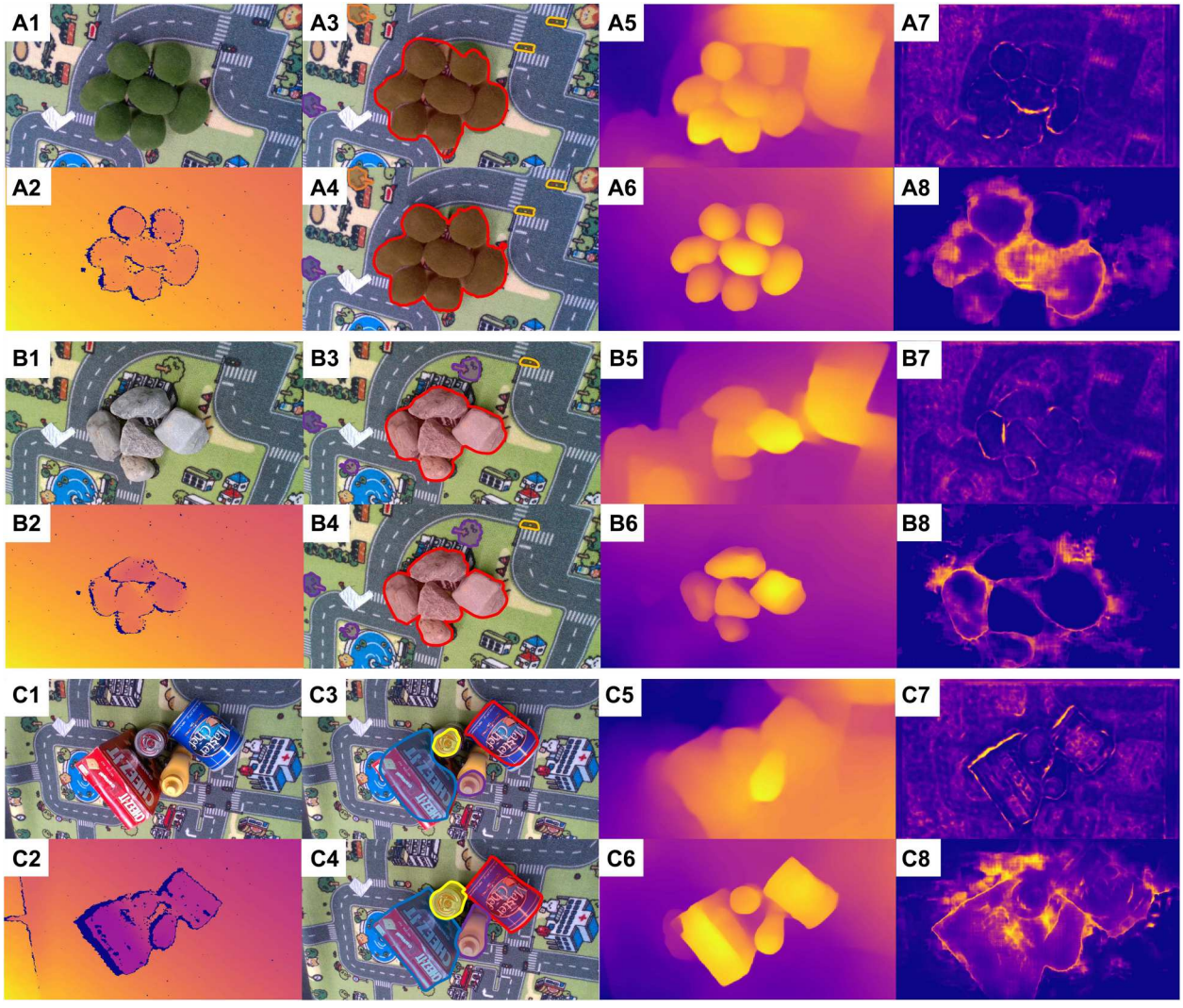

Fig. 6. Outputs for segmentation experiments using various methods on different datasets. (**A**) GrassMoss. (**B**) Rocks. (**C**) YCB. In each subfigure, the outputs are shown in the following order [subfigures of (A)]: (A1) RGB image as seen by the robot, (A2) D435i depth image, (A3) Mask R-CNN output, (A4) PointRend output, (A5) MiDaS-S output, (A6) MiDaS output, (A7) OccMask, and (A8) Ajna output. Different colors in (A3) and (A4) show different objects with different labels being detected by the instance segmentation.

like MiDaS or MiDaS-S, and one can segment the objects that "stand out" of the plane of the table. None of the later methods generalizes to novel/zero-shot objects because they rely on learning to predict outputs based on textures rather than geometry. Ajna, when combined with NudgeSeg, can be used as a method to learn novel objects to train methods like Mask-RCNN, PointRend, or MiDaS. This can further enable the operation of robots in the wild. Furthermore, it is simple to know whether an object is zero-shot by "looking" at the epistemic uncertainty of predictions, and this presents an interesting avenue for future work.

Network speed on different hardware

In this section, we test the Ajna network's inference speed on various computing platforms such that they can be deployed on various-sized quadrotors (as low as 120 mm as shown in Fig. 1). The inference time for Ajna on an Intel i9 CPU, NVIDIA Titan Xp GPU, NVIDIA Jetson TX2, Intel NCS2, and Google Coral TPU is 140.7, 9.1, 49.0, 268.9, and 34.4 ms, respectively. For a detailed comparison of various network architectures for related tasks, we refer the readers to (*54*). Here, the time on the Intel i9 CPU and NVIDIA Titan Xp GPU is presented to act as a baseline and cannot be deployed on small robots. The NVIDIA TX2 used in our experiments performs well for small aerial robots. Specialized neural network accelerators such as the Intel NCS2 and the Google Coral TPU are tailor-made for tiny aerial robots. Our work and its unifying conceptualization will enable a multitude of tasks on tiny aerial robots when coupled with such neural network accelerators.

DISCUSSION

Perception on robots for various autonomous operations is generally centered around building a 3D representation of the scene using mature simultaneous localization and mapping (SLAM) or odometry algorithms. Various sensor suites have been used to accomplish

the aforementioned tasks. Central to such methods are fusing multiple noisy measurements from either single or multiple sensors to obtain more accurate results. This has facilitated the modeling and utilization of uncertainty of various sensing and processing modalities. However, these uncertainties also have latent informational cues that are rarely used in robots. Furthermore, when one is building a parsimonious solution to various robotics problems to comply with the size, weight, area, and power (SWAP) constraints to perform all sensing and computation on board, one has to use every bit of informational cue available. This will further lead to unifying various robotic tasks on a parsimonious agent.

In our work, we present one such unifying framework for four common robotic tasks by a resource-constrained aerial robot: dodging dynamic obstacles, navigating through a cluttered environment, flying through gaps, and segmentation of object piles. All these methods rely on the activeness of the agent, in other words, its ability to obtain images from different views. On the basis of the heteroscedastic aleatoric uncertainty Υ of optical flow, we obtained object boundaries due to accretions and deletions that are central to performing "depth-based segmentation" that was used in solving the aforementioned four problems using a simple perception stack. Note that Υ of optical flow has additional informational cues that aid the detection of dynamic obstacles and navigation problems: motion blur, which leads to optical flow being ill-defined. In the dynamic obstacle case, when the obstacle is moving much faster than the robot, the amount of motion blur leads to an ill-posed estimation of optical flow, which gives rise to high Υ. This has the same effect as the properties of event cameras: Dynamic obstacles stand out because of producing a large number of events, which is mimicked by high values of Υ.

In the navigation case, the closer the object, the more amount of motion blur it will have because of slight movement while the image/frame is being acquired, coupled with the fact that the optical flow of the closer objects is higher, so the inherent value of Υ will be high. In a qualitative sense, Υ has information from both worlds: depth and motion boundaries. This is exactly what we observe from the results from Fig. 4 and Table 2. Results using Ajna are in the middle (in terms of SR, path length, and safe point error) of using depth-based methods and occlusion methods. In the occlusion-based method OccMask, there is no trivial way to solve the boundary assignment problem, that is, which part of the high occlusion region belongs on the tree. This is because the informational cue about depth is missing in this representation. However, in depth-based methods, the boundary assignment is trivial, but the computational cost to obtain the depth map is much higher (about seven times as compared with Ajna). Furthermore, works such as (*55*) avoided processing holes in optical flow, especially near the FOE, which led to a very inefficient solution. Our method can be easily incorporated into such methods to avoid holes in optical flow as shown in Fig. 7A. Particularly in the cases where optical flow values are small near the FOE even though the obstacle is present, our method can provide robustness. Notice how Υ is high even though the optical flow magnitude is low near the FOE because it is on the tree trunk. Such approaches can lead to minimal representations for high-speed agile flight through the forest and can advance the work of (*56*, *57*).

We obtained and analyzed uncertainty in various different settings (see Fig. 7, A to G). Figure 7B shows how uncertainty behaved in the case when only light illumination changed [blinking light-emitting diode (LED)], whereas Fig. 7C represents uncertainty in the case of both light changes as well objects under motion. Figure 7D shows that our method was invariant to black-box adversarial attacks (*58*). We observe that in Fig. 7D3, the input image had an adversarial patch superimposed, yet this did not affect our uncertainty predictions. Figure 7 (E to G) illustrates three different experiments: flying through unknown gaps, navigating in static environments, and dodging dynamic objects. For Fig. 7 (E to G) experiments, the first column represents the depth map of the scene (black represents 0 m; white represents 4 m), the second and third columns are the consecutive input images to our neural network, and the last column is the predicted uncertainty. We observed that uncertainty at depth boundaries is almost always far greater than the uncertainty at low-texture regions. This study was performed on a variety of real-world textures with varying the number of texture components (variations in the smoothness of the scene including color-flat or low texture regions) in over 7000 images and 100 textures in three scenarios of detection of unknown gaps (50 differently shaped gaps; one such example is shown in Fig. 7E), static obstacle environments (one such example is shown in Fig. 7F), and dynamic obstacles (one such example is shown in Fig. 7G).

We can also analyze how neural networks see depth using a single image (*59*) versus two or more images. Let us shed some light on the simulation forest experiment. Figure 4B represents the input RGB image of a simulated forest, and Fig. 4C represents the ground truth depth map. If we look closely, in MiDaS output (Fig. 4E), the branch of the tree is missing. In addition, notice that the tree branch is present in OccMask and Ajna outputs (Fig. 4, F and H, respectively). Note that MiDaS used a single image to predict depth, whereas OccMask and Ajna required two images to predict its output. From Table 1, we can conclude that MiDaS has about 128× more parameters as compared with our Ajna, and yet it fails to capture small details like the tree branch. Neural networks often fail to capture this subtle information from a single image, irrespective of the size of the pre-trained model. For robotics applications such as navigation, this subtle information is crucial to avoid these obstacles. Thus, the utilization of multiple images (rather than a single image) to predict quantities like depth can be more beneficial for such applications.

Limitations and future work

It is important for us to also discuss a few limitations of using just optical flow uncertainty for the tasks described before. Uncertainty can be high because of a multitude of reasons, such as depth boundaries, color-flat regions, extreme brightness changes, and blinking lights. In general, we observed that (more examples can be found in the Supplementary Materials) depth boundaries or dynamic obstacles have higher uncertainties than other factors and can be used for navigation. Nevertheless, uncertainty from multiple sources (such as optical flow and surface normals) can be used to disambiguate among various scenarios and can be an interesting avenue for future work. Furthermore, using uncertainty might not present a complete solution but can act as a safety/attention mechanism that highlights the part of the image that needs more processing/attention. From a logical perspective, high uncertainty and the solution of a task are not necessary and sufficient for each other. As an example, let us take the task of detecting an independently moving object (IMO). If P is the proposition, where P = "There is a high uncertainty in this region" and Q is the proposition such that Q =

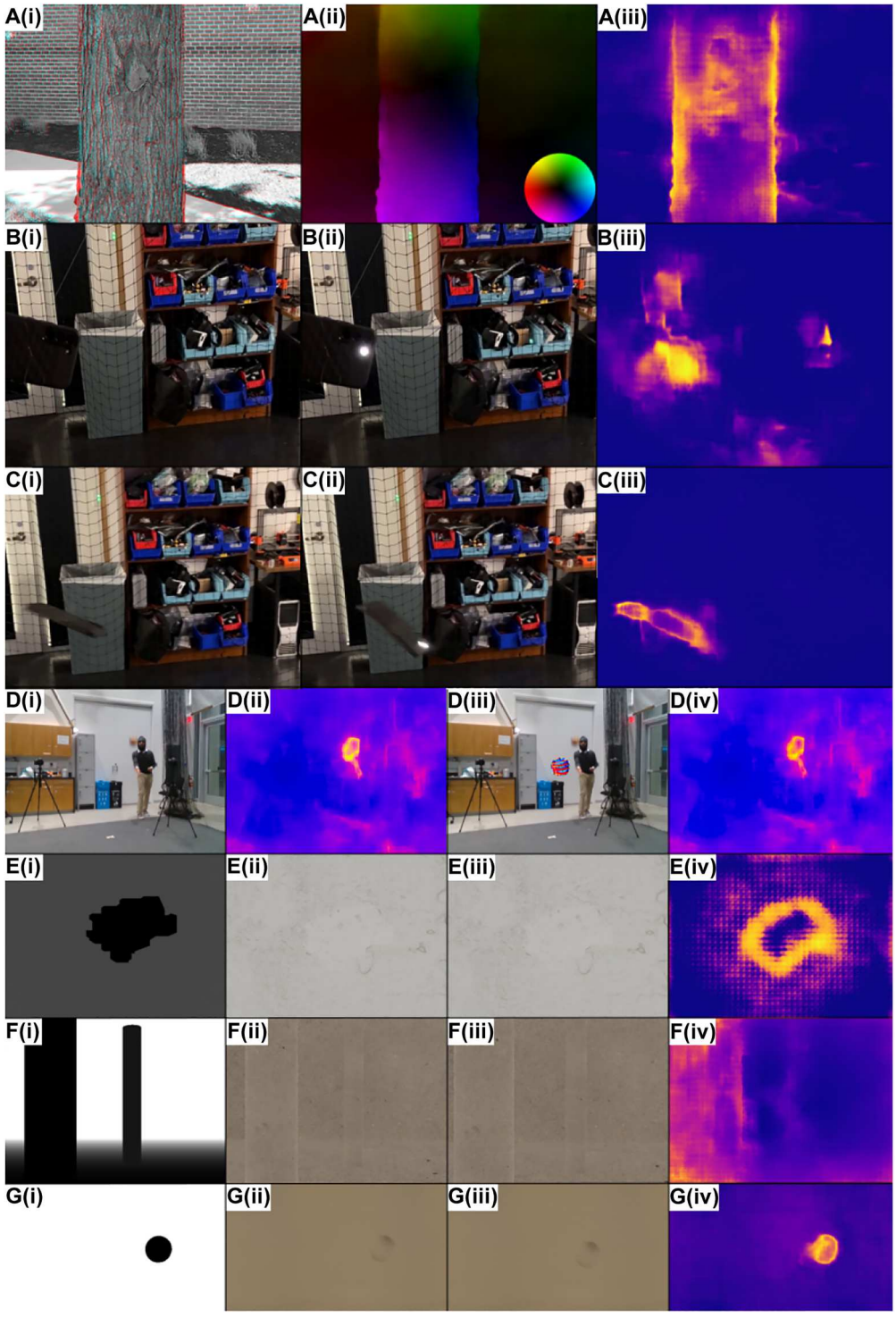

Fig. 7. Uncertainty predictions in extreme scenarios. (**A**) (i) Input image pair as an anaglyph, (ii) optical flow with the color map shown as inset, and (iii) Ajna's predicted uncertainty. Despite low $\dot{\mathbf{p}}$ in the highlighted white region, the quadrotor needs to dodge this area. This is correctly predicted as high Y. This is a common example where Y provides additional information over $\dot{\mathbf{p}}$. (**B**) (i to iii) Input image frames and $\dot{\mathbf{p}}$ under blinking LED without motion. (**C**) (i to iii) Input image frames and under blinking LED with motion. (**D**) (i to iv) Image input, predicted Y, image input with flow attack, and predicted Y under attack. (**E**), (**F**), and (**G**) are experiments of flying through gaps, flying through a forest, and detecting dynamic obstacles. Left to right: ground truth depth (white is 4 m and black is 0 m), input images 1 and 2, predicted $\mathbf{Y}$.

"There is an IMO there," then $Q \Rightarrow P$ but $P \nrightarrow Q$. We could have high uncertainty because of a blinking light, for example, or because there is an object nearby. Therefore, in principle, a system based on uncertainty estimation will never miss an IMO—It may think, however, in some instances, that there is an IMO when there is not one. This is the price that qualitative minimal perception has to pay. Thinking that there is an IMO at a close-by location is not necessarily a bad thing for a minimal system, because close-by objects could be potential obstacles and should be avoided.

We see several interesting avenues for future work because our work is at a nascent stage. First and foremost, it would be interesting to analyze the aleatoric uncertainty of various sensors commonly used in robotics, such as inertial sensors, LiDARs, SONARs (sound navigation and ranging), and so on. We think that this could be a new way to fuse sensors using informational cues rather than just fusion using raw uncertainty values using Bayesian formulation and its variants. Furthermore, these latent informational cues might help unravel nontrivial and efficient ways to solve common robotics problems in a unified manner. Although we presented analysis and experimental results to showcase that Υ can be used to solve common robotics problems, it is still not clear how a task planner would be built to switch between the aforementioned tasks and how one would disambiguate between the various cases when deployed in the wild. In addition, the use of uncertainty in low-light environments with traditional RGB cameras will largely hinder the performance of obstacle detection. It would be interesting to explore the uncertainty principles for high dynamic range (HDR) or event cameras in the near future.

Overall, our method takes baby steps by presenting a generalized uncertainty formulation that can be used to solve common robotics problems in an unconventional manner. We hope that this can open new doors for robot autonomy at sizes that were not thought possible before by using Υ to aid parsimonious solutions.

MATERIALS AND METHODS

General heteroscedastic aleatoric uncertainty formulation

Let x be the input to a neural network $\mathbb{N}$ with weights W and $\tilde{y}$ be its estimated output (Eq. 7) where the ground truth prediction is given by $\hat{y}$.

$$\tilde{y} = \mathbb{N}(x \mid W) \tag{7}$$

We want to learn weights W to optimize the following problem:

$$\arg\min_{W,\Upsilon} f(\hat{y}, \tilde{y}) \text{ such that } \Upsilon = k(f(\hat{y}, \tilde{y}), x) \tag{8}$$

Here, f is a distance metric between the predicted $\hat{y}$ and ground truth $\hat{y}$, and Υ is a monotone function k of heteroscedastic aleatoric uncertainty of the underlying probability distribution $p(x, \tilde{y} \mid W)$ with a positive correlation to the expected error/risk. Formally, the correlation between two random variables X and Y is given by the Pearson correlation $\rho_{X,Y}$ in Eq. 9, where E represents the expectation operator.

$$\rho_{X,Y} = \frac{\mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y)}{\sqrt{\mathbb{E}(X^2) - \mathbb{E}(X)^2}\sqrt{\mathbb{E}(Y^2) - \mathbb{E}(Y)^2}} \tag{9}$$

To reiterate, Υ is a function of x and is correlated with the estimated error between $\tilde{y}$ and $\hat{y}$ and is formally defined below:

$$\Upsilon(x \mid W) := h(\mathbb{E}(d(\hat{y}, \tilde{y}))) \text{ such that } \rho_{\Upsilon, f(\hat{y},\tilde{y})} > 0 \tag{10}$$

Here, d and f are distance metrics on a set X such that $f, d : X \times X \rightarrow [0, \infty)$ and satisfy the properties of identity, symmetry, and triangle inequality. Note that Υ need not be the variance of the distribution $p(x, \tilde{y} \mid W)$ but has to satisfy $\rho_{\Upsilon,\nu} > 0$, where ν is the variance (which might be hard to compute for arbitrary distributions). Intuitively, Υ represents the expected error or risk or the absence of confidence in the predicted output. To obtain Υ, which we will call uncertainty for ease of understanding in a self-supervised way, we need to optimize the following function.

$$\arg\min_{\tilde{y},\Upsilon} h(\Upsilon) f(\hat{y}, \tilde{y}) + \lambda g(\Upsilon) \tag{11}$$

In the above optimization function, g is a monotone function of the uncertainty (to preserve the domain order and convexity) and h is a function that inverts the nature of monotonicity of g (such that $\rho_{h,g} < 0$ is satisfied; here, h could also be a function of g). The intuition behind the above formulation is that there is a two-way coupling between Υ and $\tilde{y}$ to ensure that trivial solutions are not encountered and that values are appropriately scaled. The term $h(\Upsilon) f(\hat{y}, \tilde{y})$ scales the value of $f(\hat{y}, \tilde{y})$ appropriately based on the uncertainty per input dimension; this is equivalent to weighing different noisy observations to mimic "outlier rejection" and can be thought of as a loss attenuator. However, this can lead to trivial solutions of $\Upsilon \rightarrow \infty$ (if unbounded) to minimize the loss. To avoid this, a simple penalty on the value of Υ is added using $\lambda g(\Upsilon)$ to counteract exploding values of Υ. This formulation is a generalization of the work in (*7*). The user is free to choose the functions g, h, and f, which can be adapted from domain-specific knowledge. We showed the relationship between f, g, and h in previous works, and it is presented in Table 3. Note that our formulation spurs by intuitively summarizing a large body of prior work across various domains that estimate uncertainty, risk, and/or learned robustness parameter(s). We intuited a trend in the large bodies of past work and formulated a blueprint function that can be used for crafting novel loss functions. To summarize, we unified previous works into a single generalized function. These previous works (Table 3) can be obtained by plugging in specific functional parameters in our formulated equation (Eq. 11).

Note that in our formulation, Υ can mean uncertainty (similar to covariance) or lack of confidence (risk) of any arbitrary distribution. Because for complicated distributions, Υ can be a complicated function of the variance ν, the uncertainty might be qualitative rather than quantitative. However, with careful consideration of functions f, g, h, and λ, Υ can be made to be a quantitative function of ν with simple closed-form solutions. In such scenarios, one can also work toward certifiability of neural networks in a limited domain of training/operating data. Formally, a network is certifiably robust when the error is bounded by a value τ for the prediction of perturbed inputs. If x is the input and x' is the perturbed input, the l_p distance in their respective outputs should be bound within τ, i.e., $\|\mathbb{N}(x|W) - \mathbb{N}(x'|W)\|_p \leq \tau$. We hypothesize that this definition of robustness should also include the network's confidence as an additional constraint. This is analogous to the network "informing us" when it is speculating a failure. Such a formulation needs a thorough

Table 3. Relation to existing works (chronological order).

$f(\tilde{y},\hat{y})$	$h(a)$	$g(a)$	λ	y	Reference
$\|\tilde{y}-\hat{y}\|_2^2$	a^{-2}	$\text{Log}(a^2)$	1.0	Semantic segmentation	(*7*)
$\|\tilde{y}-\hat{y}\|_1$	a	$\text{Log}(a)$	0.2	Monocular depth	(*29*)
$(\tilde{y}-\hat{y})^2$	a^{-1}	$\text{Log}(a)$	1.0	Optical flow	(*8*)
$\|\tilde{y}-\hat{y}\|_1$	a^{-2}	$\text{Log}(a)$	1.0	Optical flow	(*18*)
$\begin{cases} 0.5(\tilde{y}-\hat{y})^2, \\ \text{if } \|\tilde{y}-\hat{y}\| < 1 \\ \|\tilde{y}-\hat{y}\|-0.5, \\ \text{otherwise} \end{cases}$		$\text{Log}(a^2)$	2.0	3D bounding box	(*15*)
$\|\tilde{y}-\hat{y}\|_2^2$	a^{-2}	$\text{Log}(a)$	6.0	Optical flow	(*12*)
$\|\tilde{y}-\hat{y}\|_2^2$	a^{-1}	$\text{Log}(\|a\|)$	1.0	Visual odometry	(*27*)
$\|\tilde{y}-\hat{y}\|_2^2$	a^{-1}	$\text{Log}(a)$	1.0	Dense depth	(*25*)
$\|\tilde{y}-\hat{y}\|_1$	a^{-1}	$\text{Log}(a)$	1.0	Monocular depth	(*26*)
$\|\tilde{y}-\hat{y}\|_1$	$\frac{1}{\log(1+e^{a+\varepsilon})}$	$\text{Log}(1+e^a)$	1.0	Optical flow	(*36*)
$\|\tilde{y}-\hat{y}\|_1$	a^{-1}	$\text{Log}(a)$	$\frac{1}{\sqrt{2}}$	Stereo disparity	(*20*)
$\|\tilde{y}-\hat{y}\|_1$	a^{-1}	$\text{Log}(a)$	1.0	Monocular depth	(*19*)
$\cos^{-1}(\tilde{y}^T\hat{y})$	$-a$	$\text{Log}\left(\frac{1+e^{\pi a}}{1+a^2}\right)$	1.0	Surface normals	(*21*)
$(\tilde{y}-\hat{y})^2$	a^{-2}	$\text{Log}(a^2)$	2.0	Optical flow	(*17*)
$(\tilde{y}-\hat{y})^2$	a^{-2}	$\text{Log}(a^2)$	1.0	Monocular depth	(*23*)
$\|\tilde{y}-\hat{y}\|_2^2$	a^{-2}	$\text{Log}(a^2)$	1.0	Pixel dolor	(*24*)
$(\tilde{y}-\hat{y})^2$	a^{-2}	$\text{Log}(a^2)$	1.0	Monocular depth	(*22*)

mathematical treatment and is beyond the scope of this paper. Furthermore, we see this as a potential direction for interesting future work.

When we use Eq. 11 to learn Υ in a self-supervised way along with $\tilde{y}$ and when both Υ and $\tilde{y}$ are dense, they vary per pixel location $\mathbf{x}$. In practice, the actual loss function that is minimized is given below:

$$\arg\min_{\tilde{y},\Upsilon} \mathbb{E}(h(\Upsilon_{\mathbf{x}})f(\hat{y}_{\mathbf{x}},\tilde{y}_{\mathbf{x}})) + \lambda\mathbb{E}(g(\Upsilon_{\mathbf{x}}))\forall \mathbf{x} \tag{12}$$

The above equation models the distribution given below (assuming that we are minimizing the negative log-likelihood in Eq. 12).

$$p(\hat{y} \mid \tilde{y},\Upsilon) \propto \prod_{\forall \mathbf{x}} \exp(-h(\Upsilon_{\mathbf{x}})f(\hat{y}_{\mathbf{x}},\tilde{y}_{\mathbf{x}}))\exp(-g(\Upsilon_{\mathbf{x}})) \tag{13}$$

In the rest of this paper, Υ refers to heteroscedastic aleatoric uncertainty unless otherwise specified, and we refer to heteroscedastic aleatoric uncertainty as just uncertainty unless specified otherwise for simplicity.

Informational cues from uncertainty Υ

The answer to the question "what informational cues does the uncertainty hold?" lies hidden inside the question "what is this uncertainty of?" For robot autonomy, we generally estimate one or more of the following quantities: (i) optical flow, (ii) monocular/stereo depth, (iii) surface normals, and (iv) semantic segmentation. Hence, in our discussion, we will focus on these three quantities in the following subsections.

Uncertainty of optical flow

Optical flow at a pixel located at $\mathbf{x} = [x\ y]^{\mathrm{T}}$ is denoted as $\dot{\mathbf{p}}_{\mathbf{x}}$ and is given by

$$\dot{\mathbf{p}}_{\mathbf{x}} = \frac{1}{Z_{\mathbf{x}}}\begin{bmatrix} xV_z - V_x \\ yV_z - V_y \end{bmatrix} + \begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix}\Omega \tag{14}$$

Here, $V = [V_x\ V_y\ V_z]^T$, $\Omega = [\Omega_x\ \Omega_y\ \Omega_z]^T$ denotes the 3D linear and angular velocities of the camera, respectively. $Z_{\mathbf{x}}$ denotes the depth at a pixel $\mathbf{x}$. To gather insight into when a high uncertainty would be obtained, we need to revisit the mathematical definition of optical flow. The optical flow $\dot{\mathbf{p}}_{\mathbf{x}}$ essentially is giving us the matching of each pixel $\mathbf{x}$ between two images and is obtained by solving the brightness constancy equation

$$\arg\min_{\dot{\mathbf{p}}_{\mathbf{x}}} \mathscr{I}_t(\mathrm{x}) - \mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathbf{p}}_{\mathbf{x}}) \mid \tag{15}$$

Here, $\mathscr{I}_t(\mathrm{x})$ and δt denote the brightness of the point at $\mathbf{x}$ at time t and small time increment, respectively. In practice, because a single point sample at $\mathbf{x}$ would be too noisy to match, a small brightness over a small patch neighborhood $\mathscr{N}$ is matched using some function of the difference in patch brightness denoted by f (this

could be as simple as the sum) and is given as

$$\arg\min_{\dot{\mathrm{p}}_\mathrm{x}} f(\mathscr{I}_t(\mathrm{x}) - \mathscr{I}_t(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x})) \mid \forall \mathrm{x} \in N \tag{16}$$

The robustness of the estimated optical flow $\dot{\mathrm{p}}_\mathrm{x}$ to noise is defined by the Lipschitzness of the estimate with respect to the noisy/perturbed estimate and is given as

$$d(f(\dot{\mathrm{p}}), f(\dot{\mathrm{p}}+\nu)) \leq Kd(\dot{\mathrm{p}}, \dot{\mathrm{p}}+\nu) \tag{17}$$

Here, ν is the noise in the estimate, K is a positive constant, and d is a distance metric. A smaller value of K indicates a lower sensitivity. Now, let us look at what condition will cause a small error on the estimated flow to affect the brightness matching score significantly. To understand this phenomenon mathematically, let us look at the mutual information (Eq. 18) between the distributions of optical flow with and without noise. For $\|\nu\| \to 0$, the mutual information $I(f(\dot{\mathrm{p}}_\mathrm{x}); f(\dot{\mathrm{p}}_\mathrm{x}+\nu))$ should be maximum.

$$I(f(\dot{\mathrm{p}}_\mathrm{x}); f(\dot{\mathrm{p}}_\mathrm{x}+\nu)) = D_{\mathrm{KL}}(P_{f(\dot{\mathrm{p}}_\mathrm{x}), f(\dot{\mathrm{p}}_\mathrm{x}+\nu)} \| P_{f(\dot{\mathrm{p}}_\mathrm{x}+\nu)} \otimes P_{f(\dot{\mathrm{p}}_\mathrm{x}+\nu)}) \tag{18}$$

Here, $f(\dot{\mathrm{p}}_\mathrm{x})$ denotes the matching score from Eq. 16, P_a denotes the marginal distribution of a, and $\otimes$ is the operator that multiplies two marginal distributions. Now, because the neighborhood $\mathscr{N}$ is small, the deviation of x, y inside the neighborhood is small. V, Ω are intrinsic camera properties and do not depend on the scene. The only geometric variable left that can affect the distributional shift of $f(\dot{\mathrm{p}}_\mathrm{x}+\nu)$ away from $f(\dot{\mathrm{p}}_\mathrm{x})$ even when ν is small is large changes to $Z_\mathbf{x}$. This means that the depth varies a lot spatially with small changes to location, i.e., $\|\nabla_\mathrm{x} Z\|$ is large. These are depth discontinuities or edges. They are generally dominated by object boundaries. Hence, a large uncertainty is correlated with object boundaries.

The keen reader might also think about the fact that the Z is a latent variable that implicitly influences Υ, but the appearance of the image should also be affecting it directly. Intuitively, should Υ not have high values for areas that cannot be matched? Indeed this is true; this is the other case where large "flat" (uniform color) regions will lead to large uncertainty because of the absence of nondistinct features. Here, a feature would be called distinct based on the structure tensor M

$$M = \begin{bmatrix} \nabla_x \mathscr{I}^2 & \nabla_x \mathscr{I} \nabla_y \mathscr{I} \\ \nabla_x \mathscr{I} \nabla_y \mathscr{I} & \nabla_y \mathscr{I}^2 \end{bmatrix} \tag{19}$$

Let λ_1 and λ_2 be the eigenvalues of M; then, if $\lambda_1 \approx \lambda_2 \to 0$, the region is flat. Note that there is a minimum size of the neighborhood $\mathscr{N}$ that is flat for uncertainty to be high, and this neighborhood size is directly related to the receptive field.

To summarize, a high uncertainty Υ on estimated optical flow $\dot{\mathrm{p}}_\mathrm{x}$ is due to either depth boundaries or flat regions in the image larger than the receptive field of the network or when severe local illumination changes are encountered. From a slightly different perspective, depth boundaries give rise to occlusions (parts of the scene are covered by other parts of the scene that are closer) or accretions (parts of the scene are revealed as the closer part occluding it has now moved away) and in turn lead to high Υ because there is no mapping from $\mathscr{I}_t(\mathrm{x})$ to $\mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x})$. Formally,

$$\nexists \dot{\mathrm{p}}_\mathrm{x} \text{ such that } f(\mathscr{I}_t(\mathrm{x}) - \mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x})) \mid \forall \mathrm{x} \in \mathscr{N} \to 0 \tag{20}$$

In the case of a flat patch, the failure is due to a nonunique mapping from $\mathscr{I}_t(\mathrm{x})$ to $\mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x})$.

Formally,

$$\exists \{\dot{\mathrm{p}}_\mathrm{x}^i \mid i > 1\} \text{ such that } f(\mathscr{I}_t(\mathrm{x}) - \mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x}^i)) \mid \forall \mathrm{x} \in \mathscr{N} \to 0 \tag{21}$$

In other words, the map from $\mathscr{I}_t(\mathrm{x})$ to $\mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x})$ is surjective.

$$\mathscr{I}_t(\mathrm{x}) \to \mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x}) \text{ such that } f(\mathscr{I}_t(\mathrm{x}) - \mathscr{I}_{t+\delta t}(\mathrm{x}+\dot{\mathrm{p}}_\mathrm{x})) \mid \forall \mathrm{x} \in \mathscr{N} \to 0 \tag{22}$$

We experimentally observe that uncertainties are generally relatively higher for depth discontinuities as compared with other factors.

Uncertainty of monocular/stereo depth

Let us focus first on the stereo depth. This is computed using the disparity, which is defined as the displacement between the pixel between the two stereo images. This is a subset of the optical flow we discussed before. Without loss of any generality, let us assume that we have a horizontal stereo camera system where $V_\mathrm{z} = V_\mathrm{y} = 0$ because our cameras are perfectly aligned (or calibrated to be aligned) and $V_\mathrm{x} = b$ (in focal lengths), the baseline of the camera system. In addition, $\Omega = 0$ because the rotation between the cameras is zero. Here, optical flow (disparity) is being computed between two stereo images (instead of different frames from a monocular camera). The equation for disparity D (or flow) boils down to

$$D_\mathrm{x} = \frac{bf}{Z_\mathrm{x}} \tag{23}$$

Here, f is the focal length in pixels, and b and $Z_\mathbf{x}$ are in physical units. Following the discussion from the previous subsection, the same argument holds true.

$$\nexists D_\mathrm{x} \text{ such that } f(\mathscr{I}_L(\mathrm{x}) - \mathscr{I}_R(\mathrm{x}+D_\mathrm{x})) \mid \forall \mathrm{x} \in \mathscr{N} \to 0 \tag{24}$$

Here, $\mathscr{I}_L$ and $\mathscr{I}_R$ are left and right camera images. To summarize, the depth boundaries between stereo pairs give us a high Υ along with large flat regions and severe illumination changes.

For monocular depth estimation, the analysis needs a little more knowledge of how the loss functions are constructed. Because estimating depth from a single view is ill-posed without any prior knowledge, most works use a penalty on $\|\nabla \tilde{Z}_\mathrm{x}\|$, penalizing for large changes in estimated depth $\tilde{Z}_\mathrm{x}$ in a small area. This is generally true because most surfaces are smooth except at object boundaries. A similar discussion as before leads us to the conclusion that even with a monocular depth estimation network, a high Υ will be encountered at object boundaries.

Uncertainty of surface normals

Imagine a surface $\mathscr{S}$ being imaged onto an image plane as follows

$$\mathrm{x} = K[R, T]\mathrm{X} \tag{25}$$

Here, K is the camera intrinsic matrix, $[R, T]$ is the relative pose of the camera with respect to the surface (without any loss of generality, we can assume $R = \mathbb{I}_{3\times 4}$), x is the points on the image plane,

and finally, X are the real-world points on the surface $\mathscr{S}$. Now, the surface normal n_X (in the real world) at the neighborhood of a point X is given by $n_x \nabla \mathscr{S}_x$. Anytime there is a large local change in surface normals, the view that appears because of a small change in viewing direction will manifest itself as large Υ. This can happen in two main cases: (i) drastic changes to surface normal or (ii) large local illumination changes. In the first case, the capture of an image is the projection of a 3D scene into two dimensions, which causes the depth dimension $Z_{\mathbf{x}}$ to manifest x and y inversely. This means that even for small changes in x, y, the only parameter that can cause a large change in n_x (this is normal on the image plane) spatially is large changes to $Z_{\mathbf{x}}$. Intuitively, the gradient is ill-defined at the intersection of two surfaces with different normal directions.

In the second case, the reflective properties of the material and/or drastic movement of the light source can affect the value of uncertainty. In particular, when looking at a reflective surface, one can expect high uncertainties because of the specularities of point sources of light. In this scenario, using this prior knowledge of the quantity to be estimated (surface normals) can be incorporated to pick the values of f, g, and h, as discussed in (*21*).

Uncertainty of semantic segmentation

Because semantic segmentation uses the local appearance (locality size limited by the receptive field of the network) to predict the per-pixel class labels, a high Υ is expected when the appearance is not distinct enough to be clearly classified into a single class. For example, a white barrel might look like a lane line, and a pavement (footpath) might look like a road because of illumination changes. In such scenarios, Υ can be used to temporally filter semantic labels for either odometry (*60*) or improving robustness. Because this work focuses on the unconventional uses of uncertainty, we leave this as a scope for future work.

In summary, the heteroscedastic aleatoric uncertainty acts as a loss attenuator while learning; hence, the uncertainty is high whenever accretions or deletions of the scene are encountered, which generally happens at the depth and object boundaries. Hence, one can treat uncertainty as an attention mechanism when estimating a quantity that has a disparity at object edges and/or depth boundaries.

Keen readers might be thinking, "why can't we use epistemic uncertainty to do the same?" Epistemic uncertainty models what is outside the data distribution that has been learned during training. This might work if one has to detect zero-shot obstacles but will require careful crafting of the training set, which is often hard. To exacerbate the situation further, epistemic uncertainty would require N passes of the network, in contrast to aleatoric uncertainty, which only requires a single pass.

Uncertainty and its relationship to confidence and inlier ratio

The concept of loss function attenuation based on a criterion has been studied in a lot of previous works. Such attenuation has been most studied under two formulations: one as an inlier ratio and the other as a robustness parameter. The first scenario is used when one wants to learn a simplified model from the data in an unsupervised or self-supervised way. For example, if one wants to regress a six-degree-of-freedom camera pose from a pair of images that contains a lot of moving objects, one would need to only consider the background regions to obtain a robust estimate. Such a method models a subset (special case) of the regions with high uncertainty. The second scenario is used when robustness is desired such as for erroneous labels. Such a formulation generally entails an optimization problem of the following form: $\arg\min_{\tilde{y},\alpha} f(\tilde{y}, \hat{y}, \alpha)$, where α is a robustness parameter (per pixel in case of images), and f is a distance function. Such a method estimates the "importance" of a pixel in predicting $\tilde{y}$ as close as possible to $\hat{y}$. Here, this important measure is correlated inversely with the uncertainty measure.

REFERENCES AND NOTES

1. A. Grant, *Think Again: The Power of Knowing What You Don't Know* (Penguin, 2021).
2. J. Denker, Y. LeCun, Transforming neural-net output levels to probability distributions. *Adv. Neural Inf. Process. Syst.* **3**, 853–859 (1990).
3. D. J. C. MacKay, A practical Bayesian framework for backpropagation networks. *Neural Comput.* **4**, 448–472 (1992).
4. R. M. Neal. *Bayesian Learning for Neural Networks* (Springer Science & Business Media, 2012), vol. 118.
5. Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with Bernoulli approximate variational inference, in *4th International Conference on Learning Representations* (*ICLR*) *Workshop Track* (2016), https://openreview.net/pdf?id=3QxqXoJEyfp7y9wltP11.
6. Y. Gal, "Uncertainty in deep learning," thesis, University of Cambridge, Cambridge, UK (2016).
7. A. Kendall and Y. Gal. What uncertainties do we need in Bayesian deep learning for computer vision?, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17 (Curran Associates Inc., 2017), pp. 5580–5590.
8. J. Gast, S. Roth, Lightweight probabilistic deep networks, in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2018), pp. 3369–3378.
9. A. Loquercio, M. Segu, D. Scaramuzza, A general framework for uncertainty estimation in deep learning. *IEEE Robot. Autom. Lett.* **5**, 3153–3160 (2020).
10. M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, S. Nahavandi, A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Inf. Fusion* **76**, 243–297 (2021).
11. W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, J. Engel, Tlio: Tight learned inertial odometry. *IEEE Robot. Autom. Lett.* **5**, 5653–5660 (2020).
12. L. Zhou, Z. Luo, T. Shen, J. Zhang, M. Zhen, Y. Yao, T. Fang, L. Quan, Kfnet: Learning temporal camera relocalization using kalman filtering, in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2020), pp. 4919–4928.
13. R. E. Kalman, A new approach to linear filtering and prediction problems. *Trans. ASME J. Basic Eng.* **82**, 35–45 (1960).
14. S. Rabiee, J. Biswas, IVOA: Introspective vision for obstacle avoidance, in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2019), pp. 1230–1235.
15. D. Feng, L. Rosenbaum, F. Timm, K. Dietmayer, Leveraging heteroscedastic aleatoric uncertainties for robust real-time LiDAR 3D object detection, in *Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2019), pp. 1280–1287.
16. S. Lee, V. Capuano, A. Harvard, S.-J. Chung, Fast uncertainty estimation for deep learning based optical flow, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2020), pp. 10138–10144.
17. J.-G. Kang, S.-D. Roh, K.-S. Chung, FlowNetU: Accurate uncertainty estimation of optical flow for video object detection, in *Proceedings of the 2021 4th International Conference on Artificial Intelligence and Pattern Recognition* (Association for Computing Machinery, 2021), pp. 36–41.
18. E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, T. Brox, Uncertainty estimates and multi-hypotheses networks for optical flow, in *Proceedings of the European Conference on Computer Vision* (*ECCV*) (2018), pp. 652–667.
19. S. Li, X. Wu, Y. Cao, H. Zha, Generalizing to the open world: Deep visual odometry with online adaptation, in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2021), pp. 13184–13193.
20. W. Yuan, Y. Zhang, B. Wu, S. Zhu, P. Tan, M. Y. Wang, Q. Chen, Stereo matching by self-supervision of multiscopic vision, in *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021), pp. 5702–5709.

21. G. Bae, I. Budvytis, R. Cipolla, Estimating and exploiting the aleatoric uncertainty in surface normal estimation, in *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision* (*ICCV*) (IEEE, 2021), pp. 13137–13146.
22. B. Roessle, J. T. Barron, B. Mildenhall, P. P. Srinivasan, M. Nießner, Dense depth priors for neural radiance fields from sparse input views. arXiv:2112.03288 [cs.CV] (6 December 2021).
23. D. Bhatt, K. Mani, D. Bansal, K. Murthy, H. Lee, L. Paull, *f*-Cal: Calibrated aleatoric uncertainty estimation from neural networks for robot perception. arXiv:2109.13913 [cs.CV] (28 September 2021).
24. R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, D. Duckworth, Nerf in the wild: Neural radiance fields for unconstrained photo collections, in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2021), pp. 7210–7219.
25. A. Eldesokey, M. Felsberg, K. Holmquist, M. Persson, Uncertainty-aware CNNs for depth completion: Uncertainty from beginning to end, in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2020), pp. 12014–12023.
26. M. Poggi, F. Aleotti, F. Tosi, S. Mattoccia, On the uncertainty of self-supervised monocular depth estimation, in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2020), pp. 3227–3237.
27. G. Costante, M. Mancini, Uncertainty estimation for data-driven visual odometry. *IEEE Trans. Robot.* **36**, 1738–1757 (2020).
28. T. Kawashima, Q. Yu, A. Asai, D. Ikami, K. Aizawa. The aleatoric uncertainty estimation using a separate formulation with virtual residuals, in *2020 25th International Conference on Pattern Recognition* (*ICPR*) (IEEE, 2021), pp. 1438–1445.
29. T. Zhou, M. Brown, N. Snavely, D. G. Lowe. Unsupervised learning of depth and ego-motion from video, in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017), pp. 1851–1858.
30. N. J. Sanket, C. D. Singh, C. Fermüller, Y. Aloimonos, PRGFlyt: AI based indoor quadrotor autonomy framework for navigation and interaction tasks (2019); https://github.com/prgumd/PRGFlyt/wiki.
31. NVIDIA Jetson TX2; https://developer.nvidia.com/embedded/jetson-tx2.
32. N. J. Sanket, C. D. Singh, C. M. Parameshwara, C. Fermüller, G. C. H. E. de Croon, Y. Aloimonos, EVPropNet: Detecting drones by finding propellers for mid-air landing and following. arXiv:2106.15045 [cs.CV] (29 June 2021).
33. A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox. Flownet: Learning optical flow with convolutional networks, in *Proceedings of the 2015 IEEE International Conference on Computer Vision* (*ICCV*) (IEEE, 2015).
34. E. Ilg, T. Saikia, M. Keuper, T. Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation, in *European Conference on Computer Vision (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss, Eds. (Springer, 2018).
35. N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, T. Brox, A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. arXiv:1512.02134 [cs.CV] (7 December 2016).
36. C. D. Singh, N. J. Sanket, C. M. Parameshwara, C. Fermüller, Y. Aloimonos. Nudgeseg: Zero-shot object segmentation by repeated physical interaction, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2021), pp. 2714–2712.
37. X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, F. Gao, Swarm of micro flying robots in the wild. *Sci. Robot.* **7**, eabm5954 (2022).
38. G. C. H. E. de Croon, J. J. G. Dupeyroux, S. B. Fuller, J. A. R. Marshall, Insect-inspired ai for autonomous robots. *Sci. Robot.* **7**, eabl6334 (2022).
39. N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermüller, D. Scaramuzza, Y. Aloimonos. EVDodgeNet: Deep dynamic obstacle dodging with event cameras, in *Proceedings of the 2020 IEEE International Conference on Robotics and Automation* (*ICRA*) (IEEE, 2020), pp. 10651–10657.
40. R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, V. Koltun, Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 1623–1637 (2020).
41. R. Ranftl, A. Bochkovskiy, V. Koltun, Vision transformers for dense prediction. arXiv:2103.13413 [cs.CV] (24 March 2021).
42. S. Zhao, Y. Sheng, Y. Dong, E. I.-C. Chang, Y. Xu, Maskflownet: Asymmetric feature matching with learnable occlusion mask, in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2020), pp. 6278–6287.
43. D. Falanga, K. Kleber, D. Scaramuzza, Dynamic obstacle avoidance for quadrotors with event cameras. *Sci. Robot.* **5**, eaaz9712 (2020).
44. N. J. Sanket, C. D. Singh, V. Asthana, C. Fermüller, Y. Aloimonos, MorphEyes: Variable baseline stereo for quadrotor navigation, in *Proceedings of the 2021 IEEE International Conference on Robotics and Automation* (*ICRA*) (IEEE, 2021), pp. 413–419.
45. N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermüller, Y. Aloimonos, Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robot. Autom. Lett.* **3**, 2799–2806 (2018).
46. A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, R. Siegwart, Receding horizon "next-best-view" planner for 3D exploration, in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation* (*ICRA*) (IEEE, 2016), pp. 1462–146.
47. R. Bajcsy, Y. Aloimonos, J. K. Tsotsos, Revisiting active perception. *Auton. Robots* **42**, 177–196 (2018).
48. J. K. Tsotsos, On the relative complexity of active vs. passive visual search. *Int. J. Comput. Vis.* **7**, 127–141 (1992).
49. S. H. Lee, G. de Croon, Stability-based scale estimation for monocular slam. *IEEE Robot. Autom. Lett.* **3**, 780–787 (2018).
50. G. C. de Croon, C. De Wagter, T. Seidl, Enhancing optical-flow-based control by learning visual appearance cues for flying robots. *Nat. Mach. Intell.* **3**, 33–41 (2021).
51. C. Tomasi, T. Kanade, Detection and tracking of point. *Int. J. Comput. Vis.* **9**, 137–154 (1991).
52. K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in *Proceedings of the 2017 IEEE International Conference on Computer Vision* (*ICCV*) (IEEE, 2017), pp. 2961–2969.
53. A. Kirillov, Y. Wu, K. He, R. Girshick, Pointrend: Image segmentation as rendering, in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2020), pp. 9799–9808.
54. N. J. Sanket, C. D. Singh, C. Fermüller, Y. Aloimonos, Prgflow: Unified swap-aware deep global optical flow for aerial robot navigation. *Electron. Lett.* **57**, 614–617 (2021).
55. J.-L. Stevens, R. Mahony, Vision based forward sensitive reactive control for a quadrotor VTOL, in *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2018), pp. 5232–5238.
56. A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, D. Scaramuzza, Learning high-speed flight in the wild. *Sci. Robot.* **6**, eabg5810 (2021).
57. K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, G. C. H. E. de Croon, Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Sci. Robot.* **4**, eaaw9710 (2019).
58. A. Ranjan, J. Janai, A. Geiger, M. J. Black. Attacking optical flow, in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (IEEE, 2019), pp. 2404–413.
59. T. van Dijk, G. de Croon, How do neural networks see depth in single images?, in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (IEEE, 2019), pp. 2183–2191.
60. H.-J. Liang, N. J. Sanket, C. Fermüller, Y. Aloimonos, SalientDSO: Bringing attention to direct sparse odometry. *IEEE Trans. Autom. Sci. Eng.* **16**, 1619–1626 (2019).
61. Intel RealSense Depth Camera D435i. https://intelrealsense.com/depth-camera-d435i/.
62. D. Sun, X. Yang, M.-Y. Liu, J. Kautz, PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume, in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2018), pp. 8934–8943.
63. E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, FlowNet 2.0: Evolution of optical flow estimation with deep networks, in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2017), pp. 2462–2470.
64. A. Ranjan, M. J. Black, Optical flow estimation using a spatial pyramid network, in *Proceedings of the IEEE 2017 Conference on Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2017), pp. 4161–4170.

Acknowledgments: We thank N. Rajyaguru and T. Thakkar for assisting with the hardware experiments. **Funding:** The support of ONR under grant award N00014-17-1-2622 and the support of the National Science Foundation under grants BCS 1824198 and CNS 1544787 are gratefully acknowledged. **Author contributions:** N.J.S. and C.D.S. formulated the main ideas, implemented the system, performed all experiments, analyze the data, and wrote the paper. C.F. and Y.A. provided funding and contributed to the paper writing and analysis of the results. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper. The video, data, codebase, and project webpage of the work can be found at the project webpage https://prg.cs.umd.edu/ajna.

Submitted 25 June 2022
Accepted 19 July 2023
Published 16 August 2023
10.1126/scirobotics.add5139

Science Robotics

Ajna: Generalized deep uncertainty for minimal perception on parsimonious robots

Nitin J. Sanket, Chahat Deep Singh, Cornelia Fermller, and Yiannis Aloimonos

Sci. Robot., **8** (81), eadd5139.
DOI: 10.1126/scirobotics.add5139

Science Robotics (ISSN) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.