Anytime Perception and Control for Safe and Intelligent Urban Air Mobility

Heechul Yun*, Ahmet Soyyigit†, Qitao Weng‡, Shawn Keshmiri§
University of Kansas, Lawrence, KS, USA

Pavithra Prabhakar¶
Kansas State University, Manhattan, KS, USA

Nelson Brown‖
NASA Armstrong Flight Research Center, Edwards, CA, USA

Urban Air Mobility (UAM) applications, such as air taxis, will rely heavily on perception for situational awareness and safe operation. With recent advances in AI/ML, state-of-the-art perception systems can provide the high-fidelity information necessary for UAM systems.

However, due to size, weight, power, and cost (SWaP-C) constraints, the available computing resources of the on-board computing platform in such UAM systems are limited. Therefore, real-time processing of sophisticated perception algorithms, along with guidance, navigation, and control (GNC) functions in a UAM system, is challenging and requires the careful allocation of computing resources.

Furthermore, the optimal allocation of computing resources may change over time depending on the speed of the vehicle, environmental complexities, and other factors. For instance, a fast-moving air vehicle at low altitude would need a low-latency perception system, as a long delay in perception can negatively affect safety. Conversely, a slowly landing air vehicle in a complex urban environment would prefer a highly accurate perception system, even if it takes a little longer. However, most perception and control systems are not designed to support such dynamic reconfigurations necessary to maximize performance and safety.

We advocate for developing "anytime" perception and control capabilities that can dynamically reconfigure the capabilities of perception and GNC algorithms at runtime to enable safe and intelligent UAM applications. The anytime approach will efficiently allocate the limited computing resources in ways that maximize mission success and ensure safety. The anytime capability is also valuable in the context of distributed sensing, enabling the efficient sharing of perception information across multiple sensor modalities between the nodes.

I. Introduction

In the 2021 report of NASA Aeronautics, Urban Air Mobility (UAM) is identified as one of the fastest-growing sectors of the U.S. Aerospace industry [1]. Despite its tremendous potential, the integration of UAM into the U.S. National Air Space (NAS) faces significant challenges because UAM vehicles may fly at low altitudes and in close proximity to buildings, populations, and other manned/unmanned aerial systems, possibly under turbulent wind and other hazardous weather conditions.

Advanced real-time perception and control are, therefore, crucial for UAM because UAM vehicles must perceive the environment and safely navigate through it from take-off to landing, either completely autonomously or partially autonomously for an extended period. Lidar, camera, and radar sensors are increasingly utilized for advanced perception systems as they provide rich high-dimensional information. Furthermore, recent advances in computer vision and artificial intelligence, especially deep neural networks (DNNs) [2, 3], have enabled dramatic improvements in perception capabilities needed for UAM.

*Associate Professor, EECS, heechul.yun@ku.edu
†PhD Candidate, EECS, ahmet.soyyigit@ku.edu
‡MS Student, EECS, wengqt@ku.edu
§Professor, Aerospace Engineering, keshmiri@ku.edu
¶Professor, Computer Science, pprabhakar@ksu.edu
‖Aerospace Engineer/Project Chief Engineer, nelson.brown@nasa.gov

Executing these advanced perception algorithms is, however, computationally expensive, requiring powerful computing resources that may be difficult to be used in safety-critical systems due to strict size, weight, power, and cost (SWaP-C) constraints. Furthermore, there is a trade-off relationship between performance (accuracy) and execution time (latency) in many perception algorithms, which further complicates the algorithm selection. For instance, a larger DNN may be able to achieve higher accuracy than smaller one, but it can take much longer, which may be unacceptable for the safety of the UAM.

Moreover, depending on the aircraft's flight phase or flight condition, different trade-offs may exist in the allocation of computing resources. Specifically, the latency and accuracy requirements of a perception task in a UAM system can dynamically change over time. For instance, a fast-moving UAM vehicle at cruising altitude may prioritize a less accurate but faster object detection system over a highly accurate but slower one. On the other hand, a slow-moving UAM vehicle in a complex urban environment may prioritize higher accuracy, even if it takes longer to detect objects. Additionally, hazardous situations, such as the sudden appearance of obstacles (e.g., birds) or changes in weather conditions (e.g., high wind gusts), may necessitate triggering new path planning and complex control maneuvers. This may require the reallocation of computing resources to process more urgent tasks in a timely manner.

Therefore, there is a strong need to make a time and accuracy trade-off for a perception system on the fly so that the system can adapt to a dynamically changing environment and make the best use of the on-board computing resources of the vehicle. Unfortunately, most existing perception algorithms are fixed at design time and cannot be dynamically adjusted at runtime once deployed. While a fixed perception algorithm may be easier to develop and validate, it may significantly underutilize computing resources and limit the performance potential of the UAM system.

In this work, we advocate for developing *anytime* perception and control algorithms capable of making time and accuracy trade-offs on the fly. This enables the system to adapt to a dynamically changing environment and provides maximum performance and efficiency for UAM systems.

Note that the concept of anytime computing is not new [4], and there has been a growing research effort to support anytime perception capabilities in the research community in recent years. For example, researchers proposed applying anytime computing in a vision-based image classification task by dynamically skipping some layers [5–8] or prioritizing a subset of neurons [9] in the backbone networks, while others applied similar ideas to vision-based real-time object detection problems [5]. Our prior work advanced the state-of-the-art by prioritizing important sub-areas of the input in vision-based object detection [10] and significant sub-classes in lidar-based 3D object detection [11]. However, these prior works do not consider holistic allocations of computing resources between the perception and control tasks, a focus we aim to tackle in collaboration with NASA for UAM application scenarios.

We note that anytime perception capability—the ability to dynamically reconfigure the characteristics of perception using on-board computing resources at runtime—is also important for *distributed sensing* [12]. Sharing raw sensory data over today's wireless networks would require high bandwidth and incur high latency, both of which are highly undesirable for UAM. On the other hand, sharing the outcomes of a vehicle's perception system would require much less bandwidth and latency on the underlying communication networks, thus improving the efficiency and robustness of distributed sensing systems. More importantly, participating nodes can negotiate among themselves the right level of accuracy and latency trade-offs depending on the available computing resources and mission scenarios, enabling extra flexibility and opportunities for additional performance optimizations.

II. Anytime Perception and Control in UAM: The Vision

Anytime algorithms refer to a class of algorithms that can trade deliberation time for the quality of results [4]. They are designed to provide useful results even when interrupted or allowed only limited time to run. These algorithms are particularly useful in real-time or interactive systems where available computational resources or time constraints may vary. UAMs are such systems that can benefit from anytime algorithms. Figure 1 shows the general characteristics of anytime algorithms.

Anytime perception is especially relevant for UAM due to the time-sensitive nature and the computational complexity of perception in UAM applications. However, most state-of-the-art perception systems, particularly deep neural network (DNN)-based ones, are not capable of making quality (accuracy) and deliberation time (latency) trade-offs on the fly. Although bigger DNNs generally achieve higher accuracy than smaller ones, they are more computationally expensive and thus take longer. Once the DNN is trained and deployed on the target computing system, its computing cost is generally fixed and cannot be altered at runtime, regardless of the available computing resources. Similar dynamics are at play in many non-DNN-based perception algorithms, which generally lack adaptability at runtime.

In this work, we envision anytime perception and control capabilities for UAM applications.

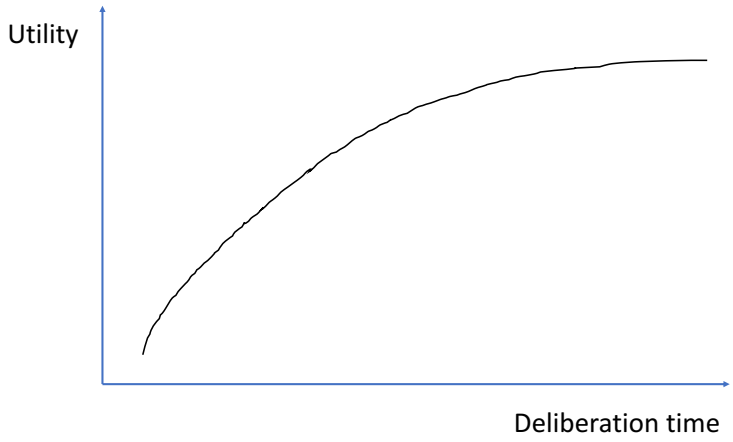


Fig. 1 A conceptual diagram of anytime algorithms.

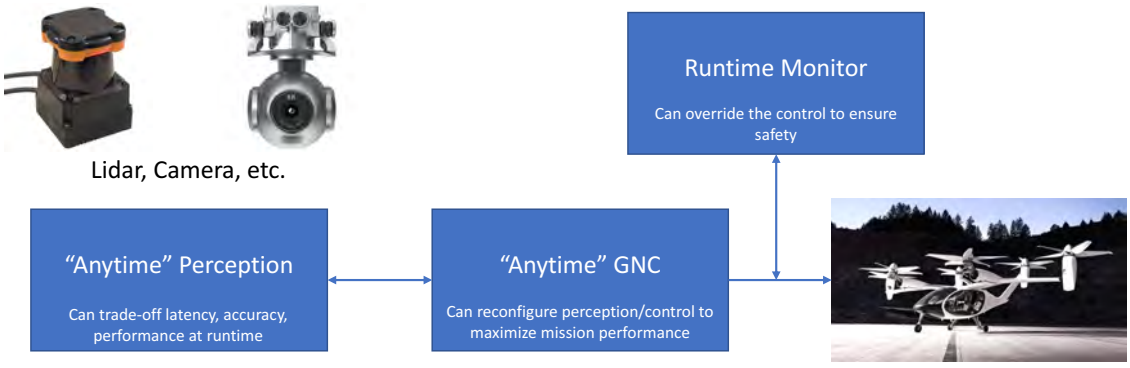


Fig. 2 Overview of *anytime* perception and control system

Figure 2 shows an overview of a UAM system with the proposed anytime perception, GNC, and system-wide performance and safety assurance capabilities. Anytime perception will target state-of-the-art perception models/algorithms based on cameras and lidars, augmenting them to be anytime-capable. Anytime GNC will integrate the anytime perception capability into the design of guidance, navigation, and control algorithms. In doing so, computing resources will be allocated to maximize the performance and safety of the UAM system depending on the flight conditions and the surrounding environment. The control actions of the anytime GNC system will be monitored by the runtime monitor, which employs formal modeling and verification techniques to ensure the safety and performance of the entire system.

III. Anytime Perception: Preliminary Work

In this section, we present three examples of anytime perception algorithms that we have explored for UAM applications.

A. Anytime Lidar Object Detection

Lidar-based 3D object detection is an important perception task for UAM systems as it can precisely and robustly detect objects in a complex surrounding environment using lidar point clouds. The state-of-the-art detection performance is achieved with DNN-based algorithms. Unfortunately, these DNN-based lidar object detectors are computationally expensive for inference operations, require costly retraining for each different model architecture, and cannot be changed at runtime once the model is trained and deployed.

To overcome these limitations, we developed Anytime-Lidar, a deadline-aware 3D object detection system that enables latency and accuracy trade-off adjustments at runtime [11]. It is based on a state-of-the-art 3D object detection network called PointPillars [13] and extends its capability to support a wide range of deadline requirements.

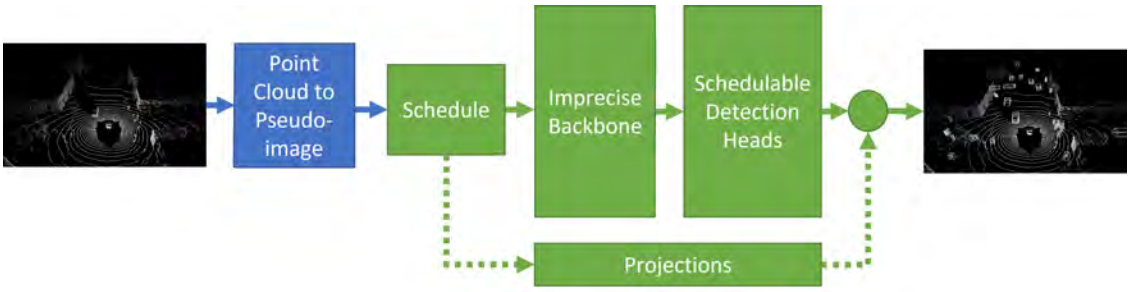


Fig. 3 Anytime-lidar 3D object detection pipeline for deadline-aware anytime execution [11] .

Figure 3 shows the overall architecture of Anytime-Lidar. In a nutshell, our enhancements can be described in four parts. First, we allow its backbone (feature extractor) to run a subset of blocks (stages) for lower latency at the cost of some accuracy loss, which we call an imprecise backbone. Second, we enable the skipping of a subset of less important detection heads in favor of more important ones to reduce the execution cost. Third, we provide approximate results for the skipped heads by projecting the past detection results to the current frame, which is less computationally expensive than executing the detection heads, albeit with some loss of accuracy. Lastly, the stage/head scheduler manages the other three modules and decides the number of blocks of the backbone and which detection heads are to be executed so that we can guarantee the whole execution will meet the user-provided deadline.

Figure 4 shows the achieved accuracy over all tested deadlines. PointPillars-3/2/1 represent three versions of the original PointPillar network, which cannot adapt to changing deadline requirements. As such, when the requested deadline is shorter than their nominal execution times, they fail to complete the detection. On the other hand, MultiStage [5] supports some degree of deadline flexibility, but Anytime-Lidar (Ours) offers a much wider range of deadlines and strictly higher accuracy. It is important to note that in Anytime-Lidar, as the deadline becomes shorter, detection accuracy gradually decreases, allowing the control system to make accuracy and deadline trade-offs dynamically at runtime.

We are currently developing Anytime-Lidar2, which further improves performance by exploiting the sparsity of the input point cloud data. Specifically, we are developing effective ways to focus on a subset of input point cloud data, significantly reducing the computation cost with minimal loss of accuracy.

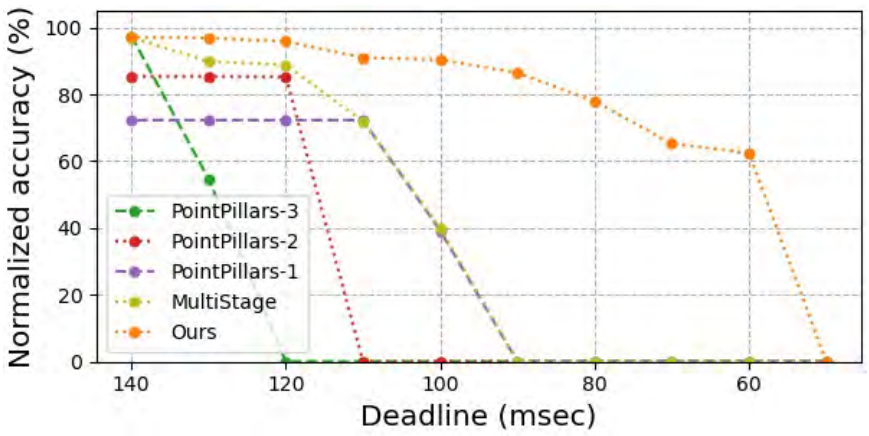


Fig. 4 Detection accuracy of Anytime-Lidar versus baselines [11].

B. Anytime Visual SLAM/Odometry

Visual Simultaneous Localization and Mapping (SLAM) is a crucial perception capability for Urban Air Mobility (UAM) because it enables real-time localization of the vehicle and the creation of a map of the surrounding environment. Visual SLAM is especially useful in GPS-denied or degraded urban areas.

In this work, we focus on OV^{2}SLAM, which is a visual SLAM algorithm designed specifically for real-time applications [14]. OV^{2}SLAM comprises four main components: the front-end, mapping, state optimization, and loop closer, as described below: The *Front-End* thread performs real-time pose estimation of the camera sensor and is responsible for creating keyframes used in creating a 3D map of the surrounding environment; The *Mapping* thread uses keyframes generated in the *Front-End* to generate new 3D map points. It primarily does this by performing triangulation on the keyframes. Then, if a new keyframe has not arrived, it will also perform local map tracking in order to minimize drift; The *State Optimization* thread performs two main operations. First, it runs a local bundle adjustment (BA) to refine camera pose estimations. Second, it runs a keyframe filtering pass that prevents redundant keyframes from being processed in future BA operations; The *Loop Closer* thread performs an online bag-of-words (BoW) operation to detect loop closures in a system's given trajectory. Note that each of these modules is implemented as a separate thread for maximum performance and efficiency. Among the threads, only the front-end thread must run for every input frame that is fed to OV^{2}SLAM. The remaining threads will run only when necessary, such as when a new keyframe is created.

1. Effect of Task Co-scheduling on SLAM Accuracy

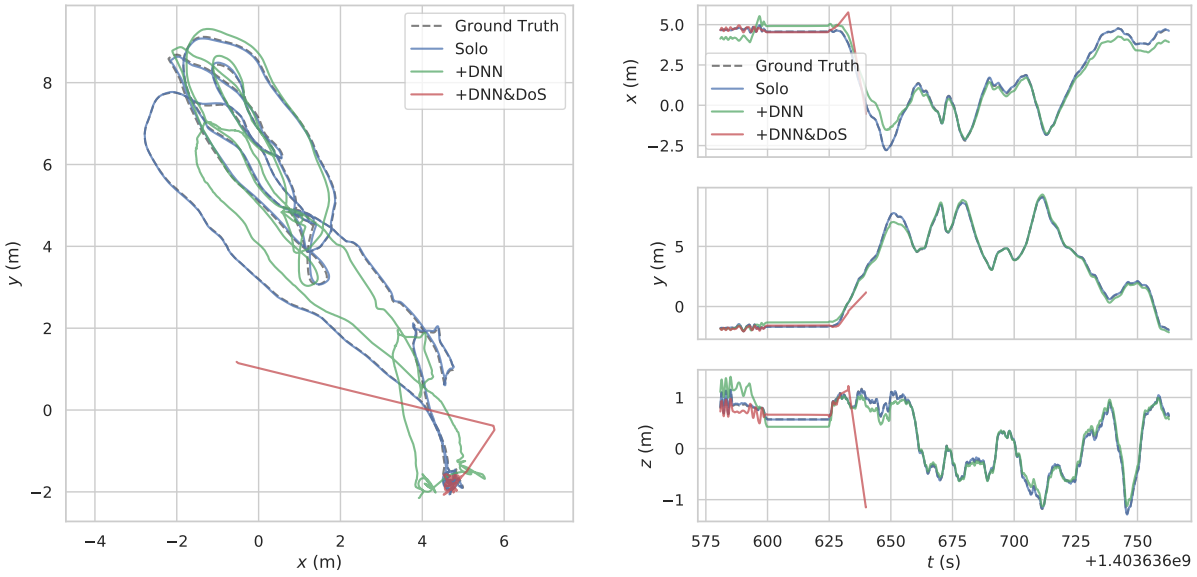
When we evaluate the algorithm using the NASA Atla8 sUAS dataset [15] and EuRoC dataset [16] alone in *isolation* on an embedded computer with sufficient computing resources, we find that it produces highly accurate trajectory estimates (See Figure 9 in Section IV.A).

In our recent study [17], however, we found that when some of the threads of OV^{2}SLAM are delayed because of other co-scheduled tasks on the same computing platform, the performance (accuracy) of the SLAM can significantly deteriorate, as shown below.

Figure 5 shows the performance of the OV^{2}SLAM algorithm when it is co-scheduled with two other tasks, a DNN and a DoS attack, on the same on-board computing platform (NVIDIA Jetson Nano). The DNN task is a deep neural network workload processing camera input in real-time while the DoS task is a memory intensive CPU workload.

Note that when the DNN task is co-scheduled (without the DoS), the trajectory errors of the SLAM increase significantly compared to when the SLAM algorithm runs alone (Solo). This is mainly because the state optimization thread runs more slowly due to contention, impacting the accuracy of the SLAM algorithm. When both DNN and DoS tasks are co-scheduled, the performance of SLAM deteriorates even further to a level that completely fails to generate any useful trajectory information. This is because OV^{2}SLAM is unable to keep up with the input data and ends up dropping a majority of the frames. Our analysis shows that only approximately 26% of the input image frames were processed in *+DNN&DoS*.

In other words, when these multiple tasks execute on an on-board computing platform without regard to available computing resources for them, their performance cannot be guaranteed, even though each task's performance was



(a) Trajectory in XY plane

(b) X, Y and Z positions over time

Fig. 5 SLAM accuracy impact of co-scheduling other compute/memory intensive workloads [17]

validated on the same platform in isolation.

The results show why perception algorithms must be designed to dynamically reconfigure themselves in response to changes in available computing resources or environmental factors to avoid such failures and successfully complete missions.

As UAM systems need to consolidate many sophisticated functions—whose computational demands may vary significantly depending on environmental and operational conditions—into a single shared onboard computing platform for SWaP-C constraints, such an adaptive capability will be increasingly important in UAM.

2. Feasibility of Anytime Visual SLAM

To test the feasibility of anytime SLAM, we investigate algorithmic parameters that affect the accuracy and latency of the OV^{2}SLAM algorithm. We find that the real-time performance of OV^{2}SLAM critically depends on the efficient detection and processing of keypoints—distinct features in images used for tracking movement and reconstructing environments. OV^{2}SLAM distinguishes itself with a strategic approach to keypoint generation, featuring a configurable grid strategy for keypoint detection. This flexibility in keypoint detection plays a pivotal role in balancing the trade-off between accuracy and execution time.

In general, a smaller grid cell size leads to a higher density of keypoints, enabling more detailed and accurate mapping. However, processing a larger number of keypoints demands more computational resources, potentially slowing down the system, especially on limited hardware. Conversely, a larger grid cell size reduces keypoint density, easing computational load and improving response time, but may compromise mapping accuracy in complex environments.

We assess the impact of varying keypoint grid cell sizes in OV^{2}SLAM, using the EuRoC dataset, while adjusting the clock frequency of the on-board computer to simulate different computing resource availability. We observe the effects on the Absolute Pose Error (APE) of the trajectory, highlighting the delicate balance between computational constraints and mapping precision.

Figure 6 illustrates the Absolute Pose Error (APE) across different keypoint grid cell pixel sizes: 50kps and 75kps. At the maximum CPU frequency (i.e., maximum computing resources available), the 50kps configuration shows a lower APE score than the 75kps configuration, indicating better accuracy with more computational resources. However, when we decrease the CPU frequency (reduced computing resources available), the 50kps configuration's performance degrades significantly, while the 75kps configuration remains relatively stable across CPU frequencies. This demonstrates the feasibility of dynamically balancing keypoint density depending on available computational

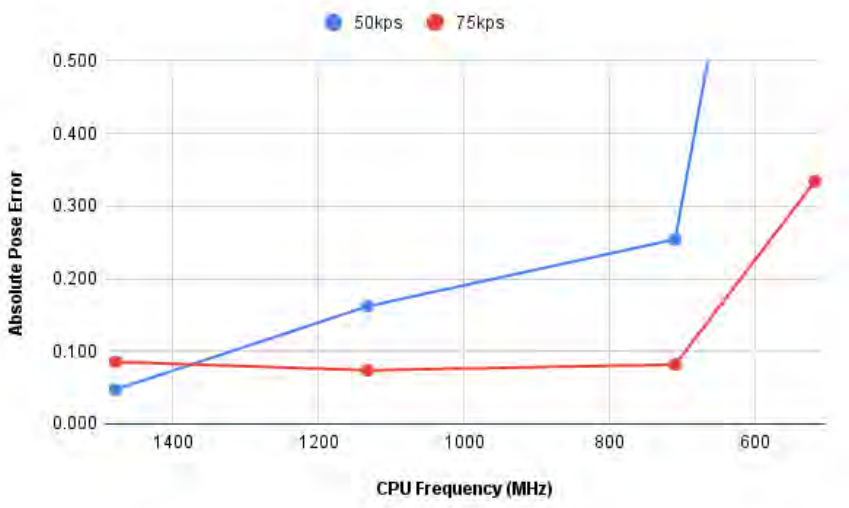


Fig. 6 Absolute pose error of OV^{2}SLAM at varying computing budget and keypoint cell size (kps) settings

resources for optimal SLAM performance and that the traditional one-size-fits-all approach is sub-optimal.

We plan to systematically explore other algorithmic parameters that can affect latency and accuracy trade-offs of the OV^{2}SLAM algorithm and develop an anytime-capable variant that can automatically and dynamically reconfigure these algorithmic parameters, such as adjusting the keypoint grid cell size, to maximize the effectiveness of the algorithm in response to changes in available computing resources and other environmental/operating conditions.

C. Anytime Lidar Odometry

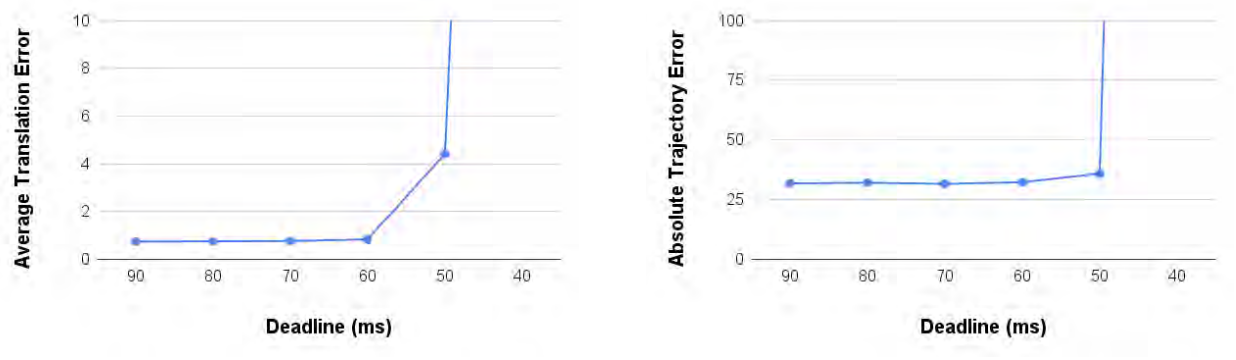
The odometry module of an autonomous system is tasked with measuring how its position and orientation change online. For this purpose, lidar scans can be processed to obtain accurate results. As a case study, we focus on the popular KISS-ICP (Keep It Simple and Stupid - Iterative Closest Point) [18] odometry algorithm and gauge its anytime computing capability using the KITTI odometry dataset [19]. We chose this algorithm as it can be easily deployed on various platforms without the need for any additional sensor besides a lidar.

KISS-ICP aims to progressively calculate the trajectory of a mobile vehicle by consecutively aligning the captured point clouds obtained from the lidar. This alignment involves an iterative refinement, where more iterations yield better alignments and thus better motion predictions. As a result, KISS-ICP can be considered as an anytime algorithm.

By default, the iterative refinement of KISS-ICP continues until a predetermined error threshold is reached. This causes the execution time to be input-dependent. We changed this condition to a dynamically determined deadline instead. When the time spent for execution is about to reach the deadline, we stop the iterative refinement process and output the odometry result. This way, we allow KISS-ICP to be deadline-aware. We call this revised version Anytime KISS-ICP (A-KISS-ICP).

We evaluate A-KISS-ICP's performance under a range of deadline constraints by utilizing the KITTI Odometry Dataset. For this purpose, we consider the Average Translational Error and Average Trajectory Error metrics. As our experiment platform, we use Jetson Orin Nano, a commercial-off-the-shelf embedded computing platform, and allocate two CPU cores for the A-KISS-ICP.

Figure 7 illustrates the results we have obtained by measuring the odometry performance under a range of deadline constraints. Each case of deadline was conducted as a separate experiment with the same dataset. In terms of translation and trajectory error, we observe a slight gradual degradation until the deadline becomes 50 milliseconds. This illustrates the fact that achieving a feasible solution is possible by not doing the iterative refinement until a predetermined error threshold is reached. At 50 milliseconds case, we observe a more noticeable decrease in performance. These results tell us that the initial iterations of the refinement play the most important role while later iterations have a lesser impact on the result.



(a) Average Translation Error over deadlines

(b) Average Trajectory Error over deadlines

Fig. 7 Anytime KISS-ICP odometry performance for KITTI dataset.

IV. Ongoing Work: UAM Case Study

In this section, we present our on-going effort to bring anytime perception and control capabilities into real UAM application scenarios.

A. Datasets

Our on-going collaboration with NASA Armstrong includes evaluation of anytime perception algorithms using the NASA sUAS dataset [15] and its successors. The NASA sUAS dataset was produced by the Revolutionary Aviation Mobility Project * [15], which includes vision, IMU, and GNSS data of several eVTOL operation scenarios, including landing and take-off on building rooftops. The NASA sUAS dataset and its successors are specifically tailored for perception in VTOL applications and thus ideally suited to evaluate anytime perception and control algorithms for UAM.

Fig. 8 Example scene from the NASA sUAS dataset

Figure 8 shows an example image of the NASA sUAS dataset. The NASA sUAS dataset is composed of images and associated position and orientation data, which were collected at NASA Armstrong Flight Research Center using a

*https://nari.arc.nasa.gov/ttt-ram/

Freefly Alta 8 multirotor sUAS. The aircraft flew approaches to a helipad landing zone with markings for the touchdown and liftoff area (TLOF), final approach and take off (FATO), and safety area (SA) as described in the FAA Advisory Circular AC 150/5390-2D - Heliport Design. The landing trajectory mimicked the notional trajectory of a larger eVTOL aircraft such as the NASA RVLT reference designs, starting at about 1500 ft (460 m) at an altitude of about 250 ft (76 m) above ground. The original imagery is in 4K resolution, which we reduced to 752x480 pixels similar to the other datasets used in this work.

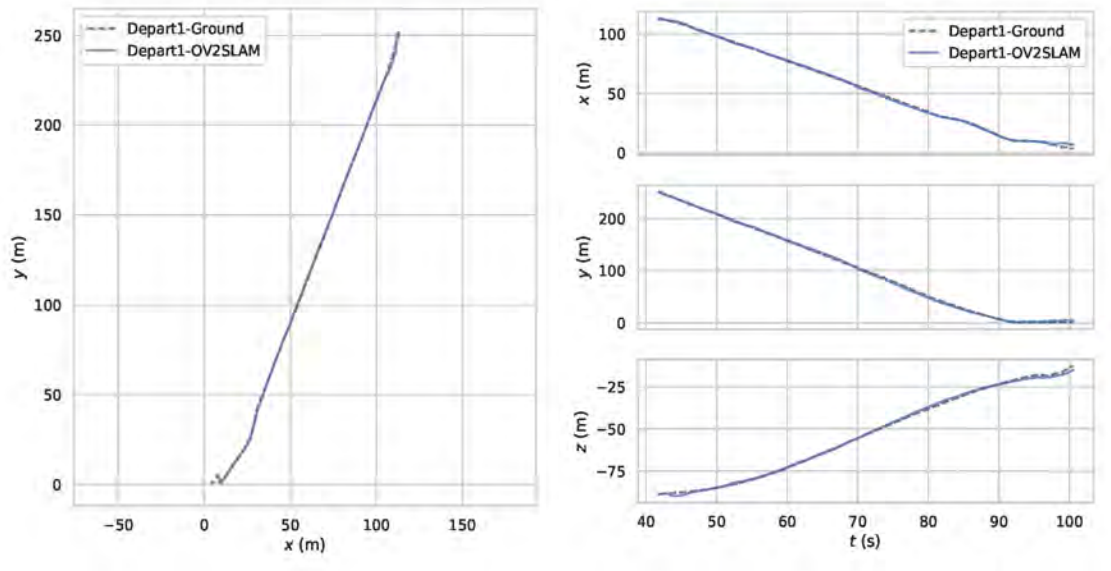


Fig. 9 OV^{2}SLAM produced trajectory of part of the NASA sUAS dataset.

Figure 9 shows the OV^{2}SLAM produced trajectory of a single flight video from the NASA sUAS dataset. The visual SLAM produced trajectory information shows a good match compared with the ground truth trajectory.

B. UAM Testbeds

In addition to dataset-driven anytime perception evaluation, we plan to leverage a hardware-in-the-loop simulation setup in which our flight computer, running our anytime perception and GNC, is connected to a simulator[20, 21] that models the flight dynamics and multi-modal sensory input (camera, lidar, and IMU/GPS) for perception. This setup will allow us to evaluate the performance of the proposed system under a range of realistic scenarios and environments in a controlled manner. Figure 10a shows a simulator-generated environment along with three types of vision sensors on a simulated drone. Our plan is to demonstrate improved mission performance by adopting anytime perception compared to the system without anytime perception. For instance, we plan to compare the time to reach the destination without collision or other safety failures in a challenging urban take-off/landing scenario to showcase the effectiveness of anytime perception, first using the AirSim simulator, and later using the real Aurelia X4 drone.

In the future, we plan to deploy our algorithms on an actual UAS and conduct flight tests to evaluate the effectiveness of the proposed anytime perception algorithms. Considering many similarities between UAM platforms with takeoff and landing (VTOL) aircraft, we will use the Aurelia X4 as a testbed, shown in Figure 10b. The Aurelia X4 is designed to support large payloads, such as LIDAR, and is capable of carrying up to 1.5 kg. The Aurelia X4 has a 20-minute flight time, and the X-6 version of this platform has been flight tested numerous times by our team.

V. Conclusion

UAM requires powerful perception and control for high intelligence and safety. However, advanced perception algorithms are often computationally expensive and involve many design-level trade-offs in terms of performance (accuracy) and latency (execution time).

In this work, we made a case for developing *anytime* perception and control algorithms for UAM applications.

(a) AirSim [20] simulator

(b) Aurelia X4 VTOL testbed

Fig. 10 Evaluation platforms

In particular, anytime perception algorithms can make time and accuracy trade-offs on the fly, allowing the UAM system to easily adapt to the dynamically changing environment and achieve maximum performance and efficiency. We presented three example anytime perception algorithms—anytime lidar object detection, visual SLAM/odometry, and Lidar odometry algorithms—and discuss their feasibility and capabilities. We then outlined our plan to integrate and evaluate these anytime perception capabilities in both simulated and real UAM testbeds.

In the future, we intend to report the effectiveness of the proposed anytime approach in maximizing mission performance through carefully designed case study scenarios.

Acknowledgments

This research is supported in part by NSF CNS-1815959, CPS-2038923, FAA A54_A11L.UAS.97, and NASA KNEP EPSCoR PDG.

References

[1] Cohen, A. P., Shaheen, S. A., and Farrar, E. M., "Urban air mobility: History, ecosystem, market potential, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 22, No. 9, 2021, pp. 6074–6087.

[2] Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Kitsukawa, Y., Monrroy, A., Ando, T., Fujii, Y., and Azumi, T., "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," *ICCPS*, 2018, pp. 287–296. https://doi.org/10.1109/ICCPS.2018.00035.

[3] Li, Y., and Ibanez-Guzman, J., "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems," *IEEE Signal Processing Magazine*, Vol. 37, No. 4, 2020, pp. 50–61. https://doi.org/10.1109/MSP.2020.2973615.

[4] Zilberstein, S., "Using anytime algorithms in intelligent systems," *AI magazine*, Vol. 17, No. 3, 1996, pp. 73–73.

[5] Heo, S., Cho, S., Kim, Y., and Kim, H., "Real-Time Object Detection System with Multi-Path Neural Networks," *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020, pp. 174–187. https://doi.org/10.1109/RTAS48715.2020.000-8.

[6] Kim, J.-E., Bradford, R., and Shao, Z., "AnytimeNet: Controlling Time-Quality Tradeoffs in Deep Neural Network Architectures," *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 945–950. https://doi.org/10.23919/DATE48585.2020.9116280.

[7] Bateni, S., and Liu, C., "ApNet: Approximation-Aware Real-Time Neural Network," *2018 IEEE Real-Time Systems Symposium (RTSS)*, 2018, pp. 67–79. https://doi.org/10.1109/RTSS.2018.00017.

[8] Yao, S., Hao, Y., Zhao, Y., Shao, H., Liu, D., Liu, S., Wang, T., Li, J., and Abdelzaher, T., "Scheduling Real-time Deep Learning Services as Imprecise Computations," *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2020, pp. 1–10. https://doi.org/10.1109/RTCSA50079.2020.9203676.

[9] Lee, S., and Nirjon, S., "SubFlow: A Dynamic Induced-Subgraph Strategy Toward Real-Time DNN Inference and Training," *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020, pp. 15–29. https://doi.org/10.1109/RTAS48715.2020.00-20.

[10] Liu, S., Yao, S., Fu, X., Shao, H., Tabish, R., Yu, S., Bansal, A., Yun, H., Sha, L., and Abdelzaher, T., "Real-Time Task Scheduling for Machine Perception in In Intelligent Cyber-Physical Systems," *IEEE Transactions on Computers*, 2021, pp. 1–1. https://doi.org/10.1109/TC.2021.3106496.

[11] Soyyigit, A., Yao, S., and Yun, H., "Anytime-Lidar: Deadline-aware 3D Object Detection," *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2022.

[12] Ippolito, C. A., Hashemi, K. E., Kawamura, E., Gorospe, G. E., Holforty, W., Kannan, K., Stepanyan, V., Lombaerts, T., Brown, N., Jaffe, A. M., et al., "Distributed Sensing and Advanced Perception Technologies to Enable Advanced Air Mobility," *AIAA SCITECH 2023 Forum*, 2023, p. 0894.

[13] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O., "PointPillars: Fast Encoders for Object Detection From Point Clouds," *CVPR*, 2019, pp. 12689–12697. https://doi.org/10.1109/CVPR.2019.01298.

[14] Ferrera, M., Eudes, A., Moras, J., Sanfourche, M., and Le Besnerais, G., "OV^{2}SLAM: A Fully Online and Versatile Visual SLAM for Real-Time Applications," *IEEE Robotics and Automation Letters*, Vol. 6, No. 2, 2021, pp. 1399–1406. https://doi.org/10.1109/LRA.2021.3058069.

[15] Brown, N., Kawamura, E., Bard, L., Jaffe, A., Ringelberg, W., Kannan, K., and Ippolito, C., "Visual Inertial Datasets for an eVTOL Aircraft Approach and Landing Scenario," *AIAA SciTech 2024 Forum*, 2024.

[16] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R., "The EuRoC Micro Aerial Vehicle Datasets," *The International Journal of Robotics Research*, 2016.

[17] Bechtel, M., and Yun, H., "Analysis and Mitigation of Shared Resource Contention on Heterogeneous Multicore: An Industrial Case Study," *arXiv preprint arXiv:2304.13110*, 2023.

[18] Vizzo, I., Guadagnino, T., Mersch, B., Wiesmann, L., Behley, J., and Stachniss, C., "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way," *IEEE Robotics and Automation Letters (RA-L)*, Vol. 8, No. 2, 2023, pp. 1029–1036. https://doi.org/10.1109/LRA.2023.3236571.

[19] Geiger, A., Lenz, P., and Urtasun, R., "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[20] Shah, S., Dey, D., Lovett, C., and Kapoor, A., "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and service robotics*, Springer, 2018, pp. 621–635.

[21] Amini, A., Wang, T.-H., Gilitschenski, I., Schwarting, W., Liu, Z., Han, S., Karaman, S., and Rus, D., "Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles," *ICRA*, IEEE, 2022, pp. 2419–2426.