Deutscher Luft- und Raumfahrtkongress 2023
DocumentID: 610274

SIMULATION TOOL FOR EVALUATING GN&C ALGORITHMS USING THE EXAMPLE OF AUTOMATIC LANDING OF AN UNMANNED HELICOPTER ON A MOVING PLATFORM

R. M. Leitner and L. Ollagnier
IABG mbH, Innovation Centre, Einsteinstrasse 20, 85521 Ottobrunn, Germany

Abstract

Unmanned aerial vehicles are becoming increasingly important in modern battlefield. In general, unmanned rotorcrafts are used for local reconnaissance. Among other things, these can be launched and landed vertically from a stationary ground vehicle and can reconnoiter the surrounding area at a certain altitude above the ground vehicle. Because of the many disadvantages in a military context, a wireless solution is preferable to a wired solution. A wireless solution gives the ground vehicle significant flexibility and enables take-offs and landings of the unmanned aircraft were possible while the ground vehicle is moving. To examine under which conditions - e.g., environmental conditions, etc. - the guidance navigation and control (GN&C) algorithms for automatic take-off and landing perform satisfactorily, a real-time test environment has been developed based on numerical simulation of the aircraft-ground vehicle, sensor technology and sensor data fusion for relative state estimation. The test environment realistically reproduces the take-off and landing processes of the rotorcraft UAV while the ground vehicle is in motion. The developed algorithms for UAV GN&C can be used in a real flying demonstrator at a later stage. The paper details the development and setup of the test environment and its components.

Keywords
Automatic Take-off and Landing, ATOL, Helicopter, Computer Vision, Sensor Fusion, Real-Time Simulation

1. INTRODUCTION

Flying close-range reconnaissance systems for ground vehicle consist in an Unmanned Aerial Vehicle (UAV) that serves as a sensor carrier and is often connected to the ground vehicle with a cable. Landing on the ground vehicle is achieved by vertical descent. This enables UAV operation while the ground vehicle is moving at a low speed. Although the cable ensures a reliable communication between the ground vehicle and the UAV, it also restricts the movements of the UAV and the ground vehicle. To avoid restricting the movements of the UAV and of the ground vehicle, wireless communication can be used. Guidance, Navigation and Control (GN&C) algorithms for low speeds have to be adapted to achieve a successful take-off and landing on a moving platform at higher speeds.

To evaluate GN&C algorithms for automatic landing of an unmanned helicopter on a moving platform, a simulation tool was developed and is presented in this paper.

2. ARCHITECTURE

The architecture of the system is based on three subsystems: visualisation, image processing and sensor fusion, and real-time emulator.

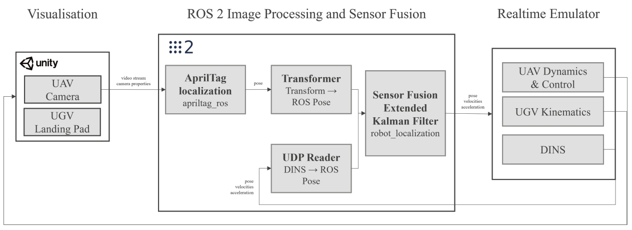

Visualisation
ROS 2 Image Processing and Sensor Fusion
Realtime Emulator
unity
UAV Camera
UGV Landing Pad
video stream camera properties
2
AprilTag localization apriltag_ros
pose
Transformer Transform → ROS Pose
Sensor Fusion Extended Kalman Filter robot_localization
UDP Reader DINS → ROS Pose
pose velocities acceleration
pose velocities acceleration
UAV Dynamics & Control
UGV Kinematics
DINS

FIG 1. Architecture overview of the system for automated take-off and landing

2.1. Visualisation

The first subsystem is visualisation. A UAV with its camera and a ground vehicle are visualised in a virtual environment, developed using Unity 18.04, which contains a test track. Real conditions like occlusions, fog, etc. can be integrated into the visualisation to evaluate GN&C algorithms. To move the ground vehicle and the UAV, the visualisation receives data from the real-time emulator. The visualisation is used to create a video stream for the image processing. To do so, a downward facing camera is attached underneath the UAV. The camera is mounted on a gimbal which can be controlled via the emulator to track the landing platform. The video stream from the visualisation is sent via network to the image processing and sensor fusion subsystem.

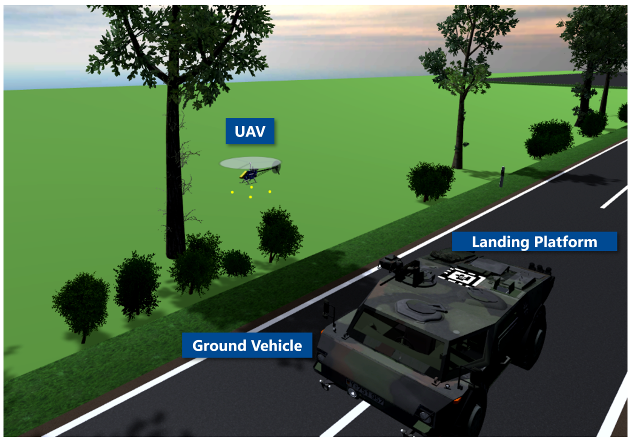


FIG 2. Screenshot of the visualisation with the UAV, the ground vehicle and the landing platform

2.2. Image processing and sensor fusion

Processing consists in two steps. The first step is image processing to assess the relative position and velocity of the UAV using the video stream from the camera. The second step is sensor fusion using all available sensor data to generate a robust state estimation, see Figure 1. This robust state estimation is sent via network to the real-time emulator.

2.3. Real-time emulator

The real-time emulator is equipped with three functions. The first function is simulation and control of the UAV dynamics. The second function is simulation of the vehicle kinematics. The third function is estimation of the position and velocity of the UAV relative to the ground vehicle using INS with differential GNSS and datalink on both vehicles. The datalink is used to send INS data from the ground vehicle to the UAV. In the paper, the combination of the INS, the differential GNSS and the datalink is called differential INS (DINS).

The real-time emulator receives from sensor fusion the state estimation of the UAV relative to the ground vehicle. Based on this information, appropriate commands are sent to control the UAV dynamics in its tasks.

3. SIMULATION AND CONTROL

The simulation mimics the dynamics of a controlled real unmanned micro helicopter with one main rotor and one tail rotor, see Figure 3, and ground vehicle kinematics, and the DINS state estimation.

3.1. Controlled UAV

The helicopter UAV is equipped with a three-bladed rotor head and the landing gear is robustly designed to allow hard landings on the moving platform. The rotrocraft is driven by a brushless electric motor.

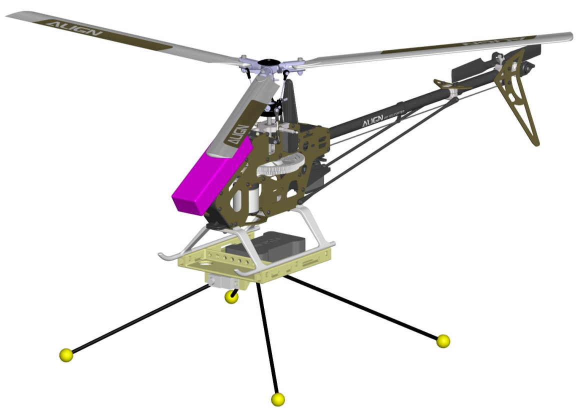


FIG 3. CAD model of the real UAV

The cyclic and collective control of the swash plate is done by three electromechanical actuators. These are arranged in a Ypsilon configuration around the main rotor shaft. The tail rotor blade pitch is actuated by an actuator near the UAV's centre of gravity, which is connected to the tail rotor control mechanism by a thin, rigid rod.

The key data of the UAV can be taken from Table 1.

TAB 1. Data of the UAV

Description	Value	Unit
Total Mass	1026	g
Length without landing gear and rotors	600	mm
Diameter of the rotor	724	mm
Diameter of the tail rotor	164	mm
Height with landing gear	335	mm
Virtual flapping hinge offset of the rotor blade	37	mm
Chord length of the rotor blade	32	mm
Flapping moment of inertia of the rotor blade	0.88	gm^2
Lift curve slope of the rotor blade	5.59	1/rad
Operating speed of the rotor	2000	rpm
Spring stiffness of the rotor blade	0.5	Nm/rad
Damper constant of the rotor blade	0.01	Nms/rad
Lock number of the rotor blade	2.78	-

The UAV consists of a fuselage, two rotors with two rotor blades for the tail rotor and three rotor blades for the main rotor, and a landing gear with four landing legs. The topology of the UAV model with respect to the Earth, which is assumed to be fixed and flat, is shown in Figure 4.

The Earth (E, e) with reference point E and base e represents the inertial system in which the equations of motion of the UAV models are developed.

The fuselage (Bo, b, B) is connected to the earth by a 6-DOF hinge Bo^{6d}, which consists of three linear and three angular degrees of freedom. Four landing legs (L, l, L), which are coupled to the fuselage by a single hinge G^{1d}, allow interaction of the UAV with the ground. The two rotor hubs (R, r, R), each equipped with a single-axis hinge R^{1d}, are rotatably mounted in the fuselage and are driven by an electric motor.

The rotor blades (M, m, M) are each connected by a single-axis hinge G^{1d} to the corresponding rotor hub to enable the flapping motion and to ensure horizontal propulsion of the helicopter by tilting the tip-path plane of the rotor longitudinally or laterally.

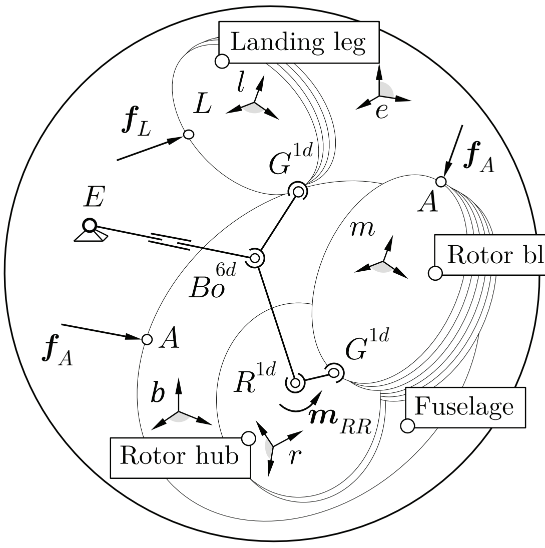


FIG 4. Topology of the UAV model

The equations of motion of the helicopter model are based on the principle of virtual velocity, also known as Jourdain's principle, see [1], which "couples" the equations of motion of the rigid bodies of the helicopter multibody model, see (1), with Jacobi matrices $\boldsymbol{J}_\eta$ and $\boldsymbol{J}_q$ that implicitly contain the helicopter linear and angular degrees of freedom.

$$\text{(1)} \qquad \boldsymbol{M}\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\eta^\top \left[\boldsymbol{e} - \boldsymbol{j} - \boldsymbol{g}\right]$$

The term $\boldsymbol{M} = \boldsymbol{J}_\eta^\top \boldsymbol{M}^\star \boldsymbol{J}_\eta$ represents the mass matrix. The generalised forces and moments correspond to the product of the transposed Jacobian $\boldsymbol{J}_\eta^\top$ and the force and moment vector $\boldsymbol{e}$. The variable $\boldsymbol{j}$ is equivalent to $\boldsymbol{M}^\star \boldsymbol{J}_q \boldsymbol{\eta}$.

In the multibody model of the helicopter, 13 rigid bodies are geometrically coupled using ideal hinges and bearings, so that the model exhibits 18 degrees of freedom. Detailed information on the multibody dynamics of the two helicopter models can be found in [2].

3.2. Ground vehicle

In contrast to the UAV, the movement of the ground vehicle is simplified to ensure the repeatability of the experiments. This is achieved by a spline trajectory on which the ground vehicle moves. With the vehicle speed and the actual curvature calculated using the spline coefficients, it is possible to very realistically approximate the roll angle of the ground vehicle when cornering.

The spline trajectory is based on waypoints. Two waypoints are connected by a cubic polynomial per spatial axis. The k^{th} polynomial, see (2), extends from the k^{th} to the $(k+1)^{th}$ waypoint.

$$\text{(2)} \qquad f_k(s) = A_k \Delta s_k^3 + B_k \Delta s_k^2 + C_k \Delta s_k + D_k$$

The subinterval of the polynomial Δs_k corresponds to $s - s_k$. For N waypoints in three-dimensional space, $3(N-1)$ polynomials result. Figure 5 shows the structure of the kinematic motion model. The motion model is programmed in Simulink® and uses a discrete integrator and lookup tables where the spline coefficients of the trajectory are stored and interpolated. The input to the generator is the speed of the ground vehicle. The outputs include the three-dimensional Cartesian coordinates of the ground vehicle's reference point.

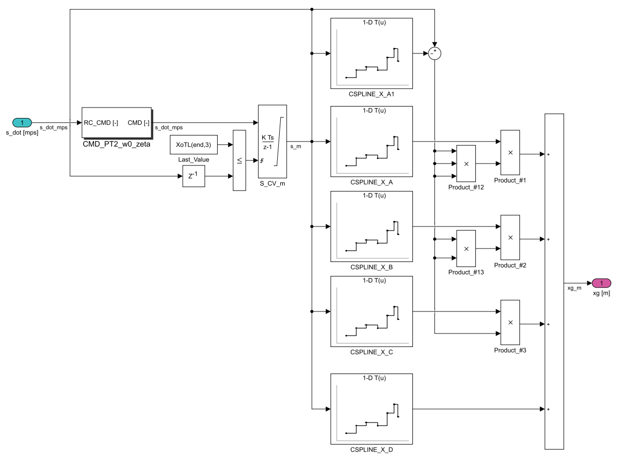


FIG 5. Spline-based motion model of the ground vehicle

The properties of the real-time emulator system are as follows:

- Software: B&R Automation Studio 4.3 and MATLAB and Simulink
- Hardware: B&R Automation PC 910
- Network: Switch, router and development laptop

4. SENSORS, COMPUTER VISION AND SENSOR FUSION

4.1. Navigation and Algorithms

Navigation is a field of study that focuses on the process of monitoring and controlling the movement of a craft or vehicle from one place to another.

In order to provide guidance and navigation, systems have to be equipped with sensors. Common choice for sensors are optical sensors, inertial sensors and satellite-based sensors. Satellite-based sensors provide an absolute positioning solution which is accurate up to a few centimeters but which is easily jammed or spoofed and can be completely unavailable in GNSS-denied areas. Such issues can be tackled by using several types of sensors and merging the information using sensor fusion.

One way to navigate, without satellite navigation, is using camera-based systems and computer vision. Two types of algorithms for optical navigation were considered: Simultaneous Localisation And Mapping (SLAM) or fiducial-based localisation. Visual SLAM uses features to compute the movement of the sensor between two data frames. Features are keypoints in camera images. These keypoints can be, for example, corner-like points detected by comparing the intensity of pixels in a defined area. Fiducial, or fiduciary markers are image like objects, which are designed to be easily detectable using a camera and which contain an interpretable meaning (id). Fiducials are used in many field, including robotics, augmented reality and medicine. Such markers can be used for localisation improvements. Fiducial-based localisation estimates the position of the marker relative to the sensor based on the marker size and orientation in the image.

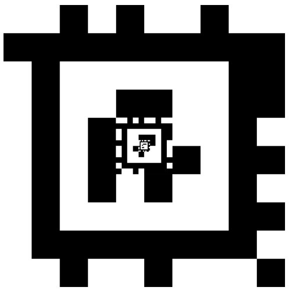


FIG 6. Three level AprilTag of the 48h12 family, with the ids 0, 1 and 2.

In this work, fiducial localisation was selected. Each marker is unique and has a known position. AprilTag [3] is a C++ library used for the detection of fiducials which offers several marker families adapted for different uses and an open-source ROS wrapper apriltag_ros. AprilTag detection was chosen using a custom three-level tag of the 48h12 family, shown in 6 for its multi-level property, making it detectable from a large range of distances. Since fiducial localisation does not rely on machine learning, it can run in real-time with limited computational resources. A 65 cm multi-level AprilTag is printed on a platform placed on top of the ground vehicle. Its size was chosen so that it can be detected by the camera onboard the UAV from a distance of up to 15 meters.

4.2. Sensors

This paper tackles the creation of a simulation tool used to evaluate GN&C algorithms. Two INS with differential GNSS and a camera were created in the simulation.

The model of the camera in the simulation has the following properties: focal length and sensor size. Focal length is 85 mm. Sensor size is kept to a classic value for image sensors: 36 x 24 mm frame. The camera properties were chosen to match as closely as possible real sensors available on the market while keeping in mind that the AprilTag was to be detected from a distance of 15 meters.

The inertial sensor on each system is an Xsens MTi 680, a high-performing GNSS/INS with an integrated RTK receiver. The sensor properties were set in the simulation to match those of the real sensor. The data from both differential INS is processed by the real-time machine to compute a relative position: the position of the AprilTag relative to the camera. The time and relative state estimation is sent via UDP network message to the ROS 2 processing node.

In order to achieve accurate navigation, markers within the environment must be properly detected and the system must output a position with a frequency of at least 10 Hz. Markers must be detected in a range of 0.1 meters to 10 meters.

The key properties for a camera for marker-based navigation are field of view, shutter type, pixel resolution and lens type. Hence the following desired properties:

- Field of View: at least 80 degrees
- Shutter type: Global shutter to reducing blurring during motion
- Colour sensor resolution: at least 1 MP
- Lens type: undistorted

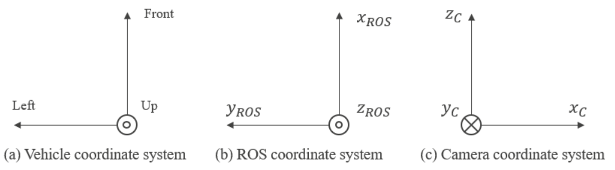
The system computes the full 3D pose of the ground vehicle relative to the UAV at all time. In order to achieve this, the system takes two inputs:

1) relative position data obtained by a datalink between the ground vehicle and the UAV, both equipped with an INS
2) video stream from the downward facing camera onboard the UAV.

4.2.1. AprilTag Detection

The AprilTag detection is done using a the ROS package called apriltag_ros. The library outputs the position and orientation $T_{CA,cam}$ of the AprilTag A wrt. the camera C in the right handed coordinate system: x_C-right, z_C-front and y_C-down.

The goal of the detection is to compute the transformation $T_{HG,H}$ of the ground vehicle G wrt. the UAV H. Figure 7 illustrates the coordinate systems.


Front
Left
Up
(a) Vehicle coordinate system
x_{ROS}
y_{ROS}
z_{ROS}
(b) ROS coordinate system
z_C
y_C
x_C
(c) Camera coordinate system
FIG 7. Vehicle (a), ROS (b) and camera (c) coordinate systems used for automated takeoff and landing.

The position $\boldsymbol{r}_{HC,ROS}$ describing the position of the camera C on the UAV, with reference point H, in ROS coordinate system ROS and $\boldsymbol{r}_{AG,ROS}$ describing the position of the AprilTag A on the ground vehicle, with reference point G, in ROS coordinate system and are known and are as follows in (3) and (4).

$$\boldsymbol{r}_{HG,ROS} = \boldsymbol{r}_{HC,ROS} + \boldsymbol{r}_{CA,ROS} + \boldsymbol{r}_{AG,ROS} \tag{3}$$

$$\boldsymbol{r}_{HC,ROS} = \begin{bmatrix} 0 \\ 0 \\ -0.08 \end{bmatrix}, \ \boldsymbol{r}_{AG,ROS} = \begin{bmatrix} 0 \\ 0 \\ -0.2675 \end{bmatrix} \tag{4}$$

4.2.2. Fiducial Marker Choice

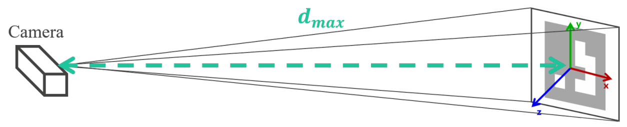


FIG 8. Maximum distance at which a sensor can be detected

Figure 8 illustrates the maximum distance at which a tag can be detected using a camera, where

$$d_{max} = \frac{t}{2 \cdot \tan(\frac{b \cdot f \cdot p}{2 \cdot r})}. \tag{5}$$

In (5), t is the tag size in meters, b is the number of bits that span the width of the tag (including the white border for AprilTag 2), f is the horizontal field of view of the camera in radians, r is the horizontal resolution of the camera in pixels and p is the number of pixels required to detect a bit (5 is recommended, lower number recommended is 2 which is the Nyquist frequency).

4.2.3. Preparing data for filtering

Before sensor data can be integrated in the filter, it is adapted to fit the requirements of the system:
1) matching timestamp
2) respecting the coordinate system of ROS for translation and rotation
3) setting the covariance matrix for the measurements

All subsystems are part of the same local network but do not necessary have the same clock. Matching the timestamp requires ensuring a common clock is used in all the subsystems. In this use case, the clock chosen is that of the simulation and it is sent to all subsystems via UDP network message. The clock is received with the same delay by all subsystems. Therefore the delay can be compensated for by a simple offset.
The test scenario for ATOL can be illustrated as in figure 9, where P is the centre of the plateform on the ground vehicle and B is the centre of the UAV.

ROS uses a right-handed coordinate system with the convention of x-forward, y-left and z-up. There exists many different conventions for coordinate system, including coordinate systems for optical sensors and

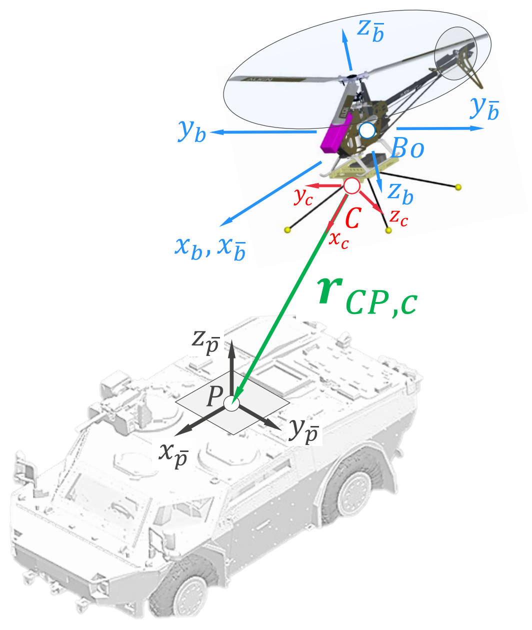


FIG 9. ATOL scenario and coordinate system

for the simulation in Unity, see Figure 7. Each pose received as an input to the Kalman Filter was transformed into the right-handed coordinate system from ROS.

The covariance matrix for the measurements is a required element for robot_localization [4]. It conveys how much a variable is to be trusted by the filter. It can be provided by the manufacturer when using sensor data.

The covariance matrices (linear acceleration, angular velocity, orientation covariance) for the INS are given by the datasheet of the ROS driver for the sensor :

$$\begin{aligned} Cov_{linear} &= \begin{bmatrix} 0.0004 & 0 & 0 \\ 0 & 0.0004 & 0 \\ 0 & 0 & 0.0004 \end{bmatrix}, \\ Cov_{angular,deg} &= \begin{bmatrix} 0.025 & 0 & 0 \\ 0 & 0.025 & 0 \\ 0 & 0 & 0.025 \end{bmatrix}, \\ Cov_{orientation,deg} &= \begin{bmatrix} 1. & 0 & 0 \\ 0 & 1. & 0 \\ 0 & 0 & 9. \end{bmatrix}. \end{aligned} \tag{6}$$

For fiducial-based localisation, hand-tuning the matrix based on trial-and-error was necessary to ensure correct handling of the measurement data by the filter.

4.3. Sensor Fusion

Sensor fusion is the ability to bring together inputs from multiple sensors, such as radars, IMUs and cameras, to form a single model or image of their surroundings. The resulting model is more accurate, because it balances the strengths of the various sensors, and more robust because it relies on several inputs. Systems based on visual navigation can use the information provided through sensor fusion to improve localisation and support more-intelligent actions.

The aim of object tracking is estimating current object state using noisy measurement of the object state. The process of state transitions is a stochastic process which can be achieved with a very popular recursive filter: a Kalman filter. Its applicability is however limited to linear dynamic systems with Gaussian noise. Kalman filters can be adapted in case of non-linear systems with Extended Kalman Filters (EKF) or Unscented Kalman Filter (UKF).

In case of non-linear transition function or measurement function or if noises are not Gaussian, an EKF or an UKF can be used. The EKF is based on a local linearization of the non-linear transition and measurement functions. The UKF uses a deterministic sampling technique to pick a minimal set of sample points around the mean. These points are then propagated through the non-linear functions, from which the mean and covariance of the estimate are recovered.

The software package robot_localization [4], for the Robot Operating System, contains an implementation of an EKF. It can support an unlimited number of inputs from multiple sensor type, including odometry and inertial measurement unit. Users can customize which sensor data fields are fused with the current estimate. Process noise covariance matrix and initial estimate covariance matrix are also taken as parameters to enable precise tuning.

The estimation of the position of the ground vehicle relative to the UAV can be described as a nonlinear dynamic system, shown in (7)

$$x_k = f(x_{k-1}) + w_{k-1} \tag{7}$$

where x_k is the state vector of the vehicle at time k , f is a nonlinear state transition function and w_{k-1} is the process noise, which is assumed by robot_localization [4] to be normally distributed. The 12-dimensional state vector consists in the 3D pose of the vehicle (translation and rotation) and velocities (angular and linear).

The projection step of the Kalman filter projects the current state estimate and error covariance forward in time. This step involves the process noise covariance which pertubes the estimate error covariance. The robot_localization [4] package exposes parameters such as a sensor configuration matrix to select which variables should be fused, outlier filtering based on Mahalanobis distance and tuning of the process noise covariance matrix and the initial estimate covariance. The process noise covariance matrix can be difficult to tune, and can vary for each application, so it is exposed as a configuration parameter in robot_localization. This matrix represents the noise added to the total error after each prediction step. The initial estimate covariance represents the initial value for the state estimate error covariance matrix. Tuning these matrices helps improve the pose estimation.

4.4. Communication

The properties of the processing system are as follows:
- Operating System: Ubuntu 20.04 LTS
- ROS version: ROS 2 Foxy
- Network: Router and debugging laptop

The Robot Operating System (ROS) is a communication software framework based on the concepts of "nodes" and "topics". A node is a unit of computation responsible for a specific task. Nodes communicate through channels which are called topics. ROS 2 [5] was used in this project for a first simulation, for its many open-source packages and for network communication.

Coordinate systems are defined as part of the ROS axis orientation standard REP103. In relation to a body, the standard is: x forward, y left, z up. For short-range Cartesian representation of geographic locations, the standard is the east north up convention: x east, y north, z up.

5. CONCLUSIONS AND NEXT STEPS

Within the scope of this paper, a simulation tool for designing and evaluating GN&C algorithms using the example of automatic landing of a UAV on a moving platform was developed. This real-time simulation tool enables testing various UAV models and GN&C algorithms. Various control algorithms can be tested with the tool. Navigation can be evaluated using different types of sensors and parameters for sensor fusion. Guidance performances are assessed by how accurately the landing is performed. The performances of the tool can also be evaluated under different disturbances like UAV loading, terrain profile, atmospheric conditions, noise on guidance signals and delays.

Contact addresses:

roland.leitner@iabg.de and ollagnier@iabg.de

References

[1] P. E. B. Jourdain. Note on an analogue of gauss' principle of least constraint. Q. J. Pure Appl. Math. 8L, page 153–157, 1909.

[2] R. M. Leitner. Modelling and Verification of Helicopter Multibody Dynamics for different Rotor Configurations. PhD thesis, RWTH Aachen, 2022.

[3] John Wang and Edwin Olson. AprilTag 2: Efficient and robust fiducial detection. October 2016.

[4] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. July 2014.

[5] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics, 7(66):eabm6074, 2022. DOI: 10.1126/scirobotics.abm6074.