Visual odometry for real-time localization of dynamic locomotion devices

Savorona Kostyantin
Computer Science, UCU
Lviv, Ukraine
savorona.pn@ucu.edu.ua

Ivaniuk Oleksandr
Computer Science, UCU
Lviv, Ukraine
ivaniuk.pn@ucu.edu.ua

Humeniuk Denis
Computer Science, UCU
Lviv, Ukraine
humeniuk.pn@ucu.edu.ua

Parnosov Nazar
Computer Science, UCU
Lviv, Ukraine
parnosov.pn@ucu.edu.ua

Krasnianskiy Tymur
Computer Science, UCU
Lviv, Ukraine
tymur.krasnianskyi@ucu.edu.ua

***Abstract*—This project explored different methods of visual odometry for real-time localization, highlighted their main advantages and disadvantages, what is used in choosing the most suitable one, and what can be applied to dynamic locomotion devices and also considered hardware specifications and requirements for using that algorithm with microcomputers.**

I. Introduction

Automated systems are developing more and more every day. At the same time, their requirements are increasing. One of the directions that needs development is the automation of the movement of devices in difficult conditions, which include the lack of high-quality connection. In this case, one of the main tasks of such processes is visual analysis of the environment. Often, on some devices, it is necessary to track the movement of the camera and predict its coordinates. And the tasks described are mostly performed using visual odometry.

It is also important to mention that there are a lot of devices that require instant frame processing, including unmanned aerial vehicles (UAVs). This is primarily because their working conditions are extremely demanding, and the market requires modern solutions capable of performing the tasks of more accurate and, most importantly, reliable use of UAVs [3]. Given the above, we will further consider the use of the algorithm for UAVs and similar devices.

II. Problem

Various technologies are used for UAV localization and control, but the most important is GPS and connection with remote control. Without it, the device will simply lose its orientation in space and will not be able to determine its location, and, accordingly, without connection with the remote control, it may simply be lost without completing its task. Therefore, it will be relevant to explore a way to ensure drone localization, namely the use of visual odometry. From that, the goal emerges: to develop a way to ensure the autonomous localization of UAVs in conditions of bad connection using visual odometry methods, taking into account hardware requirements and capabilities.

III. Requirements

To develop a device, it will first of all be appropriate to consider the requirements of its operation to select the hardware and implement the algorithm. The hardware and software requirements are as follows.

A. Hardware requirements

One of the frequent tasks of unmanned vehicles, which will be considered in this work and can be improved by real-time localization using visual odometry, is to follow so-called "routes", which, for example, is often used in video filming. To plan such routes, it is necessary to have:

- **GPS coordinates**: Provide the location data necessary for navigation.
- **Stabilization**: Ensure stable movement and filming, which is particularly important for smooth video footage.
- **Camera**: Capture video footage along the route.
- **Height measurement**: Determine the altitude of the device.
- **Direction determination**: Know the orientation of the device to navigate accurately.

B. Software requirements

In the face of a cluster of communication devices and a diverse landscape, GPS localization can be unstable and not accurate enough [8]. When organizing the automated movement of the device, you need to take this into account. In the absence of communication, the device should be able to localize without it and continue its planned route. Therefore, it is necessary to ensure the following requirements at the program level:

- **Missions**: Organization of missions based on GPS coordinates.
- **Localization**: Instant determination of device coordinates based on the environment in the absence of reliable communication.

IV. Hardware for applying algorithms

In projects involving mobile platforms such as drones or autonomous vehicles, selecting the right hardware is critical. This document outlines our approach to selecting and integrating key hardware components to meet our project's requirements for navigation, stability, video capture, altitude measurement, and orientation.

We have chosen a popular GPS module (see Fig. 1) to provide accurate location data. The module supports both GPS and GLONASS to improve accuracy and reliability in different geographical regions. We need it to obtain the initial coordinates and to test and evaluate the accuracy of the algorithm. Additionally, we selected the MATEKSYS GNSS compass (M10Q-5883/HP024.0115) because of its high sensitivity and multi-constellation support, which provides good positional accuracy, the MATEKSYS has an integrated magnetometer and barometer, offering more comprehensive data integration in a compact form factor, increasing system reliability and reducing the need for additional sensors.

For stabilization, we use INAV firmware integrated with a compatible flight controller that supports both multi-rotor and fixed-wing aircraft.

We use a high-quality camera module that connects directly to our main processor via a high-speed interface, enabling high-quality video capture. This setup is optimized for sensitivity in different lighting conditions. We use a barometer for accurate and fast altitude measurements. It provides better accuracy than GPS altitude data, especially in scenarios where altitude changes are frequent and subtle.

We integrate an inertial measurement unit (IMU), which combines an accelerometer and a gyroscope to provide us with orientation and displacement data in 6 axes. Previously, our IMU also combined a magnetometer that was part of the GPS module, but this caused a lot of fluctuation in the YAW. So we dropped it.

We selected the SpeedyBee F405 V3 (Fig. 1) because it offers a robust built-in sensor suite that simplifies setup and scaling and reduces compatibility issues. In addition, this controller supports a wide range of communication protocols, has enough I/O ports for further expansion, and is known for its reliable performance and easy firmware updates.

We chose the Raspberry Pi 4 as the main computer equipped with a quad-core ARM Cortex-A72 processor and 4GB RAM. A general view of our system is shown in Fig. 1.

V. UAV configuration

For the configuration of our UAV, we are using iNav – a cross-platform flight control configuration tool. The main points on which we are focusing in iNav:

- **Setting up GPS coordinates.** We can set up UART to receive GPS from Compass using Mavlink packages or Raspberry PI with the help of NMEA packages. Also, we can set up a data rate and disable arming if the device doesn't get GPS before the flight.

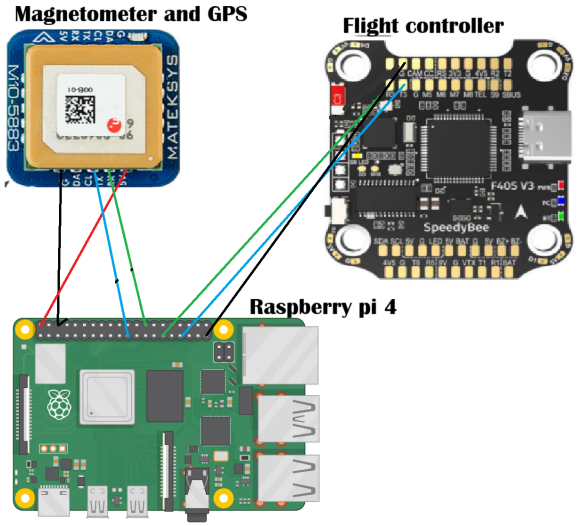


Fig. 1: General view of our system.

- **Setting up the remote control**. When we want to control the drone remotely, we need to set up communication parameters: SBUS protocol, bounds of throttle/roll/pitch/yaw, etc.
- **Troubleshooting**. When something goes wrong with arming or receiving information from external modules, problems can be debugged and fixed directly in Inav.
- **Setting up missions**. In iNav, we can set missions for the drone in the format "Fly to A – then B – then C and come back". It is an important feature for testing approximate GPS coordinates in our further steps.

VI. Visual Odometry

In tasks that require the work of the device observer with images and motion, the use of visual odometry could be beneficial [1].

A. Approaches

There are two primary approaches to applying visual odometry: feature-based and appearance-based (direct) approaches [1], [2].

The feature-based approach entails extracting features from images, like corners, lines, and curves, across consecutive frames, identifying and tracking distinctive features among them, and ultimately determining the motion. Such methods have been successful in visual odometry due to the availability of robust feature detectors and descriptors [1]. However, feature-based approaches have several disadvantages [2]:

1) Dependence on choice: the accuracy of feature-based methods is sensitive to the choice of detection and matching thresholds;
2) Necessity for robust estimation techniques to deal with wrong correspondences;
3) Most feature detectors are optimized for speed rather than precision.

Appearance-based approach checks changes in the appearance of the frame and estimates motion directly. More precisely, it uses local intensity gradient magnitude and direction for the optimization of correctness compared to the feature-based approach. From statistics of testing appearance-based method [2] it is good with:

1) scenes with little textures;
2) camera defocus;
3) motion blur;

Also, it saves time by not pulling out features but analyzing frames directly [1].

VII. Algorithm

A. Introduction

This project utilizes monocular visual odometry to estimate the motion of a camera through sequence image analysis. Employing a single camera setup reduces hardware complexity and costs, making this approach valuable for dynamic locomotion devices where weight and power consumption are critical. The algorithm is designed to integrate visual data with GPS input, ensuring robust localization even when GPS signals are unreliable or absent, and work in real time.

B. Choice of Libraries

For the implementation of our visual odometry algorithm, we used several libraries:

1) OpenCV – essential for real-time feature detection and optical flow algorithms.
2) GeographicLib – for accurate computation of GPS coordinates using relative displacements [4].
3) MAVSDK – for communication with UAV hardware [5].
4) libserialport – for communication with GPS modules.

C. Data Flows

The algorithm processes data through several stages:

1) Input images: Continuous image capturing from the onboard camera.
2) Feature detection and matching, using the OpenCV ORB detector to detect and match objects in consecutive frames.
3) Motion calculation: Calculates motion based on the movement of relevant objects, scaled using depth information from GPS Module and telemetry from Flight Controller.
4) GPS data integration, providing real positioning for our experiments.
5) Output: The system outputs the GPS coordinate trajectory, combining visual odometry and GPS data to accurately track the movement of the drone.

D. Implementation

GPS data integration corrects for drift inherent in visual odometry by applying corrections based on new GPS readings. This is essential for autonomous operation devices with dynamic movement like drones.

The implementation utilizes the capabilities of the Raspberry Pi 4B+, applying multi-threading to manage real-time data processing and ensure that the navigation tasks are performed efficiently.

VIII. Testing

An integral part of our project is the testing of implemented solutions to check their correctness, which also requires the simulation of real conditions and requirements. This work considers two methods: simulation and the real world.

A. Real-world testing

Since aircraft are quite vulnerable to external influences, conducting testing in real conditions is important. Before their implementation, we determined the main criteria for evaluating the tests and the requirements for the conditions in which they will be conducted.

Metrics:

1) Mean square error.
2) Graphs of trajectories of real GPS compared to predicted ones.

Conditions:

1) Non-linear motion trajectory.
2) Speed change.
3) Average flight altitude.

B. PX4 Autopilot

When conducting real tests, there are great risks of damaging the device with minimal malfunctions. So, using the simulations is required.

We considered two possible options: Gazebo-garden and PX4 Autopilot. The Gazebo-garden had serious drawbacks – informally speaking, it was too modern, lacked documentation, and was unable to work with some established parts of our system.

PX4 Autopilot is quite an optimal choice for testing because it has all the functionality required and has clear documentation. To simulate flights, it uses gazebo-classic, MAVLink communication protocol, and others [7].

For more convenient testing, it was decided to use the remote control to control the drone in the simulator in real time which leads to more realistic performance. For this, we used the SBU protocol and developed a corresponding converter using the MAVSDK library [5].

Receiving video from the simulator was accomplished using GStreamer, with the following pipeline [6]:

Listing 1: GStreamer pipeline code

```
gst-launch-1.0 -v udpsrc port=5600
caps='application/x-rtp,
media=(string)video,
clock-rate=(int)90000,
encoding-name=(string)H264' !
rtph264depay ! avdec_h264 !
videoconvert ! autovideosink
```

An example of the simulation is shown in Fig. 2.

htb

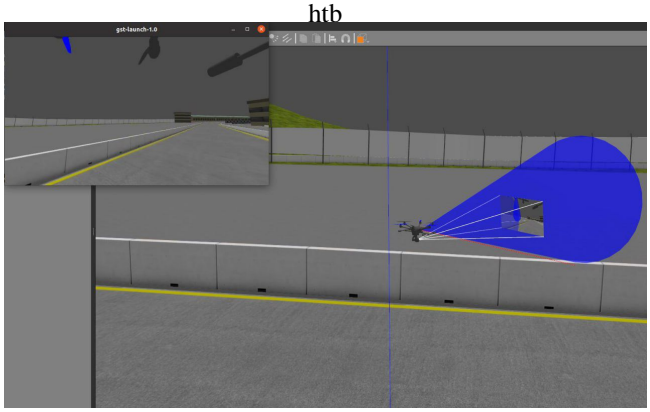


Fig. 2: PX4 simulator

IX. Results

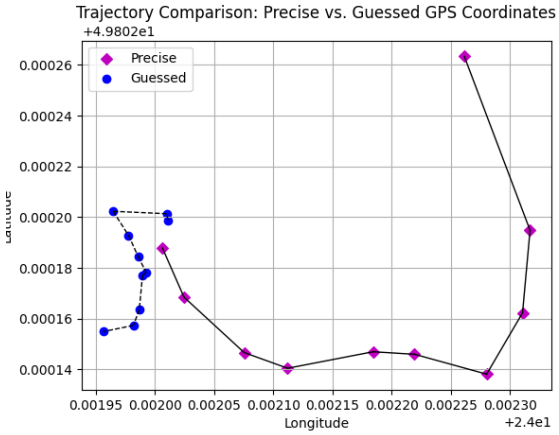


Fig. 3: First test.

Several test flights were performed to determine the correctness of our algorithm:

- **First**: at a low altitude with a stable speed. As we can see in Figure 3 and 4, the trajectory is similar, but it has a different scale and a different orientation. We identified the following reasons for the deviation: the algorithm does not take into account the orientation from the compass. Also, it does not determine the height to calculate the pixel scale.
- **Second**: In the second test, the problem with the orientation is corrected, and directions are close (see Fig. 5), but the pixel scale still does not depend on the altitude of the device.
- **Third**: the test was carried at an average flight altitude with a variable speed and a non-linear trajectory. Built-in sensors of the flight controller were used to determine the direction and orientation in place of the GPS. The altitude and tilt of the camera is taken into account in determining the pixel scale. The obtained trajectory, presented in Fig. 6a and Fig. 6b, is close to the real trajectory both in scale and orientation. The mean square error of the estimated trajectory is shown in Fig. 7.

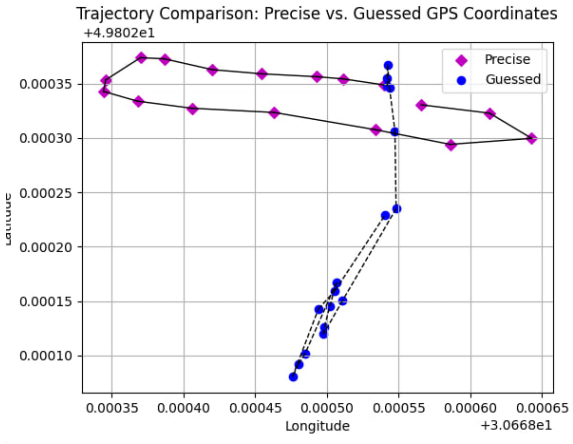


Fig. 4: Second test.

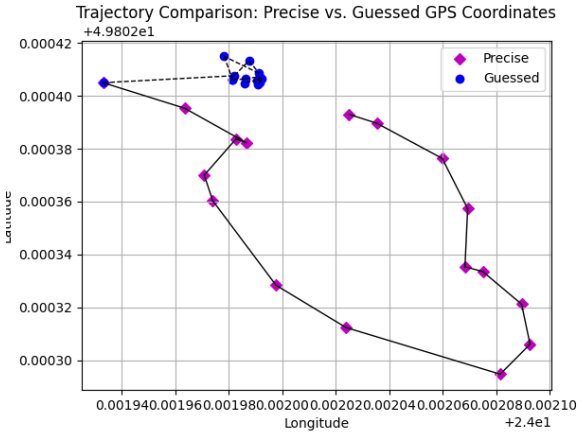


Fig. 5: Third test.

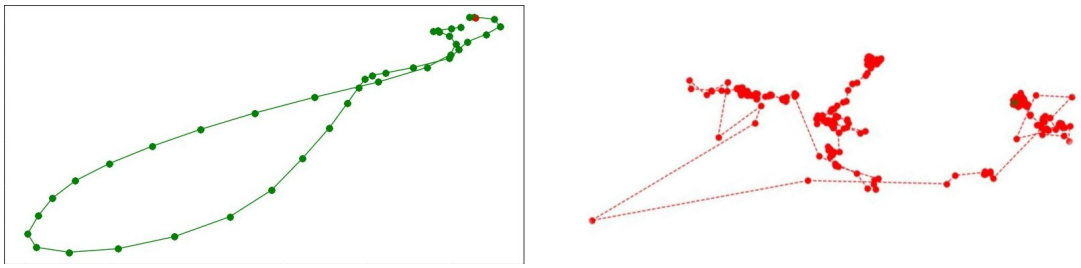
As we can see from this distribution, most errors are lower than $0.002°$, but values up to 0.014 are present. The problem is that when there is a big turbulence during the flight, the approximation becomes worse. The solution is to develop an algorithm that will also consider the intricacies of flying a drone with no stabilization.

X. Problems and troubleshooting

In this work, we faced a lot of obstacles, which we will describe here.

A. Calibrating the GPS module

GPS compass is a very sensitive module, so it must be calibrated in many situations: after some flights, moving it over a long distance, or during magnetic storms. Furthermore, the position of the compass at the flight controller should also be taken into account and calibrated in iNav or several similar configurators.



(a) Real trajectory.

(b) Approximated by our algorithm trajectory.

Fig. 6: Fourth test (real trajectory).

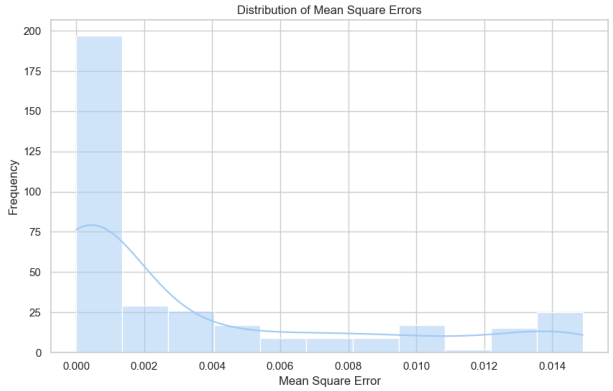


Fig. 7: Mean square error graph

B. Creating custom 3D models for specific needs

Another significant challenge we faced was creating customized 3D models tailored to the specific requirements of our project.

C. Hardware stability

During flights, it is important to take into account several characteristics of the module. For example, the video transmitter can burn if the voltage is very high and the flight takes a short period, as it will get a lot of energy.

D. Appropriate testing

Testing is important for such an algorithm, as real-world usage may be expensive. The problem with testing is that we need datasets with video frames and GPS coordinates on every frame to track the difference between the real and approximated ones.

XI. Summary

In summary, this project develops a method for autonomous localization of unmanned aerial vehicles in poor communication conditions using visual odometry methods and considering the hardware's properties and capabilities.

The requirements for the operation of the entire system were set, and the hardware was selected in accordance with them: Raspberry Pi 4, GPS module, and flight controller Speedybee f405 v3.

An algorithm has been developed that takes into account all the necessary aspects of the drone's operation and allows for predicting real coordinates in the absence of communication.

The developed solution was tested after setting the requirements for the test implementation and conducting them in real conditions. A simulation testing method was also created using PX4 Autopilot with Gazebo-classic.

The tests have shown that the developed solution fulfills the requirements and allows the estimate of the location of the device in the absence of GPS communication.

The code of the project is available at https://github.com/Oleksandr0605/VO_localization_and_mapping

References

[1] Mohammad O. A. Aqel, Mohammad H. Marhaban, M. Iqbal Saripan, and Napsiah Bt. Ismail. Review of visual odometry: types, approaches, challenges, and applications. https://link.springer.com/article/10.1186/s40064-016-3573-7, 2016.

[2] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. https://rpg.ifi.uzh.ch/docs/TRO17_Forster-SVO.pdf, 2021.

[3] Mordor Intelligence. Drones market size and share analysis - growth trends and forecasts (2024 - 2029). https://www.mordorintelligence.com/industry-reports/drones-market, 2023.

[4] Charles F. F. Karney. Geographiclib 2.3. https://geographiclib.sourceforge.io/C++/doc/index.html, 2023.

[5] MAVSDK Development Team. Mavsdk (main). https://mavsdk.mavlink.io/main/en/index.html, 2023.

[6] PX4 team. Gazebo classic simulation. https://docs.px4.io/main/en/sim_gazebo_classic/#video-streaming, 2023.

[7] PX4 team. Px4 autopilot user guide. https://docs.px4.io/main/en/index.html, 2023.

[8] Falin Wu, Kefei Zhang, and Gang-Jun Liu. A study of gps/galileo performance in urban environment using a simulation tool. https://web.archive.org/web/20070618114230id_/http://user.gs.rmit.edu.au/falin/english/publications/proceedings/Wu-etal2006Geo.pdf, 2006.