Energy-Aware Design of Vision-Based Autonomous Tracking and Landing of a UAV

Zamanakos, Georgios; Seewald, Adam; Midtiby, Henrik Skov; Schultz, Ulrik Pagh

Published in:
2020 Fourth IEEE International Conference on Robotic Computing (IRC)

DOI:
10.1109/irc.2020.00054

Publication date:
2020

Document version:
Accepted manuscript

Energy-Aware Design of Vision-Based Autonomous Tracking and Landing of a UAV

Georgios Zamanakos, Adam Seewald, Henrik Skov Midtiby, and Ulrik Pagh Schultz
SDU UAS Center, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark
Contact email: ups@mmmi.sdu.dk

***Abstract*—In this paper, we present the design and evaluation of a vision-based algorithm for autonomous tracking and landing on a moving platform in varying environmental conditions. We use an energy-aware approach, where the design of the algorithm is based on an evaluation of the energy consumption and Quality of Service (QoS) of each computational component. We evaluate our approach with an agricultural use case where a moving platform is tracked using a landing marker and the YOLOv3-tiny CNN is used to detect ground-based hazards. We perform all computations onboard using an NVIDIA Jetson Nano and analyse the impact on the flight time by profiling the energy consumption of the marker detection and the CNN. Experiments are conducted in Gazebo simulation using an energy modeling tool to measure the computational energy cost as a function of QoS. We test the energy efficiency and robustness of our system in various dynamic wind disturbances. We show that the marker detection algorithm can be run at the highest QoS with only a marginal energy overhead whereas adapting the QoS level of CNN results in a considerable power saving. The power saving is significant for a system executing on a fixed-wing UAV.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly used for applications such as monitoring, surveillance, and agriculture. Extending the flying time of a UAV can be done by landing to replace or charge the battery before continuing the mission. GPS is however insufficient for robust precision landing. We investigate the use of a vision-based autonomous landing system and evaluate its robustness towards environmental conditions such as visual disturbances and wind.

Extension of flight time can be achieved by using *energy-aware design of algorithms* to reduce energy consumption by reducing the Quality of Service (QoS) where it has little impact on overall system performance. Energy-costly computations are adapted by selecting the desired QoS to match the available energy [1]. We aim to combine energy-aware algorithm design with autonomous landing capabilities to extend the flight time and increase the total availability of the UAV.

The main contribution of this paper concerns the experimental study of a vision-based algorithm for autonomous tracking and landing in varying environmental conditions. The algorithms are executed on an NVIDIA Jetson Nano companion computer controlling a simulated drone. The algorithms provide novel capabilities in terms of tolerance to visual disturbance and varying environmental conditions such as wind. Our experiments are based on an agricultural use-case where a UAV performs visual identification of ground-based hazards while tracking and landing on a moving platform.

II. State of the Art

Vision-based autonomous landing has used landing markers in an "H"-shape pattern; landing markers inspired by QR code [2]; ArUco markers [3], AprilTag markers [4]; and special-pattern black and white markers [5] and color markers [6]. An onboard implementation of the computer vision algorithms to detect a moving platform has been demonstrated using CPU [7] and GPU [8]. Wind conditions are generally kept stable, i.e., a constant wind speed of $5\,\mathrm{m/s}$ has been used as an external disturbance in a simulation environment [9]. Concerning position estimation for landing, similar to our approach a Kalman Filter or Extended Kalman Filter (EKF) has been used for the estimation [10], as well as a velocity observer algorithm based on calculating the actual moving distance of the moving platform over a period of time [8].

Energy modeling of computer vision algorithms was investigated in SLAMBench [11], a framework that automatically evaluates algorithm configuration alternatives for energy efficiency. Mission-based energy models [12] have focused mostly on ground-based autonomous vehicles rather than UAVs. The relation between motion and energy in a robot has been investigated [13], but without accounting for the energy required for computation. Energy modeling of mobile robots [14] has provided the basis for our concept of energy-aware algorithm design. Indeed, such modeling has evolved from an energy-efficient motion planning technique [15], a design strategy that allows accounting for motion and computations separately [16], to an energy-efficient robot deployment algorithm [14].

In this work we propose a new marker pattern robust to occlusions and suitable for a realistic outdoor case. Since the GPU cannot normally run different algorithms simultaneously, we utilize the CPU for detecting the landing marker and the GPU for a CNN to detect ground hazards. By doing so, a different QoS can be chosen for each algorithm. We use `powprofiler`, a generic energy modeling tool [17], to measure the energy impact of different configurations of the ROS-based system implementing the agricultural use-case. The `powprofiler` tool is part of the TeamPlay toolchain, which aims to make tradeoffs between energy and other non-functional properties accessible to the developer. In this paper, we present extensions to `powprofiler` that facilitates the initial exploration of the energy usage of ROS-based systems.

III. Energy-Aware System Design

The energy-aware design approach is a mission-oriented concept that adjusts the computations to the mission being performed while taking into account energy requirements. Here, we profile and adapt the computationally heavy algorithms performing autonomous tracking, landing, and hazard detection. This adaptation is energy-aware in the sense that QoS parameters are adapted to enable the mission to be completed at the highest possible QoS level that does not significantly impact the available energy budget. Tradeoffs between QoS parameters are thus be made as part of the system design, i.e., trading the robustness towards wind during landing for precision of hazard detection.

Energy-aware design using `powprofiler` relies on empirical experiments to measure the actual power consumption on the robot hardware [17]. In this paper, we focus on profiling the energy usage of the companion computer, which from the point of view of energy consumption can be studied independently from the specific drone it is mounted in. The developer must specify the maximum and minimum QoS level for each algorithm running on the system. During mission execution the levels are statically defined: automatic, dynamic adaptation is considered future work.

The developer executes the system to empirically determine the power consumption in one of two ways:

1) Automatically using `powprofiler` to control the experiment execution [17]. For a ROS-based system, we assume that the algorithms are wrapped as ROS nodes, and we require the developer to declare the QoS parameters. The QoS parameters are interpreted by `powprofiler`, iterating through all possible combinations and empirically sampling the energy consumption of each combination of QoS parameters. Once all combinations have been iterated through, `powprofiler` automatically combines the energy consumption data into a complete model.
2) Semi-automatically using `powprofiler` to sample energy and combine the results of all experiments, but allowing the developer to control all aspects of the experiment execution. This new approach uses a dedicated ROS node interfaced to `powprofiler`, enabling the developer to start/stop sampling in a given configuration by publishing on a topic. Once all experiments have been completed, `powprofiler` is invoked to combine the energy consumption data into a complete model.

Regardless of the approach, `powprofiler` builds a model mapping QoS to total system energy consumption. Coarse-grained sampling is employed to reduce the number of experiments, and missing values are automatically inferred from the others by the means of a multivariate linear interpolation.

In the context of this paper, sampling experiments are iterated in a simulated environment with different configurations. For example, the autonomous tracking allows changing the tracking algorithm QoS in terms of frequency, the landing algorithm in terms of frequency, and hazard detection QoS in terms of frequency.

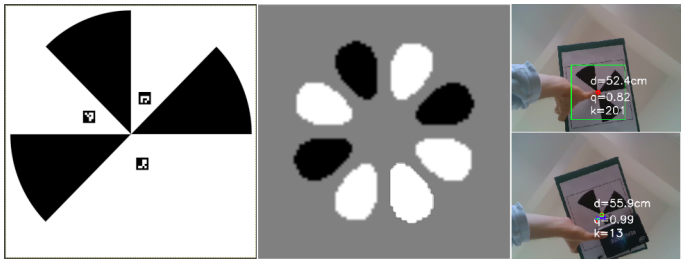

Fig. 1. Left: landing marker. Middle: n-fold marker detection kernel (white regions on kernel and marker correspond to real values and black regions to imaginary values, grey area on the kernel is of no concern). Top-right: detection under occlusion on the tip of the n-fold marker. Bottom-right: detected under occlusion of a sector of the n-fold marker.

IV. Vision-Based Tracking and Landing

To mark the moving platform a special pattern is used, consisting of an n-fold marker [18] and three ArUco markers [19] with different ids. This pattern is referred to as the *landing marker* and can be seen in Fig. 1. The n-fold marker is primarily used to detect the moving platform from a high altitude, while the ArUco markers are used as extra landmarks in case the marker is partially visible in the image frame.

To extract the pixel coordinates of the tip (center) of the n-fold marker, a kernel size of 13×13 pixels consisting of a real and imaginary part is created. We present a larger kernel size for visualisation purposes in Fig. 1. For every pixel in the image, a convolution is performed with this kernel and the magnitude of the convolution is stored. For the pixel with the highest magnitude, an overall normalized quality score q between 0.0 and 1.0 is calculated. If the score is above the threshold then the pixel is accepted as the tip of the n-fold marker. An adaptive kernel selection function is implemented to ensure the selection of a proper minimum kernel size k based on a threshold quality score value q, balancing between the computationally expensive convolution process and the effective marker detection. The marker detection results under two different occlusion cases can be seen in Fig. 1. To detect the ArUco markers the standard OpenCV library is used.

To convert the pixel coordinates into a relative position the PX4 flight controller's onboard sensor measurements are utilized. The altitude of the UAV is obtained from the flight controller, whereas the downward facing camera's horizontal and vertical field of view along with the IMU and barometric sensor measurements are used to project the image plane down to the ground plane. A perspective homography matrix is calculated between the two planes to transform the pixel coordinates into a relative position. For ArUco markers an offset vector is added depending on the distance from the tip of the n-fold marker.

The tracking algorithm uses an EKF to provide an accurate prediction for the position of the moving platform at any given time. This prediction allow images to be processed at different frames per second (fps) according to a desired QoS. Furthermore, the overall robustness of the system is increased in case the moving platform is not detected in every image

Fig. 2. On the left, a top view of the Gazebo scene. On the right, a view of the UAV attempting a landing on the moving platform.

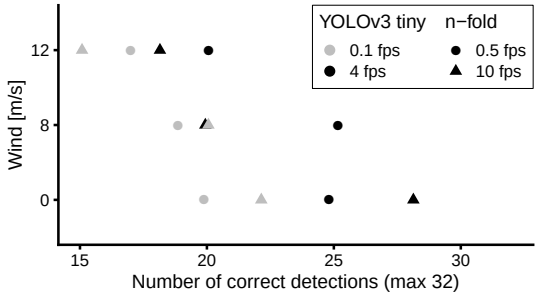

Fig. 3. Number of correctly detected objects under different conditions. Grey color denotes a 0.1fps and black color denotes a 4fps update rate in *YO*. Circles denote a 0.5fps and triangles a 10fps update rate in the *LM*.

frame. A velocity estimator for the moving platform is also implemented as a part of the tracking algorithm.

V. Evaluation

A. Use-case: agricultural safety

We evaluate our approach based on a simulated use-case where a multirotor UAV identifies hazardous objects around a moving platform, and lands on the moving platform. A simulated field is created in Gazebo with randomly placed objects as seen in Fig. 2. Object detection and classification of cars, humans, tractors and cows is performed by feeding the input image from the downward facing camera into a *YOLOv3-tiny* CNN [20] implemented in ROS [21]. A dataset based on Gazebo models was created consisting of 1200 images, 300 for each class. We trained the CNN with batch size 64 for 130 epochs using default parameters and for an input image size of 416×416 pixels.

We assume a windy environment where wind forces are applied on the center of gravity of the UAV with direction parallel to the ground. The magnitude of the applied force is calculated from the wind speed [22], [23]: The applied force on the UAV is found to be $3.45\,\mathrm{N}$ for an $8\,\mathrm{m/s}$ wind speed, and $7.76\,\mathrm{N}$ for a $12\,\mathrm{m/s}$ wind speed. The direction of the wind force remains constant while its magnitude changes every $5.5\,\mathrm{s}$ between -10% and +20% of the initial wind force. For a UAV altitude of $6.0\,\mathrm{m}$ and below, the wind force decreases linearly. This model is used for all simulated tests.

B. Experimental setup

All experiments are performed in Gazebo using Ubuntu 18.04 and ROS Melodic on a standard laptop. The *PX4* Software In The Loop (SITL) Firmware v1.10.2 is used as the flight controller and the *IRIS* quadcopter is used as the UAV platform. A downward facing RGB camera is placed on the UAV providing a 640×480 pixel image at 10 fps. An NVIDIA Jetson Nano with Ubuntu 18.04 and ROS Melodic is used as the UAV's companion computer. Energy profiling is performed directly on the NVIDIA Jetson Nano using `powprofiler` as outlined in Section III.

In *tracking* mode the UAV follows the moving platform at a fixed altitude and maps the ground environment. In *landing* mode the UAV follows the moving platform and gradually lowers its altitude until it lands on it. Both *tracking* and *landing* modes are evaluated for energy consumption and QoS under cases of no wind, $8\,\mathrm{m/s}$ and $12\,\mathrm{m/s}$ wind disturbances.

C. Results

For *energy evaluation of tracking*, 8 tests were executed for different rates of the *YOLOv3-tiny* ROS node *YO* (4fps, 1fps, 0.5fps, 0.1fps) and the *landing marker* detection node *LM* (10fps, 0.5fps). For a *YO* 4fps update rate a power consumption of $6.30\,\mathrm{W}$ is observed while for 1fps and 0.1fps update rates the power consumption drops to $4.8\,\mathrm{W}$ and $3.9\,\mathrm{W}$ accordingly. By reducing the *LM* update rate from 10fps to 0.5fps, a further power saving of $0.15\,\mathrm{W}$–$0.19\,\mathrm{W}$ is achieved.

For *QoS evaluation of tracking*, 12 tests were executed for different fps rates for *YO* (4fps, 0.1fps) and for *LM* (10fps, 0.5fps) under 3 different wind speeds. The QoS is determined as the number of correctly detected objects: objects detected within a distance of $2\,\mathrm{m}$ from their actual position and classified correctly. The detection results can be seen in Fig. 3. The best results were obtained for high *YO* and *LM* fps update rates and no wind, where 28 out of the 32 objects were detected. (High fps rates imply that each object is viewed more times as the UAV moves.) For the same high fps values but with a wind speed of $12\,\mathrm{m/s}$ only 18 out of 32 objects were detected. This difference is due to the UAV being at an angle to balance itself against the wind, which causes some objects to be outside the camera's field of view or to be viewed at an angle which the CNN has not been trained for.

For the *energy evaluation of landing*, 4 tests were executed for different *LM* fps rates (10fps, 2fps, 1fps, 0.5fps). The largest observed difference in power consumption is $0.14\,\mathrm{W}$ and is observed between the 0.5fps and the 10fps *LM* update rate. Landing with 10fps update rate is however $30\,\mathrm{s}$ faster.

For the *QoS evaluation of landing*, 12 tests were executed for different *LM* fps rates (10fps, 2fps, 1fps, 0.5fps) under 3 different wind speeds. The QoS is determined as the mean squared error (MSE) between the predicted and actual position of the moving platform. Four different altitude bins were used, see Fig. 4. A large MSE of around $3\,\mathrm{m}^2$ is observed for an altitude greater than $20\,\mathrm{m}$ for a 0.5fps rate, while an MSE close to zero is observed for an altitude of less than $5\,\mathrm{m}$ for an update rate of 10fps. Wind disturbances do not significantly influence the MSE. We believe the larger MSE for the y coordinates compared to the x coordinates is caused by sudden changes of the moving platform's direction on the y axis.

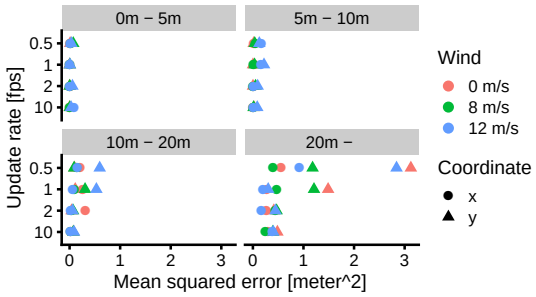

Fig. 4. Position error during landing and how it depends on the marker detection rate at different altitudes and wind disturbances. Circles denote errors in the x direction and triangles errors in the y direction.

D. Discussion

The experiments show that both *tracking* and *landing* are operational in a simulated environment with a moving platform and random wind conditions. The performance of both modes is QoS-sensitive with a high success rate at high QoS levels, and significantly lower performance at lower levels.

The potential energy savings from adapting the QoS by changing the *YO* and *LM* fps values should be seen in relation to the total energy consumption of the UAV. Consider a DJI Phantom 4 multirotor and a Sky-Watch Cumulus fixed-wing (the fixed-wing would need to circle while tracking and would need VTOL capabilities to land). Using respective product pages regarding battery capacity and flight time we estimate that the Phantom uses roughly $140\,\mathrm{W}$ while cruising whereas the Cumulus uses roughly $40\,\mathrm{W}$ while cruising. The saving gained from changing the *YO* rate is $6.30\,\mathrm{W} - 3.9\,\mathrm{W} = 2.4\,\mathrm{W}$ whereas the saving gained from changing the *LM* rate is $0.2\,\mathrm{W}$. For the Cumulus, there is thus a 6.5% potential energy saving, whereas the potential energy saving only is 1.9% for the Phantom. For the Cumulus this is considered large enough to significantly impact the flying time of the drone. For the Cumulus, the potential saving from adapting the *LM* QoS is however only 0.5%. For the tracking mode, changing the *LM* rate provided a minor saving of only $0.14\,\mathrm{W}$, but at the cost of increased landing time. Therefore, although the higher-QoS landing algorithm is marginally more expensive by $0.14\,\mathrm{W}$, the UAV will overall save energy due to a reduced flight time.

VI. Conclusion and Future Work

We have presented the energy-aware design of a vision-based algorithm for autonomous tracking and landing in varying environmental conditions, based on experiments with an NVIDIA Jetson Nano companion computer. Our experiments show that the proposed computer vision algorithms for detecting the moving platform can be run at the highest QoS level with only a marginal energy overhead, whereas adapting the QoS level of the CNN results in a considerable power saving for the system as a whole, representing a significant power saving for a fixed-wing UAV. In terms of future work, we are interested in automatically adapting the QoS level and in testing this approach on a physical drone.

Acknowledgment

This work is supported and partly funded by the European Union's Horizon2020 research and innovation program under grant agreement No. 779882 (TeamPlay).

References

[1] A. Seewald, H. Garcia de Marina, H. S. Midtiby, and U. P. Schultz, "Mechanical and computational energy estimation of a fixed-wing drone," in *2020 4th IEEE IRC*, 2020, p. to appear.

[2] H. Yuan, C. Xiao, S. Xiu, W. Zhan, Z. Ye, F. Zhang, C. Zhou, Y. Wen, and Q. Li, "A hierarchical vision-based localization of rotor unmanned aerial vehicles for autonomous landing," *Int. J. Distributed Sensor Networks*, vol. 14, no. 9, 2018.

[3] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *2012 IEEE ICRA*, 2012, pp. 971–976.

[4] O. Araar, N. Aouf, and I. Vitanov, "Vision based autonomous landing of multirotor UAV on moving platform," *J. Intelligent & Robotic Systems*, vol. 85, no. 2, pp. 369–384, 2017.

[5] P. H. Nguyen, M. Arsalan, J. H. Koo, R. A. Naqvi, N. Q. Truong, and K. R. Park, "LightDenseYOLO: A fast and accurate marker tracker for autonomous UAV landing by visible light camera sensor on drone," *Sensors*, vol. 18, no. 6, p. 1703, 2018.

[6] H. Lee, S. Jung, and D. H. Shim, "Vision-based UAV landing on the moving vehicle," in *2016 Int. Conf. on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 1–7.

[7] X. Chen, S. K. Phang, M. Shan, and B. M. Chen, "System integration of a vision-guided UAV for autonomous landing on moving platform," in *2016 12th IEEE ICCA*, 2016, pp. 761–766.

[8] T. Yang, Q. Ren, F. Zhang, B. Xie, H. Ren, J. Li, and Y. Zhang, "Hybrid camera array-based UAV auto-landing on moving UGV in GPS-denied environment," *Remote Sensing*, vol. 10, no. 11, p. 1829, 2018.

[9] Y. Feng, C. Zhang, S. Baek, S. Rawashdeh, and A. Mohammadi, "Autonomous landing of a UAV on a moving platform using model predictive control," *Drones*, vol. 2, no. 4, p. 34, 2018.

[10] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, "Vision-based autonomous quadrotor landing on a moving platform," in *2017 IEEE SSRR*, 2017, pp. 200–207.

[11] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. Kelly, A. J. Davison, M. Luján, M. F. O'Boyle, G. Riley *et al.*, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for slam," in *2015 IEEE ICRA*, 2015, pp. 5783–5790.

[12] A. Sadrpour, J. Jin, and A. G. Ulsoy, "Experimental validation of mission energy prediction model for unmanned ground vehicles," in *2013 American Control Conf.* IEEE, 2013, pp. 5960–5965.

[13] J. Morales, J. L. Martinez, A. Mandow, A. J. García-Cerezo, and S. Pedraza, "Power consumption modeling of skid-steer tracked mobile robots on rigid terrain," *IEEE Trans. Robotics*, vol. 25, no. 5, pp. 1098–1108, 2009.

[14] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Trans. Robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[15] ——, "Energy-efficient motion planning for mobile robots," in *2004 IEEE ICRA*, vol. 5, 2004, pp. 4344–4349.

[16] ——, "A case study of mobile robot's energy consumption and conservation techniques," in *2005 IEEE ICRA*, 2005, pp. 492–497.

[17] A. Seewald, U. P. Schultz, E. Ebeid, and H. S. Midtiby, "Coarse-grained computation-oriented energy modeling for heterogeneous parallel embedded systems," *Int. J. Parallel Programming*, pp. 1–22, 2019.

[18] Henrik Skov Midtiby, "N-fold marker tracker repository," https://github.com/henrikmidtiby/MarkerLocator, 2015.

[19] OpenCV, "Detection of ArUco markers," https://docs.opencv.org/3.4/d5/dae/tutorial_aruco_detection.html, 2020.

[20] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv*, 2018.

[21] M. Bjelonic, "YOLO ROS: Real-time object detection for ROS," https://github.com/leggedrobotics/darknet_ros, 2016–2018.

[22] NASA, "Dynamic pressure (NASA)," https://www.grc.nasa.gov/WWW/K-12/airplane/dynpress.html, 2020.

[23] J. D. Anderson Jr, *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 2010.