Object Geolocalization Using Consumer-Grade Devices: Algorithms and Applications

Harald Lampesberger*, Eckehard Hermann and Markus Zeilinger

Department Secure Information Systems, University of Applied Sciences Upper Austria

Abstract

In this article, we present advancements in object geolocalization, originally intended for humanitarian aid, specifically for the localization of reminiscences of war, but also applicable in ecology, e.g., estimating geolocations of observed wildlife. The research utilizes consumer-grade devices such as smartphones and drones, aiming to localize objects geometrically without laser range finders. The main contributions are three geolocalization algorithms operating in a global coordinate system – Ray Marching, Ray Intersection, and Gradient Descent – each suited for different scenarios depending on the availability and nature of measurement data. We also discuss the limitations and sources of error, such as GNSS accuracy and magnetic disturbances, and suggest a method for error correction. The implemented software prototypes were tested using the Parrot ANAFI drone and an Apple iPhone 14 smartphone, with future work focusing on improving error correction and incorporating object detectors for enhanced real-time geolocalization capabilities.

Keywords

UAV, drone, smartphone, geolocalization, algorithms, prototype

1. Introduction

Object geolocalization is crucial in various scenarios such as humanitarian aid and the support for civilian and military organizations in identifying and geolocalizing landmines, unexploded ordnances (UXOs), and unexploded bombs (UXBs). These are ecologically problematic as areas contaminated with those reminiscences of war pose a significant risk for people and animals, making agricultural production in these areas hazardous. As described in previous work [1], our efforts are focused on developing tools for identifying and locating suspicious objects.

So far, we resorted to variants of YOLO [2] neural network-based object detection, e.g., TPH-YOLOv5 [3], for identifying suspicious objects in sensor data. We have accumulated experience industrially, institutionally, and militarily, using 2D camera and 3D LIDAR systems, in building and labeling our own 2D and 3D LIDAR datasets. This paper focuses, however, on geolocalizing objects that are detected by aforementioned techniques. We have already outlined our initial considerations for this in previous work [1]: Determining the geolocation of a suspicious object should be possible from a safe distance, using a consumer-grade device, and a geolocation should be provided in three dimensions (latitude, longitude, altitude) to include placement at the ground, in the air, in or on buildings, or on trees. Our goal is to enable consumer-grade devices, like smartphones or drones, in localizing objects through optical and geometrical means alone, for instance without the use of special sensors and laser rangefinders. In addition, we assume a changing environment, e.g., caused by an earthquake, heavy flood, or large-scale bombing campaign, which hinders visual geolocalization techniques based on prerecorded image datasets. In this paper, we lay the groundwork for the first supported devices in this endeavor based on the Parrot ANAFI drone and recent Apple iPhones.

Since the publication of [1] there have been several enquiries regarding the use of our software prototypes beyond acquiring locations of mines and UXOs/UXBs. Given that the implementation supports real-time geolocalization of objects in a video feed, potential applications also include counting livestock and wildlife. Another envisioned use case is to track the flight route and speed of bird flocks

4th International Workshop on Camera Traps, AI, and Ecology, September 05–06, 2024, Hagenberg, Austria

*Corresponding author.

harald.lampesberger@fh-hagenberg.at (H. Lampesberger)

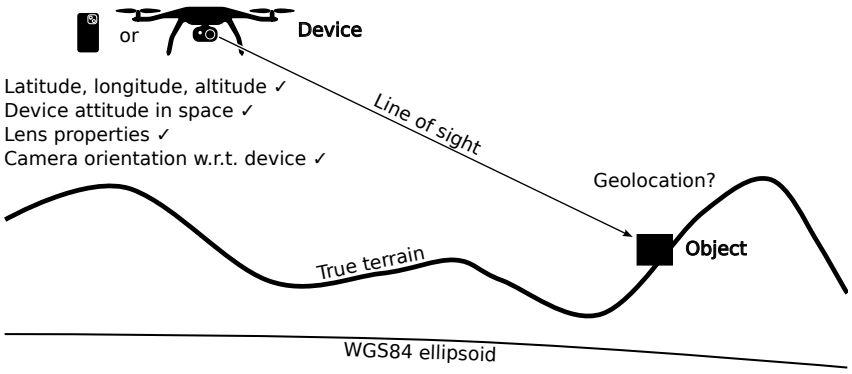


Figure 1: Problem setting considered in this paper

for preventing collisions in offshore wind parks.

Until now, the dual-use character resulting from these different application areas is the reason why the software has only been made public in demos, presentations, and videos but not yet published as such. This paper summarizes the technical background of our work in geolocalization so far.

1.1. Problem Definition

Figure 1 illustrates the problem setting addressed in this paper. Our considered devices possess onboard sensors to estimate the position and attitude in space. Geographic positions are commonly acquired from global navigation satellite systems (GNSS), e.g., expressed as latitude and longitude, relative to a *global reference frame*. This study adopts the *WGS84 reference frame* of the Global Positioning System (GPS) because of its widespread use and the availability in our considered devices. Regarding altitude, options include height above ground level, mean sea level, or height above ellipsoid (HAE). In our research, we focus on HAE for calculations to stay consistent with WGS84.

Attitude signifies the device's orientation within a *local reference frame*. This is typically determined by an inertial measurement unit (IMU) using magnetometers, accelerometers, and Earth's gravitational data. The north-east-down (NED) local reference frame is widely used in aerospace applications and also employed in this work because of the drone context [4]. Additionally, the intrinsic camera parameters are well known, encompassing optical lens attributes and the orientation of the camera axis to the device body in the NED frame.

The *line of sight* to an object within the device's camera field of view can therefore be fully characterized using a position vector, originating at the global device location (origin) and extending in the specified direction. Consequently, computing the geolocation of an object boils down to estimating the magnitude of this position vector, which is the distance between the device and the target object. Ideally, this should be possible in real-time for practical use in the field. Various concepts for geolocalization are presented and discussed in literature, however, no public prototypes suitable for our intended use cases were available to us for verification. Commercial products, such as DJI PinPoint [5] and SmartCam3D [6], typically incorporate an active range finding apparatus (i.e., laser range finder) to measure this distance. Laser range finders are typically found in higher-priced drone models, and due to a limited financial budget, these commercial solutions were not considered for evaluation in this paper.

1.2. Research Questions

To address the problem of geolocalization, we state the following research questions:

- How can geolocalization of an object within the field of view be implemented without resorting to a laser range finder?
- What are the potential sources of error and what magnitude of error can be expected by such an approach?

1.3. Methodology

Utilizing a design science methodology [7], we address the posed questions as follows. The analysis of related work in Section 2 reveals that existing solutions for the problem can be categorized into two main approaches: 1.) geometric methods aligning with the problem description as illustrated in Figure 1 and 2.) machine learning-based techniques that match camera perspectives to a georeferenced image dataset. Considering that our motivation stems from humanitarian demining, potentially in conflict zones, where environmental changes may render prerecorded image datasets obsolete (due to factors such as vegetation growth), the machine learning approach becomes less applicable. Consequently, this paper focuses on the geometric approach that can complement existing object detectors based on neural networks such as TPH-YOLOv5 [3].

The reference frames and coordinate systems, necessary for the geolocalization algorithms, are in line of the works of Johnston [8] and Koks [4] and presented in Section 3. For the design of our prototypes, we consider three use cases depending on the number of different positions and perspectives the camera device (i.e., drone or smartphone) needs to take. All cases require a direct line of sight from the camera to the object in question. The main contribution of this paper are three algorithms that have been implemented in software and enable real-time geolocalization. The algorithms are explained in detail in Section 4, and Section 5 summarizes prototype implementation details for the Parrot ANAFI drone and Apple iPhone smartphone. The distance between true location and estimated location is the fundamental metric for error in the proposed geolocalization setting. To evaluate our prototypes, Section 6 discusses preliminary results and investigates the sources and magnitudes of error in the respective implementations.

2. Related Work

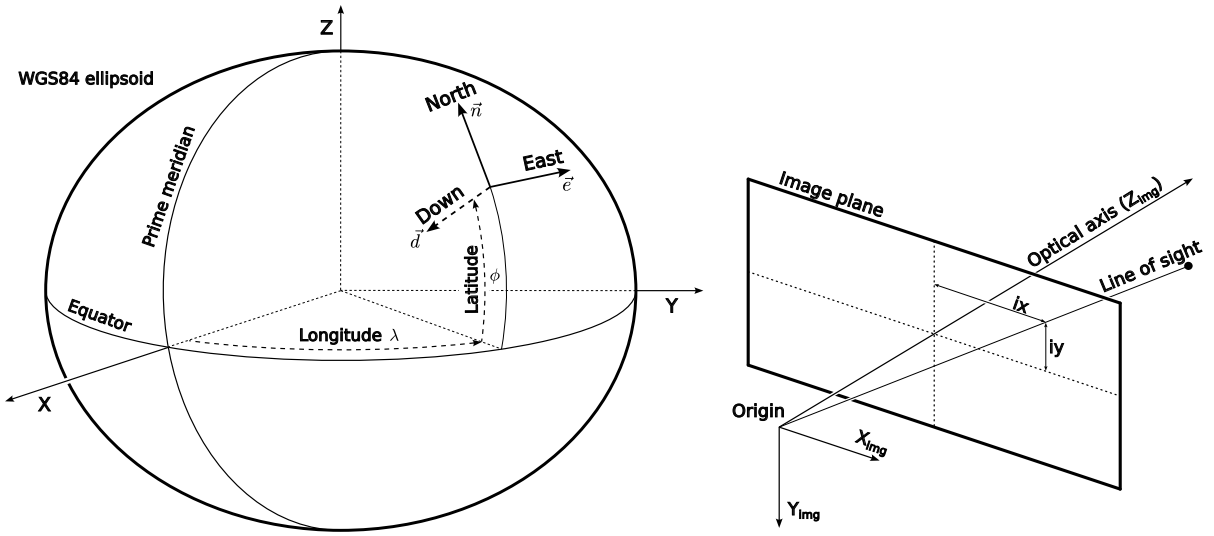
In recent years, several projects from varying application domains that combine machine learning object detectors with geographic localization have been published. These include cattle counting [9], vehicle tracking [10], and surveillance for situational awareness [11].

The foundations for estimating an object's geographic location in the field of view of an unmanned aerial vehicle (UAV) were established by Johnston [8]. A local line-of-sight vector is transformed into a global Cartesian reference frame using rotations, facilitating further computations. The author argues that assuming the ground levels beneath the UAV and the target object are identical simplifies computation but only holds true on relatively flat terrain. Otherwise, estimation error occurs in uneven landscapes [10, 11]. Johnston [8] furthermore emphasizes that a digital elevation model (DEM) can significantly reduce the estimation error when the object is on ground level. This concept underpins in the ray marching algorithm presented in Section 4.

For high precision, errors in UAV sensors must be minimized as they directly impact geolocalization accuracy. Therefore, filtering of UAV sensor data, as studied by Barber et al. [12], is essential. Our prototypes target the Parrot ANAFI drone and a recent Apple iPhone, which already incorporate state-of-the-art filtering techniques [13, 14]; this aspect was not further explored in our work.

Visual object geolocalization represents a different approach to solving localization problems. According to Wilson et al. [15], identifying an object's location can be reformulated as either a regression or classification problem, addressable by machine learning and deep learning models, requiring a dataset of georeferenced images. Due to this data dependency, visual geolocalization is not considered in this study.

Visual Simultaneous Localization and Mapping (SLAM) and Structure from Motion (SfM) are techniques for three-dimensional environment estimation, particularly relevant in robotics and augmented reality applications. For further insights into these methods, we refer to the survey by Saputra, Markham, and Trigoni [16]. Although geolocalization is not directly mentioned, its connection is obvious; the ray intersection and gradient descent algorithms in Section 4 for estimating geopositions of objects, not necessarily on ground level, are inspired by *triangulation* in motion segmentation [16].



(a) Local NED reference frame (reproduced from [4])

(b) The pinhole camera model and its axes after transformation in the NED frame

Figure 2: The core models used in this work

3. Reference Frames and Coordinate Systems

Regarding reference frames, the presented work builds upon the results published by Koks [4]. We assume that our considered devices, e.g., drones, have GNSS capabilities to sense the global position in terms of latitude (ϕ), longitude (λ), and HAE altitude (h) in the *global reference frame* WGS84 and we expect a geolocalization algorithm to return its findings in the same format. Earth is modelled by a reference ellipsoid in WGS84, as shown in Figure 2a. Computations prefer a metric space, and we therefore resort to the *Earth-centered, Earth-fixed* (ECEF) coordinate system. The origin is situated in the center of the reference ellipsoid, Z extends the rotational axis, the X axis passes through the equator and prime meridian ($\lambda = \phi = 0$), and Y is orthogonal to X and Z. The algorithms for transformation between a geodetic point (λ, ϕ, h) in WGS84 and a geocentric point (x, y, z) are well established and sufficiently accurate [4]. Points near Earth's surface entail large numbers in ECEF, and we therefore resort to the constants and optimizations developed by Osen [17] to overcome accuracy issues with floating-point numbers.

Note that if we express *north* in the ECEF coordinate system, the direction of the vector is different for every point on Earth. Attitude sensors in our considered devices measure the orientation relative to a *local reference frame*, NED in our case, also including the line-of-sight vectors. The geolocalization algorithms in Section 4 require a transformation of local vectors into the global coordinate system, and according to Koks [4], this can be achieved by constructing a NED basis in the ECEF coordinate system. A NED basis $\{\vec{n}, \vec{e}, \vec{d}\}$ is defined by three ECEF vectors, representing north, east, and down, as shown in Figure 2a. Intuitively, vectors $\vec{n}$ and $\vec{e}$ span a plane, tangential to the ellipsoid, at a particular latitude and longitude. Suppose, a device is at a particular λ_{dev} and ϕ_{dev}. First, the NED basis $\{\vec{n}, \vec{e}, \vec{d}\}$ is defined at $\lambda = \phi = 0$ by setting $\vec{n} = [0, 0, 1]^T$ parallel to the Z axis and $\vec{e} = [0, 1, 0]^T$ parallel to the Y axis. Vector $\vec{e}$ is then rotated around $\vec{n}$ by λ_{dev} degrees, $\vec{n}$ is rotated around then new $\vec{e}$ by $-\phi_{dev}$ degrees, and $\vec{d}$ is finally the cross product of rotated vectors $\vec{n} \times \vec{e}$. This basis is different for every point on Earth, but due to Earth's size, the changes of the basis are negligible within the flying range of a commercial drone.

We also need a model for the camera. In our work, a simple *pinhole camera model*, depicted in Figure 2b, represents the viewport of a device and has a local camera reference frame, where the optical axis is Z_{img}, axis X_{img} is along the width of the viewport, and Y_{img} is in downward direction of the viewport height. By directing Y_{img} downward, the reference frame stays compatible with NED in later steps. The considered intrinsic lens parameters such as horizontal and vertical field of view (*hfov*, *vfov*)

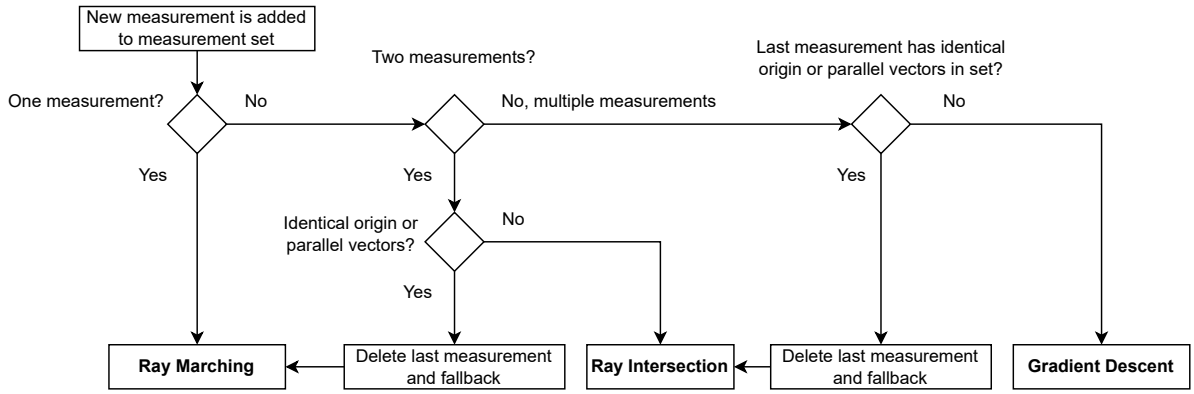


Figure 3: Algorithm selection process for a set of measurements

are known, and a virtual projection plane spans in front of the origin representing the current image. There are two use cases for a camera model: A geographic point in the world within the camera's field of view is mapped to a point/pixel in the image for rendering; and selecting a point/pixel at image coordinates (ix, iy) is considered a geolocalization *measurement* and translated to a line-of-sight unit vector based on *hfov* and *vfov*.

Finally, the relation between optical axis and local reference frame needs to be established. The orientation of a device in the local reference frame does not necessarily reflect the direction of the optical axis. While the optical axis of a smartphone camera is static with respect to the device body, the drone camera of our Parrot ANAFI is mounted on a gimbal that smooths out movements and enables view directions independent from the drone body orientation. From device attitude and gimbal sensor data, the combined direction of device and optical axis can be described as a single rotation from north $\vec{n}$ to the camera axis Z_{img}. Suppose we have performed a measurement and selected a point in the viewport's image; the corresponding line-of-sight unit vector in the local camera frame is then transformed into the local NED reference frame by aforementioned rotation and subsequently transformed by change of basis into the global ECEF coordinate system. On the other hand, a known point in ECEF is mapped to a point/pixel in the viewport by reversing this process.

4. Proposed Geolocalization Algorithms

In the previous section, a measurement of an object was already introduced by selecting a point/pixel in the viewport. More precisely, a measurement is a position vector $(p, \vec{v})$, where p is the origin (location of the device) and $\vec{v}$ is a unit vector in the direction of an object, both in the ECEF coordinate system. A *measurement set* is then a set of measurements $M = \{(p_1, \vec{v}_1), \ldots, (p_n, \vec{v}_n)\}$, all referring to the same object from different device positions.

An initial measurement creates a new measurement set, and the number of measurements in M dictates the choice of a suitable geolocalization algorithm. The selection process is illustrated in Figure 3. A single measurement is only suitable for ground objects and necessitates a digital elevation model of the terrain. Adding additional measurements is only considered when the origin is unique and $\vec{v}$ is not parallel to existing measurements. There are two cases when parallel measurements can occur: 1.) the same measurement is recorded twice by accident or 2.) the unlikely case, where sensor disturbances cause parallel vectors despite different origins. For two measurements, the geolocation can be computed by solving a linear equation system of two skew rays. More than two measurements overdetermines this system and we resort to gradient descent. The iterative refinement of a solution using gradient descent simplifies the computational process in our application scenario significantly because a measurement set is considered dynamic, where new measurements are added and old measurements can expire over time. Instead of repeatedly solving a closed form when the measurement set changes, gradient descent only needs a few iterations to update the previous solution.

4.1. One Measurement: Ray Marching

Ray marching extends the line-of-sight unit vector until it intersects with an elevation model of the terrain. A digital elevation model (DEM) is typically generated from airborne laser scanning and captures terrain altitude data. Thanks to services like COPERNICUS [18], this data is accessible for research. The data resolution in our experiments was 10m and 25m for the Austrian region. A DEM is typically provided as GeoTIFF [19] and can be queried with tools such as GDAL [20]; for a particular latitude and longitude, an interpolated altitude is returned. Algorithm 1 sketches the loop that extends the unit vector until the altitude of a computed point matches the DEM. The algorithm requires bounds for the acceptable error and maximum distance. The unit vector is extended by a static step size as long as the computed point stays above the DEM surface. When the computed point is beneath the DEM surface, the step is not taken and the step size is halved for the next iteration to achieve a similar effect as in binary search algorithms. Setting the step size too large could lead to false results when the ray, e.g., passes through a hill in a single step and continues marching. As a heuristic, choosing a step size smaller than the DEM resolution is a good starting point, e.g., 24 for 25m resolution. This algorithm was reliable in experiments but has limitations: the object needs to be on ground level, up-to-date DEM data must be available, and localization error increases in shallow perspectives, i.e., when height difference between device and object is small. The increased error sensitivity due to shallow perspectives has already been described by Barber et al. [12].

Algorithm 1: Ray marching algorithm

Data: Measurement $(p, \vec{v})$, where p is above ground level and $\vec{v}$ is a unit vector; GDAL interface to a DEM; accuracy target and maximum distance
Result: ECEF Geolocation q and distance d
$d \leftarrow 0$ and $stepsize \leftarrow 24$;
do
  $\vec{u} \leftarrow \vec{v} \cdot d$;
  $q \leftarrow p + \vec{u}$;
  h is gathered from transforming q into WGS84 reference frame;
  h_{GDAL} is gathered from querying the DEM with current estimated point q;
  $err \leftarrow h - h_{GDAL}$;
  if $err > 0$ *(computed point is still above terrain)* **then**
    $d \leftarrow d + stepsize$;
  else
    $stepsize \leftarrow stepsize/2$;
  end
while $abs(err) >$ *accuracy target and* $|\vec{u}| <$ *maximum distance*;

4.2. Two Measurements: Ray Intersection

For cases when an object is not on ground level, the geolocation can be estimated by two measurements from different positions, i.e., triangulation, a well-known concept in geometry [16]. A challenge in our setting in three-dimensional space is that two rays are extremely unlikely to intersect; two rays are typically skew to each other. Intuitively, the point in question would be in the middle, where both rays are nearest. More formally, we solve for a line, orthogonal to both rays, and we take the middle of this line as geolocalization estimate. Suppose, we have two measurements $\{(p_1, \vec{v}_1), (p_2, \vec{v}_2)\}$ in the ECEF coordinate system; we compute $\vec{u} = \vec{v}_1 \times \vec{v}_2$ orthogonal to both rays. By introducing scale factors α, β, and γ, the following linear equation holds:

$$p_1 + \alpha \cdot \vec{v}_1 + \beta \cdot \vec{u} = p_2 + \gamma \cdot \vec{v}_2.$$

By rephrasing the equation such that the three unknown scale factors are on the left side, we end up with a linear equation system that can be solved by standard Gauss-Jordan Elimination:

$$\alpha \cdot \vec{v}_1 + \beta \cdot \vec{u} - \gamma \cdot \vec{v}_2 = p_2 - p_1.$$

The ray intersection algorithm is straight forward by solving the equation for the three unknowns and returning an estimated ECEF geoposition q:

$$q = p_1 + \alpha \cdot \vec{v}_1 + \frac{\beta \cdot \vec{u}}{2}.$$

This method can be applied when objects are not on ground level, e.g., on a rooftop or in the air, but it requires two measurements and movement between two positions, ideally such that the rays are orthogonal to each other.

4.3. Three or more Measurements: Gradient Descent

Suppose a drone overpasses terrain and an object detector creates a stream of measurements of the same object seen from different origins at different moments of time. As in the previous case, we assume that the rays are skew to each other. The previous algorithm is not applicable in this case because the equation system would be overdetermined by more than two measurements. By building upon the results of Traa [21] we can rephrase geolocalization as a minimization problem for a set of measurements, numerically solvable by gradient descent. First, we define the squared orthogonal distance d between a position vector $(p, \vec{v})$ and a point q in accordance to Traa [21, Eq. 8]:

$$d(p, \vec{v}; q) = (p - q)^T \cdot (\mathbf{I} - \vec{v} \cdot \vec{v}^T) \cdot (p - q).$$

Suppose that $M = \{(p_1, \vec{v}_1), \ldots, (p_n, \vec{v}_n)\}$ is a set of measurements. In line of Traa [21, Eq. 10], the sum of squared distances, or $loss$ in machine-learning terms, of point q is then:

$$loss(M; q) = \sum_{(p_i, \vec{v}_i) \in M} d(p_i, \vec{v}_i; q) = \sum_{(p_i, \vec{v}_i) \in M} (p_i - q)^T \cdot (\mathbf{I} - \vec{v_i} \cdot \vec{v_i}^T) \cdot (p_i - q).$$

Finding the best estimate for point q then becomes a minimization problem $\hat{q} = \operatorname{argmin}_q loss(M, q)$. This can be computed by gradient descent when the derivative of $loss$ with respect to q is known. The partial derivative definition is in accordance to Traa [21, Eq. 12]:

$$\frac{\partial loss}{\partial q} = \sum_{(p_i, \vec{v}_i) \in M} -2 \cdot (\mathbf{I} - \vec{v_i} \cdot \vec{v_i}^T) \cdot (p_i - q).$$

Algorithm 2 sketches the optimization loop that continues until the estimated point $\hat{q}$ stabilizes. The learning rate can be configured and 0.065 was found to converge sufficiently fast in real time given the magnitudes of values in the ECEF coordinate system. The inner for loop can be significantly optimized by precomputing the 3x3 matrix $(\mathbf{I} - \vec{v_i} \cdot \vec{v_i}^T)$ for every measurement in M.

To summarize, the gradient descent algorithm is applicable when multiple measurements of the same object but from different origins are available, e.g., from an object detector during device movement. Geolocalization is not restricted to ground level, however, the object must not move while measurements are collected from a single device.

5. Implementations

The concepts and ideas in this work have been implemented in one library and two applications.

Algorithm 2: Gradient descent algorithm

Data: Measurement set $M = \{(p_1, \vec{v}_1), \ldots, (p_n, \vec{v}_n)\}$
Result: ECEF Geoposition $\hat{q}$
Learning rate $\eta \leftarrow 0.065$;
Current and previous estimate $\hat{q} \leftarrow \hat{q}_{prev} \leftarrow [0, 0, 0]^T$;
do
  Partial derivative $dp \leftarrow [0, 0, 0]^T$;
  for $(p_i, \vec{v}_i) \in M$ **do**
    $dp \leftarrow dp - 2 \cdot (\mathbf{I} - \vec{v}_i \cdot \vec{v}_i^T) \cdot (p_i - q)$;
  end
  $\hat{q} \leftarrow \hat{q} - \eta \cdot dp$;
  $v \leftarrow \hat{q} - \hat{q}_{prev}$;
  $\hat{q}_{prev} \leftarrow \hat{q}$;
while *Estimated point still changes sufficiently* $|v| > 10^{-7}$;

5.1. libgeo2pixel

The reference frames, coordinate systems, and geolocalization algorithms are implemented in the *libgeo2pixel* library. For continuously updating the state of the world using device telemetry data, a function is provided. For simplifying computations, the NED basis for translating a local direction into the global ECEF frame is only recomputed when the device has moved more than 2 kilometers from its initial location. The library is written in the C language with portability and speed as core goals. The code has been optimized, e.g., rotations are solely expressed in quaternion operations, so it can be integrated in real-time applications. Furthermore, the library is dependency free with two exceptions: the C standard library and, for accessing DEM information, an interface for the *gdalinfo* [20] tool. Everything else is self contained for portability. Consequently, in our implementations,

5.2. PDrAW for Parrot ANAFI (video-analyzer)

The Parrot ANAFI drone was chosen for experiments because of its software design and open-source policy. Telemetry data is already processed by a Kalman filter onboard and embedded as metadata track in recorded video streams and live video feeds [22]. Parrot developers also provide an open-source command line tool for video viewing named PDrAW [23] that seamlessly integrates the live video stream on a computer through the drone controller. For integrating libgeo2pixel, we created a fork of PDrAW, named *video-analyzer*, with a reduced feature set but enriched user interface, i.e., mouse clicks for taking or deleting measurements and augmented reality overlays. For example, DEM data can be visualized as grid overlay in the live view as shown in Figure 4. The grid overlay has a practical purpose in error correction. Early on, experiments had shown that magnetic disturbances are the most significant error source in the drone model by introducing an angular deviation between the sensed north axis and the local NED reference frame. The video-analyzer app integrates a simple form of error correction by allowing manual alignment between DEM and true terrain in terms of an error-compensating quaternion that is considered in libgeo2pixel computations.

Measurements in video-analyzer are always annotated by a number representing the line-of-sight distance in meters to the respective point, similar to a range finder. This can be seen in the screenshots in Figures 8a and 8b. Video-analyzer also enables automatic import and export of geographic points via CSV files. Figure 8c is a screenshot of the mapping software QGIS [24] that reads the exported measurements and displays them on a map. Our fork of PDrAW is implemented in C/C++ with a custom cmake build system, and it can be compiled on Linux, MacOS, and Windows WSL.

Furthermore, video-analyzer has been ported to the Nvidia Jetson Nano platform that runs an embedded version of Linux. This platform is energy efficient and has artificial intelligence capabilities that we used for video decoding. Integrating and accelerating object detectors on this platform is

Figure 4: Live view of the digital elevation grid overlay in the video-analyzer app

Figure 5: The video-analyzer app was ported to an Nvidia Jetson Nano platform for real-time field tests

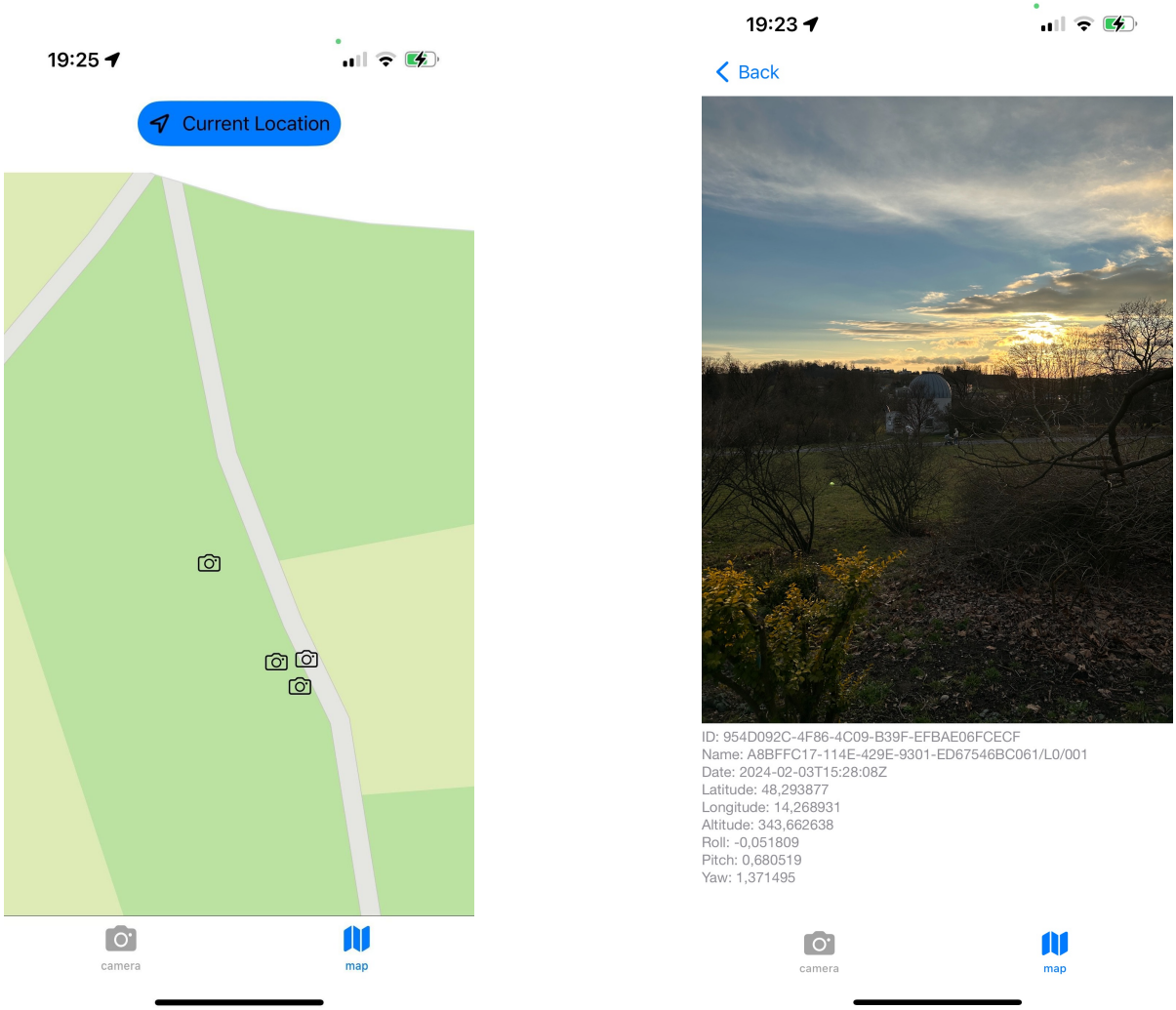
planned for future work. Figure 5 shows a prototypical setup using the battery-powered Jetson platform in field use, where the live feed of the drone, including its telemetry data, is captured and visualized by video-analyzer instead of the smartphone controller app.

5.3. Apple iPhone 14 (GeoMatApp)

The proof-of-concept implementation of a smartphone app was carried out with the goal of giving a wide audience the ability to capture, identify, and geolocalize objects from a safe distance. We see potential use case scenarios in natural disaster management, in the search for missing persons, and for farmers working in areas, potentially contaminated by reminiscences of war, like in former Yugoslavia or currently in Eastern Ukraine. The app itself was implemented in Swift 5.10, compiled for iOS 17, and tested on an iPhone 14 Plus.

The app consists mainly of three views: a *CameraView* for taking pictures and a *MapView*, as shown in Figure 6a, for displaying the geolocations of images origins on a map using camera icons. By selecting a camera icon on the map, an *ImageView*, shown in Figure 6b, is opened that displays the image taken at that location along with the data captured during photography, which are primarily GPS data as well as roll, pitch, and yaw of the device. Libgeo2pixel was integrated as an external service.

Unlike drones, the acquisition of sensor data does not happen automatically in one shot along with



(a) The camera icon indicates the location where a picture was taken

(b) The captured image with the device Euler angles and the GNSS data when the image was taken

Figure 6: MapView and ImageView of the GeoMatApp

the query of image sensor data, i.e., there is no global shutter on all sensor readings. Initially, this led to significant inaccuracies when calculating the geolocation of objects identified in the image later on. The cause of this was that even small movements of the camera in the time window between the query of the image sensor, GPS sensor, and camera orientation via DeviceMotionManager resulted in massive inaccuracies in the computed geolocation for distant objects. A solution for achieving acceptable results involved setting the DeviceMotionUpdateInterval of the DeviceMotionManager to 0.01 seconds. Another notable error source was the difference between the iPhone local reference frame and our NED frame. The CMAttitudeReferenceFrame of the DeviceMotionManager needs to be set to xTrueNorthZVertical (the Z axis is vertical and the X axis points to the geographic north pole) [25] and an additional transformation needs to be performed in libgeo2pixel before geolocalization calculations.

Despite successful functional tests and initial experiences, the GeoMatApp was still considered a proof of concept and not yet ready for testing as discussed in the next section.

6. Preliminary Evaluation and Discussion

The video-analyzer prototype was evaluated in the years 2022–2023 by conducting drone flights with two different Parrot ANAFI drones in the greater area of Upper Austria and Tirol. Figures 7 and 8 show

(a) Manual clicks on horse positions in an overflight (b) Corresponding QGIS live view on Google Maps

Figure 7: Ray marching requires a top-down view for sufficient accuracy

(a) First measurement (ray marching) (b) Second measurement (ray int.) (c) Real-time export on Google Maps

Figure 8: Ray intersection of a reference object during overflight (from [1])

screenshots for demonstrating functional tests of the video-analyzer app. Figure 7a is a video still of an overflight recording, where each horse was manually clicked once for estimating the geolocation by ray marching. The corresponding live view in QGIS in Figure 7b was updated accordingly in real time. Figures 8a and 8b are video stills from a ray intersection measurement by two clicks on a small reference object. Figure 8c is the according QGIS live view.

As mentioned in our previous work [1], we randomly chose landmarks that are identifiable on maps for geolocalization and estimated the error in meters between computed and true location. As ground truth we used the official maps provided by DORIS [26]. Our findings can be summarized as follows:

- We distinguish between translational and rotational errors. Translational errors are the consequence of dilated precision in GNSS. The estimated geolocation of an object inherits the horizontal and vertical dilution of precision of the drone or smartphone independent from the distance to the object. The state-of-the-art approach to increase GNSS precision is real-time kinematics (RTK) which is not available for our devices.
- Rotational errors with respect to the local NED reference frame originate from deviations in IMU and gimbal sensor readings. As already recognized by Nuske et al. [27], the biggest source of error is deviation in magnetic heading. We observed errors up to 10 degrees with respect to true north. The estimation error unfortunately grows linearly with increasing line-of-sight distance between device and object when a rotational error is present. Switching to a different drone model that utilizes a differential GNSS with multiple antennas for sensing the attitude would reduce rotational errors.
- We integrated a simple manual error correction method by aligning the DEM grid overlay with the visible terrain. An error-correcting quaternion captures the manual adjustments. Without

error correction, the geolocalization error was up to 10% with respect to the line-of-sight distance, e.g., up to 1m at 10m distance. Aligning the DEM manually reduced the error to 1-2% with respect to the line-of-sight distance.

- For drones, ray marching was surprisingly accurate in open terrain, but unreliable next to or in forests. Analyzing the DEM data highlighted that laser airborne scanning in principle aims for capturing the ground level, however, dense trees can introduce height deviations that directly translate to bad estimations. The worst observed case in experiments was at a low flight altitude next to a forest such that the DEM elevation was greater than the drone altitude. Ray marching failed in this case.

7. Conclusions

To answer our research questions, this paper shows that under certain conditions it is possible to geolocalize an object within the field of view of a camera given that GNSS and attitude sensors are sufficiently accurate and intrinsic camera parameters are accounted for. We have conceptualized three geolocalization algorithms and implemented them in a library and two software prototypes, for the Parrot ANAFI drone and Apple iPhone 14 respectively, to gain insight. A preliminary evaluation has indicated that magnetic heading was the primary source of error in experiments and we therefore integrated a simple error correction method in video-analyzer by manually aligning a DEM grid overlay with the terrain.

There are several aspects that can be improved in future work. Automating the manual error correction would be the top priority. Also, integrating object detectors and object trackers for automating the gradient descent algorithm for geolocalization during flight would be a further step. We currently investigate camera calibration methods because the intrinsic model in our prototypes only considers *hfov* and *vfov* in the simplified pinhole model. So far, this was not an issue because our drone and smartphone perform various image enhancement procedures onboard but for increased accuracy the optical lens properties must be considered. Another research direction that emerged from the gradient descent algorithm is to synchronize distributed cameras, so a measurement becomes instant. An application that was already discussed in this context is to track the flight route and altitude of bird flocks near offshore wind parks.

Releasing the presented tools under open-source license is not planned yet, but currently under discussion. Given the roots of the project in humanitarian and military demining, the dual-use character is non-repudiable and more research is needed before making the source code accessible.

References

[1] H. Lampesberger, E. Hermann, Real-time geo-localization of unexploded ordnance, in: 19th International Symposium Mine Action 2023, Vodice, Croatia, 2023, pp. 36–39.

[2] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, https://pjreddie.com/darknet/yolo/, 2018. Last access: 2024-07-25.

[3] X. Zhu, S. Lyu, X. Wang, Q. Zhao, TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Montreal, BC, Canada, 2021, pp. 2778–2788. doi:10.1109/ICCVW54120.2021.00312.

[4] D. Koks, Using Rotations to Build Aerospace Coordinate Systems, Technical Report DSTO–TN–0640, Electronic Warfare and Radar Division Systems Sciences Laboratory, 2008. URL: https://apps.dtic.mil/sti/pdfs/ADA484864.pdf, last access: 2024-07-21.

[5] DJI, Zenmuse H20 Series, https://www.dji.com/at/zenmuse-h20-series, 2024. Last access: 2024-07-21.

[6] Rapid Imaging, SmartCam3D, https://www.rapidimagingtech.com/smartcam3d/, 2024. Last access: 2024-07-21.

[7] A. R. Hevner, S. T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (2004) 75—-105.
[8] M. G. Johnston, Ground object geo-location using UAV video camera, in: 2006 IEEE/AIAA 25TH Digital Avionics Systems Conference, Portland, OR, USA, 2006, pp. 1–7. doi:10.1109/DASC.2006.313770.
[9] V. Soares, M. Ponti, R. Gonçalves, R. Campello, Cattle counting in the wild with geolocated aerial images in large pasture areas, Computers and Electronics in Agriculture 189 (2021) 106354. doi:10.1016/j.compag.2021.106354.
[10] X. Zhao, F. Pu, Z. Wang, H. Chen, Z. Xu, Detection, tracking, and geolocation of moving vehicle from UAV using monocular camera, IEEE Access 7 (2019) 101160–101170. doi:10.1109/ACCESS.2019.2929760.
[11] R. Geraldes, A. Gonçalves, T. Lai, M. Villerabel, W. Deng, A. Salta, K. Nakayama, Y. Matsuo, H. Prendinger, UAV-based situational awareness system using deep learning, IEEE Access 7 (2019) 122583–122594. doi:10.1109/ACCESS.2019.2938249.
[12] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, C. N. Taylor, Vision-based target geo-location using a fixed-wing miniature air vehicle, Journal of Intelligent and Robotic Systems: Theory and Applications 47 (2006) 361–382. doi:10.1007/s10846-006-9088-7.
[13] Parrot, Parrot ANAFI Technical Specifications, https://www.parrot.com/assets/s3fs-public/2020-07/white-paper_anafi-v1.4-en.pdf, 2020. Last access: 2024-07-25.
[14] Apple Developer, Core Motion, https://developer.apple.com/documentation/coremotion, 2024. Last access: 2024-07-25.
[15] D. Wilson, X. Zhang, W. Sultani, S. Wshah, Visual and object geo-localization: A comprehensive survey, CoRR abs/2112.15202 (2021). arXiv:2112.15202.
[16] M. R. U. Saputra, A. Markham, N. Trigoni, Visual slam and structure from motion in dynamic environments: A survey, ACM Comput. Surv. 51 (2018). doi:10.1145/3177853.
[17] K. Osen, Accurate Conversion of Earth-Fixed Earth-Centered Coordinates to Geodetic Coordinates, Technical Report, Norwegian University of Science and Technology, 2017. URL: https://hal.science/hal-01704943v2, last access: 2024-07-22.
[18] The European Space Agency, Copernicus DEM - Global and European Digital Elevation Model (COP-DEM), https://spacedata.copernicus.eu/collections/copernicus-digital-elevation-model, 2022. Last access: 2024-07-24.
[19] E. Devys, T. Habermann, C. Heazel, R. Lott, E. Rouault, OGC GeoTIFF Standard, https://docs.ogc.org/is/19-008r4/19-008r4.html, 2019. Last access: 2024-07-26.
[20] GDAL, GDAL documentation, https://gdal.org, 2024. Last access: 2024-07-21.
[21] J. Traa, Least-Squares Intersection of Lines, Technical Report, University of Illinois at Urbana-Champaign, 2013. URL: https://silo.tips/download/least-squares-intersection-of-lines, last access: 2024-07-21.
[22] Parrot Developers, libvideo-metadata - Parrot Drones video metadata library, https://github.com/Parrot-Developers/libvideo-metadata, 2024. Last access: 2024-07-21.
[23] Parrot Developers, PDrAW - Parrot Drones Awesome Video Viewer, https://github.com/Parrot-Developers/pdraw, 2024. Last access: 2024-07-21.
[24] QGIS, QGIS documentation, https://www.qgis.org/resources/hub/, 2024. Last access: 2024-07-26.
[25] Apple Developer, xTrueNorthZVertical, https://developer.apple.com/documentation/coremotion/cmattitudereferenceframe/1616019-xtruenorthzvertical, 2024. Last access: 2024-07-26.
[26] Digitales Oberösterreichisches Raum-Informations-System, DORIS, https://www.doris.at/, 2024. Last access: 2024-07-25.
[27] S. Nuske, M. Dille, B. Grocholsky, S. Singh, Representing substantial heading uncertainty for accurate geolocation by small UAVs, in: AIAA Guidance, Navigation, and Control Conference, Toronto, Ontario, Canada, 2010, pp. 1–13. doi:10.2514/6.2010-7886.