Energy-Sensitive Vision-Based Autonomous Tracking and Landing of a Multirotor on a Moving Platform

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

***Abstract*—abstract abstract abstract abstract abstract abstract abstract abstract abstract abstract abstract abstract abstract abstract**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly used for applications such as monitoring, surveillance, transportation of small payloads, and agricultural applications [1], [2]. One of the major constraints of such applications is their limited level of autonomy due to battery limitations. Extending the flying time of a UAV is normally done by having it land in order to replace or charge the battery before continuing the mission. Performing landings autonomously can however be challenging depending on the environment and whether the landing platform is stationary or mobile. Moreover, relying solely on the availability of a GPS signal for autonomous precision landing is not considered safe, since GPS signals can be temporarily lost or even tampered with. As an alternative, in this paper we investigate the use of a novel vision-based autonomous landing system, and evaluate its robustness towards environmental conditions such as visual disturbances and wind.

Extension of the flight time can be also achieved by using *energy-sensitive algorithms* that can reduce energy consumption by reducing the quality of service (QoS). With this approach, energy-costly computations such as computer vision are adapted by selecting the desired quality of service (QoS) to match the available energy [3]. By combining energy-sensitive algorithms with autonomous landing capabilities, we aim to increase the total availability of the UAV to perform operations, by extending the flight time and using autonomous recharging when needed.

The main contribution of this paper concerns the experimental study of a robust, energy-sensitive, 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 vision-based tracking and landing 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 multirotor UAV performs visual identification of ground-based hazards while tracking and landing on a moving tractor.

II. State of the Art

Vision-based autonomous landing on a marker has been extensively studied by many researchers. Key distinctions include whether the marker is on a moving platform, the type of the marker, the algorithms used to detect it, as well as the mounted sensors on-board of the UAV.

For stationary platforms, one of the first experiments with vision-based autonomous landing was conducted by Saripalli et al. [4]. Here, a helicopter with a color camera facing vertically towards the ground would land on an H-shape pattern (similar to ones found on a helipad) using a hierarchical behavior-based control architecture. In physical tests a marker of 122cm$\times$122cm size was detected for a maximum altitude of 10m. A landing marker inspired by a QR code but consisting of three artificial markers is demonstrated by Yuan et al. [5], and was shown to provide a 6-DOF pose over an altitude range of 0-20m. Our work is however focused on the ability to land on moving platforms.

Saripalli et al. [6] also demonstrated the use of a Kalman Filter to track a moving platform. However all the computations were performed offline. Similarly, an ArUco marker was used as a landing marker by Lee et al. [7] to detect a moving platform. The control of the UAV is performed based on the error provided by the vision algorithm but all the computations were performed off-board. Arrar, et al. [8] focus on extending the detection range by using an AprilTag [9] as a landing marker. Again all the computer vision algorithms were also executed off-board. A crucial aspect of our application is to perform all the computations on-board, and to evaluate them according to their energy efficiency as a function of QoS.

HEMI: I think the last sen-

The design of the marker and choice of sensors can facilitate doing the computations on-board. Chen et al. [10] utilized a marker consisting of a circle and rectangles of different colors along with a LiDAR scanning range finder for height estimation. The marker was detected by performing color segmentation on the incoming image frame. By fusing the height measurement from the LiDAR into the vision measurement, a relative pose of the UAV from the moving platform was obtained. A color segmentation approach was also implemented by Lee et al. [11]. A red rectangle was used as a landing marker and a vertically facing camera with a fish-eye lens was used to detect it, and a successful landing from an altitude of 70m was demonstrated. Both teams have used an on-board companion computer to perform all the computation on the UAV. However a color segmentation approach is not considered as a safe option for a realistic (outdoor) case it would be difficult, if not impossible, to ensure that the landing marker will be the only object of a specific color in the scene.

The use of a hybrid camera system consisting of a fish-eye IR camera and a stereo camera was demonstrated by Yang et al. [12]. An ArUco marker was used to mark the moving platform and a convolutional neural network (CNN) Yolo v3 was trained specifically for marker detection. A similar approach concerning the detection of a landing marker was demonstrated by Nguyen et al. [13]. Here a specific landing marker was used and a specific CNN was used to detect it: successful detection of a 1m×1m marker size was demonstrated from a distance of 50m. An AprilTag marker was used as a landing marker by Kyritsis et al. [14] for the purpose of “2016 DJI Developer Challenge”. The identification of the AprilTag marker was performed through Graphics Processing Unit (GPU). The three teams have utilized the companion’s computer GPU to detect the landing marker. In the agricultural use case addressed in this paper, the GPU is however needed for a CNN to detect ground hazards, and since the GPU cannot simultaneously run different algorithms, the CPU should be used for detecting the landing marker. By doing so, a different QoS could be chosen for each algorithm.

HEMI: I thought that the cores on a GPU could be assigned different jobs that could run simultaneous

To account for the energy modeling of computer vision algorithms, we considered the work previously carried by Nardi et al. [15]. The authors present SLAMBench, a framework that investigates SLAM algorithms configuration alternatives for energy efficiency. Whereas, we use `powprofiler`, a generic energy modeling tool [16]. This tool enables measuring the energy impact of different configurations of the ROS-based system implementing the agricultural use-case. In this paper we present extensions to `powprofiler` that facilitates the initial exploration of the energy usage of complex, ROS-based robotic systems.

Other approaches to energy modeling, such as the mission-based energy models studied by Sadrpour et al. [17], [18], focus mostly on ground-based autonomous vehicles instead of the UAVs. Morales et al. [19] extensively investigated the relation between motion and energy in a robot, but do not account for on the energy required for computation. Energy modeling of mobile robots as carried by Mei et al. [20]–[22] has provided the ground for the concept of modeling computation for energy-sensitive algorithm design. The authors’ approach has evolved from an energy-efficient motion planning technique in [22], a design strategy that allows accounting for motion and computations separately in [21], to an energy-efficient deployment algorithm in [20].

The battery in our system is considered in the context of a drone being able to perform its mission while accounting for the eventuality of a battery shortage; to this end, we investigated the approach presented by Berenz et al. [23], where a battery management mission-based dynamic recharge approach is presented. A set of recharge stations are used, along with self-docking capable robots. Our approach similarly allows landing on a moving platform for recharging, which is in the context of this paper is considered in the proximity of the drone. The actual landing is handled by the proposed algorithm, and we also account for the energy required for executing this algorithm during landing.

Taking into account varying environmental conditions and unpredictable movements of the platform to land on is relevant for the use of landing in outdoor, mobile scenarios. Regarding wind conditions, an AprilTag marker was used by Feng et al. [24] with a constant wind speed of 5 m/s as an external disturbance in a simulation environment. Nevertheless, a fluctuation in the wind’s magnitude and direction is likely to happen in realistic cases. Concerning estimation of the moving platform’s position and velocity, a Kalman Filter or Extended Kalman Filter (EKF) has been used for the estimation [8], [24], [25], whereas Yang et al. [12] constructed a velocity observer algorithm by calculating the actual moving distance of the moving platform over a period of time.

III. Energy-Sensitive Mission Deployment

A. Overall approach

The energy-sensitive design is a mission-oriented concept that adjusts the computations to the mission being performed while taking into account energy requirements, including energy consumed by actuation, computation, and the presence of a limited power source. Specifically, in the agricultural use-case, the concept is employed to adapt the computationally heavy algorithms performing autonomous tracking, landing, and hazard detection. This adaptation enables energy-sensitivity, in the sense that QoS parameters can be modified to enable the mission to be completed at the highest possible QoS level that does not exceed the available energy budget. Tradeoffs between QoS parameters can be performed by an end-user, i.e., trading the robustness towards wind during landing for precision of hazard detection.

The energy-sensitive design using `powprofiler` relies on empirical experiments to measure the actual power consumption on the robot hardware [16]. In this paper we focus on the companion computer, which from the point of view of energy consumption can be studied independently from the specific drone it is mounted in.

First, the developer specifies the maximum and minimum QoS level for each algorithm running on the system. During mission execution the levels are statically defined: automatic adaptation during different phases of the mission is being currently being investigated and is considered future work. Then, the developer executes the system to empirically determine the power consumption. This can be done in two different ways:

1) Automatically using `powprofiler` to control the experiment execution [3]. We assume that the algorithms are wrapped as ROS nodes, and we require the developer to specify the QoS parameters using a ROS configuration.We use a configuration file in a key-value pair format which is then interpreted by `powprofiler`, enabling `powprofiler` to iterate through all possible combinations and empirically sample 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 approach is new and is described in more detail later in this section. Basically, a ROS node interfaces to `powprofiler` and is used by the developer to start/stop sampling in a given configuration. Once all experiments have completed, `powprofiler` is invoked by the developer to combine the energy consumption data into a complete model.

Regardless of the approach, `powprofiler` builds a single model mapping QoS to total system energy consumption. Coarse-grained sample 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.

B. Semi-automatic energy profiling

ADAM: TODO? Short description of ROS node and combining level 1 models using pow-

IV. Vision-Based Autonomous Tracking and Landing

The vision-based autonomous tracking and landing can be split into four main sub-problems: detection of the moving platform, navigation, guidance, and control of the UAV. From an energy-sensitive design approach, our focus is on the detection of the moving platform and the computer vision algorithms used to detect it, and the parameterization by a QoS influencing energy consumption and performance, as described in Section IV-A. Furthermore, the navigation block is designed to increase the robustness and overall performance of the system, as described in Section IV-B. Last, a model of dynamically changing wind disturbances is analysed and described in Section **??**, allowing the system to be tested on a realistic use case.

A. Detection of the moving platform

To mark the moving platform a special pattern is constructed, consisting of an n-fold marker [26] along with three ArUco markers [27] with different ids. This pattern will be referred to as the “landing marker” and can be seen in Figure 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 evaluate the detection algorithm of the landing marker, real images, of 640×480 pixel size, were captured with an Intel RealSense D435 camera. In gazebo simulation a color camera with the same distortion coefficients as the Intel camera, is used to output a 640×480 pixel image at 10 frames per second (fps).

HEMI: Add details about the simulated camera.

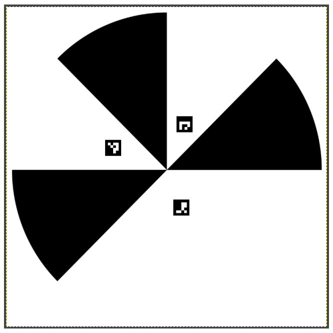

Fig. 1. The landing marker

To extract the pixel coordinates of the tip of the n-fold marker, a kernel size of 13×13 pixels consisting of a real and imaginary part is created. For every pixel in the image, a convolution is performed with this kernel and the magnitude of the convolution is stored. The pixel with the highest magnitude is considered as a candidate tip of the n-fold marker. For that candidate pixel only, an estimation of the orientation/phase of the marker is made and an overall normalized quality score between 0.0 and 1.0 is calculated. If the score is above a desired threshold value then the pixel is accepted as the tip of the n-fold marker. If an n-fold marker is detected, the result will be the pixel coordinates of the tip of the n-fold marker along with its orientation/phase.

Since a convolution is a computational expensive process, an increase in the kernel size would also increase the computation time and therefore the energy consumption. However a higher energy consumption is preferred over a non-detection of the n-fold marker. To balance between energy consumption and effective marker detection, an adaptive kernel selection function is created to ensure the selection of a proper kernel size based on a threshold quality score value. In figure 2 an Intel RealSense D435 Depth camera is used to capture the image and measure also the distance d from the nfold marker. Based on a desired quality q, the proper kernel size k is selected. It is seen that an occlusion on the tip of the nfold marker results in a significant increase on the selected kernel size.

HEMI: To me this looks like a significant *decrease* in kernel size . . .

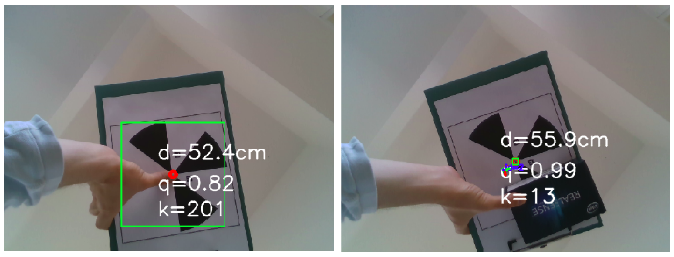

Fig. 2. Detection of the landing marker under different occlusions

To detect the ArUco markers the standard OpenCV library is used and if any ArUco markers are detected, their central pixel coordinates and pose are stored.

The next step is to convert these pixel coordinates into a real world relative position $[X, Y, Z]$ according to a local coordinate frame. The origin of the local coordinate frame $[0, 0, 0]$ is defined as the center of the landing marker and alignment according to the North, East, Down (*NED*) frame. The available sensor measurements and sensor fusion algorithms from the flight controller will be used in this process. The PX4 flight controller outputs through mavlink messages the altitude of the UAV and the attitude of the UAV (roll,pitch,yaw). For the Z component, the altitude of the UAV from the flight controller's EKF is used. For the X, Y components, an algorithm is constructed to convert pixel coordinates into real world X, Y coordinates, in meters:

1) The pose of the camera in UAV's *BODY* frame is calculated by utilizing the roll and pitch IMU data.
2) The normalized coordinates of the four image corners, according to the camera's horizontal, vertical field of view and the camera's *BODY* pose from step 1, are calculated.
3) The world coordinates of the four image corners are determined by using the normalised coordinates from step 2 along with the UAV's altitude. The result is a projection plane of the image corners on the ground.
4) The perspective homography matrix is calculated between the 2 planes, the image plane and the world plane from step 3.
5) The pixels coordinates are converted into world coordinates by using the homography matrix from step 4.
6) The world coordinates from step 5, are converted from the camera's *BODY* frame into the camera's *NED* frame by using the yaw IMU data from the flight controller.
7) The world coordinates from step 6, are converted from camera's *NED* frame into the landing site's local coordinate frame.
8) For ArUco markers only, an offset vector in the x,y axis is added depending on the ArUco marker's id. It is assumed that this vector is prior known.

The mean measurements from the detected n-fold and/or ArUco markers are used to determine the position and orientation of the landing marker. The result is an $[X, Y, Z]$ relative position of the UAV from the moving platform along with the orientation of the landing marker.

B. Navigation

The purpose of the navigation block is to provide an accurate prediction for the state of the UAV at any given time. This is important because it will allow us to process images at different 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 frame. A velocity estimator for the moving platform will also be implemented as a part of the navigation block.

The variables of interest that describe the state of the UAV for this project are:

- The relative position of the UAV from the moving platform $[X, Y, Z]$ calculated as described in Section IV-A.
- The attitude of the UAV [roll, pitch, yaw], obtained from the flight controller's IMUs.
- the velocity of the UAV $[vx, vy, vz]$ in the *NED* frame, obtained from the flight controller's EKF.
- the acceleration of the UAV $[ax, ay, az]$ in the *NED* frame, obtained either from the flight controller's EKF or by differentiating the velocities.

The altitude, attitude, velocity and acceleration variables of the UAV's state are obtained from the flight controller's onboard sensors and already implemented sensor fusion algorithms (EKF). To fuse those state variables with the obtained position measurements from Section IV-A, a Kalman Filter will be used. The measurements from the flight controller will be used in the prediction step.

Prediction Step:

$$\begin{aligned} \hat{x}_t &= F_t * \hat{x}_{t-1} + G_t * u_t \\ P_t &= F_t * P_{t-1} * F_t^\top + Q_t \end{aligned} \tag{1}$$

where:

$$\hat{x}_t = \begin{bmatrix} x_{NED} \\ y_{NED} \end{bmatrix}, \quad F_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$G_t = \begin{bmatrix} dt & 0 & \frac{dt^2}{2} & 0 & -dt & 0 \\ 0 & dt & 0 & \frac{dt^2}{2} & 0 & -dt \end{bmatrix}, \quad u_t = \begin{bmatrix} vx_{NED} \\ vy_{NED} \\ ax_{NED} \\ ay_{NED} \\ vx_{m_{NED}} \\ vy_{m_{NED}} \end{bmatrix}$$

$$P_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q_t = \begin{bmatrix} dt & 0 \\ 0 & dt \end{bmatrix} * \sigma_{IMU}^2$$

where:

- x_{NED}, y_{NED} is the position of the UAV in the landing site's local coordinate system (aligned with NED frame),
- dt is the time interval the PX4's EKF outputs the vx_{NED}, vy_{NED} data, usually around 33ms,
- vx_{NED}, vy_{NED} is the linear velocity of the UAV in NED frame, estimated from the PX4's EKF,

- ax_{NED}, ay_{NED} is the linear acceleration of the UAV in NED frame, estimated on the companion computer from differentiating the velocities.
- $vx_{m_{NED}}, vy_{m_{NED}}$ is the linear velocity of the moving platform in NED frame, estimated from the velocity estimator for the moving platform,
- σ^2_{IMU} is the variance of the velocities based estimated by the PX4's EKF.

In the correction step the observed measurements from the vertically positioned camera will be used to correct and update the predicted X, Y position of the UAV. However due to the computation time needed to detect the landing marker, that incoming measurement will be delayed by some time dt_{obs}. During that time dt_{obs} the UAV will be relocated by an interval (dx_{obs}, dy_{obs}). To compensate that extra displacement, the interval (dx_{obs}, dy_{obs}) is calculated and added to the observed measurements.

Correction step:

$$\begin{aligned}
&e_t = z_t - H_t * \hat{x}_t \\
&S_t = H_t * P_t * H_t^\top + R_t \\
&K_t = P_t * H_t^\top * S_t^{-1} \\
&\hat{x}_t = \hat{x}_t + K_t * e_t \\
&P_t = (I - K_t * H_t) * P_t
\end{aligned} \tag{2}$$

where:

$$z_t = \begin{bmatrix} x_{obs} + dx_{obs} \\ y_{obs} + dy_{obs} \end{bmatrix}, \quad H_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \sigma^2_{obs}$$

$$dx_{obs} = vx_{mean_{obs}} * dt_{obs} \quad , \quad dy_{obs} = vy_{mean_{obs}} * dt_{obs}$$

A velocity estimator is also constructed to determine the magnitude and direction of the moving platform's velocity vector. The magnitude is calculated by differentiating two sequential X, Y position measurements of the moving platform and taking into consideration the UAV's *NED* velocities according to the PX4's EKF. A low-pass filter is used to provide a smooth estimation of the velocity's magnitude by filtering out high frequency noise. High frequency noise can be caused by oscillations of the UAV along with a fast update rate in the landing marker detection algorithm.

HEMI: Please explain why the signal is the *smoothest.*

In the agricultural use-case the moving platform is likely to change its direction up to 180 degrees. Furthermore, it is assumed that the moving platform is a nonholonomic system, like a tractor. To compensate for sudden turns, the moving platform's yaw orientation will be taken into account and based on the velocity's magnitude, the moving platform's velocity in X, Y *NED* frame can be obtained.

C. Wind Disturbances

The exact and accurate estimation of the applied wind forces on a body is a complex matter studied by the field of fluid dynamics. We use a simplified approach, as follows. We assume that the wind will be applied on an area of $0.09m^2$. Such an area is emulating a UAV with extra payload attached on its frame. Two different wind speeds of $8m/s$ and $12m/s$ will be used to calculate the applied wind forces on that area. The wind forces are considered to be applied on the center of gravity of the UAV with direction parallel to the ground.

The magnitude of the applied force, corresponding to a certain wind speed is calculated by the following equations [28], [29]:

$$\begin{aligned}
&F = p_d * A \\
&p_d = \tfrac{\varrho * v^2}{2}
\end{aligned} \tag{3}$$

where: F is the force, p_d is the dynamic pressure, A is the area of the applied pressure, ϱ is the density of the air (around 1.2 kg/m^3), v is the wind velocity in m/s.

By solving the above equations the applied force on the UAV is found to be $3.45N$ for an $8m/s$ wind speed, and $7.76N$ for a $12m/s$ wind speed. The wind forces will be applied on the UAV in Gazebo simulation, to test the performance of the whole system. This will be done by constructing a program that will apply the wind forces on the virtual UAV.

To simulate a wind pattern the wind direction and magnitude must be defined. The wind direction is assumed to remain the same for the whole duration of the tests. That direction vector is defined as $[0.8, 0.2]$ according to Gazebo's (x, y) axes. The magnitude of the wind is calculated as follows:

- An initial wind power of $7.0N$ is chosen.
- An update cycle of $5.5s$ is chosen between two different applied forces.
- A random float number between 0.9 and 1.2 is chosen at every update cycle. This number is defined as *Random_noise*.
- The applied wind force is calculated as:

$$\begin{aligned}
&force_x = -x_{norm} * power_{init} * Random\_noise \\
&force_y = -y_{norm} * power_{init} * Random\_noise
\end{aligned}$$

  where $x_{norm} = 0.8$, $y_{norm} = 0.2$, $power_{init} = 7.0N$
- For a UAV altitude of 6.0 meters and below the power decreases as: $power = power_{init} * (altitude/6.0)$, where $power_{init}$ is $7.0N$
- For a UAV altitude of 3 meters and below, $power = 0.5$.

This model is created to simulate different scaling in the wind's magnitude and will be used as it is for all simulated tests.

V. Evaluation

A. Use case: agricultural safety

Briefly describe the use case and simulation, including the use of

A multirotor UAV, with a vertically facing camera, is used to detect a moving platform. The moving platform follows a path similar to a plowing tractor's and no communication link is considered between the moving platform and the UAV. For this agricultural use case, two different UAV actions are tested.

The first action is to track and follow the moving platform from a fixed altitude, while detecting and classifying objects on the ground. Four different classes are selected, cars, humans, tractors and cows. Object detection and classification is performed by feeding the input image from the vertically facing camera into a $YOLOV3\ tiny$ CNN [30] implemented in ROS [31]. Furthermore, the UAV, based on the CNN's predictions and onboard sensors, maps the detected objects onto a 2D map. The second action is to track and land on the moving platform.

A simulated field is created in gazebo with objects placed in random positions and orientations as seen in figure 3.

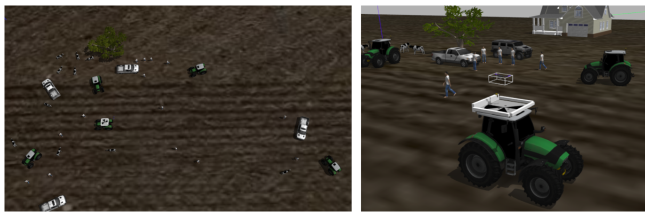

Fig. 3. On the left, a top view of the gazebo scene. On the right, a view of the UAV attempting a landing on the moving platform

Pre-trained weights for $YOLOV3\ tiny$, were initially used but the performance on detecting objects from a vertically facing camera was not the expected. Since an available dataset for detecting the above four classes was not available, we created an artificial dataset based on the gazebo models. This dataset consists of 1200 images, 300 for each class. We trained the $YOLOV3\ tiny$ by its default training parameters for 5000 epochs for an input image size of 416×416 pixels. For all the experiments, the input image to $YOLOV3\ tiny$ will be resized to a 416×416 pixel size.

B. Experimental setup

How the experiment will be done concretely, i.e., time to land as a function of QoS and wind or whatever.

All experiments are performed in gazebo simulation under Ubuntu 18.04 OS and ROS Melodic on a i7-8550U 1.8GHz (4.0GHz Boost), 8GB DDR4 laptop. The $PX4$ under SITL Firmware v1.10.2 is used as the flight controller and the $Iris$ quadcopter is used as the UAV platform. A vertically 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 OS and ROS Melodic is used as the UAV's companion computer. All the computer vision, guidance and control algorithms needed for an autonomous flight are implemented on the Jetson Nano.

Two groups of experiments are conducted. The first group evaluates the energy consumption and QoS of the "tracking" mode and the second group evaluates the energy consumption and QoS of the "landing" mode. In both groups, the experiments start with the tractor moving with a constant speed of $0.3m/s$, according to a square path similar of a plowing tractor, and the UAV taking-off and hovering at an altitude of $25m$. Once this altitude is reached, the UAV starts searching for the landing marker in the image frame. Once the landing marker is detected the UAV commences its actions.

On a "tracking" mode, the UAV will fly above the tractor at a fixed altitude and use its vertically facing camera to map the environment while on a "landing" mode, the UAV will position itself above the moving platform and will descend until it lands on it. Both "tracking" and "landing" modes are tested under three different cases of no wind disturbances, wind disturbances of $8m/s$ and wind disturbances of $12m/s$ according to the wind model described in section IV-C.

C. Results

Results of the experiments

D. Discussion

Discussion of the results

VI. Conclusion and Future Work

What we did, why it was exciting, and what we want to do.

Acknowledgment

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

References

[1] F. G. Costa, J. Ueyama, T. Braun, G. Pessin, F. S. Osório, and P. A. Vargas, "The use of unmanned aerial vehicles and wireless sensor network in agricultural applications," in *2012 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2012, pp. 5045–5048.

[2] E. Salamí, C. Barrado, and E. Pastor, "Uav flight experiments applied to the remote sensing of vegetated areas," *Remote Sensing*, vol. 6, no. 11, pp. 11 051–11 081, 2014.

[3] A. Seewald, H. Garcia de Marina, H. S. Midtiby, and U. P. Schultz, "Mechanical and computational energy estimation of a fixed-wing drone," in *Proceedings of the 2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2020, p. to appear.

[4] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 2799–2804.

[5] 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," *International Journal of Distributed Sensor Networks*, vol. 14, no. 9, 2018.

[6] S. Saripalli and G. S. Sukhatme, "Landing on a moving target using an autonomous helicopter," in *Field and service robotics*. Springer, 2003, pp. 277–286.

[7] 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 international conference on robotics and automation*. IEEE, 2012, pp. 971–976.

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

[9] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3400–3407.

[10] 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 International Conference on Control and Automation (ICCA)*. IEEE, 2016, pp. 761–766.

[11] H. Lee, S. Jung, and D. H. Shim, "Vision-based uav landing on the moving vehicle," in *2016 International conference on unmanned aircraft systems (ICUAS)*. IEEE, 2016, pp. 1–7.

[12] 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.

[13] 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.

[14] S. Kyristsis, A. Antonopoulos, T. Chanialakis, E. Stefanakis, C. Linardos, A. Tripolitsiotis, and P. Partsinevelos, “Towards autonomous modular uav missions: The detection, geo-location and landing paradigm,” *Sensors*, vol. 16, no. 11, p. 1844, 2016.

[15] 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 International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5783–5790.

[16] A. Seewald, U. P. Schultz, E. Ebeid, and H. S. Midtiby, “Coarse-grained computation-oriented energy modeling for heterogeneous parallel embedded systems,” *International Journal of Parallel Programming*, pp. 1–22, 2019.

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

[18] ——, “Mission energy prediction for unmanned ground vehicles using real-time measurements and prior knowledge,” *Journal of Field Robotics*, vol. 30, no. 3, pp. 399–414, 2013.

[19] 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 Transactions on Robotics*, vol. 25, no. 5, pp. 1098–1108, 2009.

[20] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, “Deployment of mobile robots with energy and timing constraints,” *IEEE Transactions on robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[21] ——, “A case study of mobile robot’s energy consumption and conservation techniques,” in *ICAR’05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* IEEE, 2005, pp. 492–497.

[22] ——, “Energy-efficient motion planning for mobile robots,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*, vol. 5. IEEE, 2004, pp. 4344–4349.

[23] V. Berenz, F. Tanaka, and K. Suzuki, “Autonomous battery management for mobile robots based on risk and gain assessment,” *Artificial Intelligence Review*, vol. 37, no. 3, pp. 217–237, 2012.

[24] 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.

[25] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, “Vision-based autonomous quadrotor landing on a moving platform,” in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, 2017, pp. 200–207.

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

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

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

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

[30] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv*, 2018.

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