

Targets Localization from Drone: AEROPVS03

vsTASKER, target localization from a drone focuses on extracting precise ground coordinates of a point of interest (POI) using a flying quadcopter, a payload-mounted gimbal camera, and AI image analysis. Because vsTASKER is a C++ code-generating simulation engine, it models this entire process dynamically to test autonomous UAV logic before deployment.

The primary challenge it addresses is the absence of a real-time terrain profile (digital elevation model) provided to the drone, relying instead on geometric calculation.

The Target Localization Logic Flow

When the onboard AI detects a target, the drone initiates a multi-step "Acquire" measurement logic to determine its location:

- ✚ Step 1: First Line of Sight – The drone shoots a theoretical ray from its center of gravity through the center of the camera payload based on the camera's absolute azimuth (0° = North) and elevation (0° = Down).
- ✚ Step 2: Elevation Change – The drone flies up vertically (e.g., by 300 meters) while maintaining the camera gimbal locked on the target, then shoots a second ray.
- ✚ Step 3: Lateral Slide – The drone slides horizontally in a random direction (e.g., another 300 meters) while keeping the target centered, shooting a third ray.
- ✚ Step 4: Sliding Rings Algorithm – The system calculates the closest intersection points between these three independent lines of sight.
- ✚ Step 5: Centroid Extraction – The centroid of these closest intersection points is computed to yield the definitive, highly accurate GPS coordinates of the target.

Core Components Modeled in vsTASKER

[Drone GPS/IMU] + [Gimbal Azimuth/Elevation] ➡ Line of Sight Ray ➡ Triangulation ➡ Target Coordinates

- ✚ The Terrain Environment: Built using databases like OpenSceneGraph (OSG), allowing 3D visualization and Line of Sight (LOS) checks during simulation editing.
- ✚ Sensor Telemetry Simulation: The software simulates the exact pitch, roll, and yaw of the drone alongside camera telemetry to calculate the vectors.
- ✚ C++ Code Generation: Once the mission logic is finalized in the vsTASKER graphical interface, it compiles directly into executable C++ code to control real autopilot systems or high-fidelity image generators like VegaPrime.
- ✚ Are you looking to configure this specific Triangulation/Acquire script in vsTASKER, or are you trying to integrate an external AI image analyzer into the simulation loop?

In the vsTASKER simulation software, **Targets Localization from Drone** is a tactical exercise designed to extract precise global coordinates of a ground-based Point of Interest (POI). It models a scenario using an autonomous flying quadcopter drone, a payload-mounted camera, and AI-driven image analysis.

The primary challenge it addresses is **geo-locating a target when the terrain's digital elevation profile is completely unknown** to the drone's onboard systems.

The Triangulation Process

Because the drone only knows its own geodetic position and the absolute angle values of the camera payload (Azimuth: 0° = North, Elevation: 0° = Down), it cannot find a target's position with a single snapshot. The vsTASKER framework utilizes dynamic **three-ray measurement logic** to solve this:

1. **Ray 1 (Initial Sighting):** The drone detects the target and shoots a mathematical ray from its center of gravity based on the camera's azimuth and elevation angles.

2. **Ray 2 (Vertical Shift):** The drone ascends vertically by **300 metres** while keeping the target perfectly centered in the camera feed, then fires a second ray.
 3. **Ray 3 (Lateral Shift):** The drone slides another **300 metres** in a random horizontal direction to gain a different perspective and fires a third ray.
-

The Coordinate Extraction Algorithm

Once the three rays are generated, they will not perfectly intersect in 3D space due to simulated sensor noise and atmospheric errors. vsTASKER resolves this by using the **Sliding Rings Algorithm**:

- ✚ The algorithm analyzes the geometric paths of all three rays simultaneously.
 - ✚ It locates the exact point on each ray that is physically closest to the other two rays.
 - ✚ It computes the **geometric centroid** of these three closest points to output the final, highly accurate 3D coordinates of the target.
-

Terrain and Environment Setup

The simulation relies on an OpenSceneGraph (OSG) database provided by TrianGraphics to build a 3D tactical map.

Mission Control: Operators use the 3D map for real-time surveillance monitoring and mission preparation.

Line of Sight (LOS): The system maps and visualizes blind spots and obstructions in 3D, ensuring that target detection holds valid throughout the flight path.

Would you like to explore how to **implement the custom logic** for this flight pattern within the vsTASKER editor,

The purpose of this Software application is to provide a method to extract Geographic Coordinates of detected targets located on a hilly terrain, from a flying drone.

A quadcopter drone with a stabilized mounted camera is used for the terrain scanning. The payload holding the camera provides relative angles from the frame body, in yaw, pitch and roll. The drone holds a GPS which provides its geodetic position, and an accelerometer for altitude and attitude.

A digital terrain has been used for the proof of concept, just to provide realistic images from the camera. Some targets are set on the scene for each mission.

An image processing model, using AI for recognition, will detect targets on images provided by the drone during the mission. Once a target is suspected, the position on the image is sent back to the drone. At this moment, the coordinate's resoluteness will initiate.

The quadcopter

For this project, the drone dynamic is not an essential component as, in fine, the camera is stabilized. But to make it more realistic, we are using a mathematical model based on automatic controllers.

Dynamic

The body motion is handled using P, PI and PID controllers. Such controllers are used for accelerations along x/y axis, altitude gain, pitch and roll command and stability. Motor thrusts are used for altitude and acceleration. A perturbation factor propagates body vibrations. The quadcopter is stable enough and reacts well to applied forces.

Position control

The position controller overrides the cinematic component and uses a PID controller to drive the body at a given 3D position. Some parameters ensure there

is neither amplification nor resonance and that the drone will smoothly join the position and not oscillate and distort around it.

The drone is equipped with a GPS and gyro to keep the accuracy of its position and altitude on the terrain. This position is converted in latitude and longitude. Errors can be added either as constant or as a linear drift over time.

Trajectory

To make the drone follow a trajectory, we use a specific module which moves the *goto-point* along the trajectory, at a specific speed. The trajectory speed cannot be greater than the drone maximum speed. Also, the module always waits for the drone to be close enough to start moving itself.

Payload Camera

The drone is carrying a camera mounted on a rotating and stabilized payload. The attitude and orientation is controlled by the software. Vibrations can be added, although there are of few uses as the processing is done on fix images and not on the video stream. Errors in angles (azimuth, pitch and roll) of the payload versus the frame body can be added (fix, random or drift).

Terrain

For the purpose of the project, a simple and old OpenSceneGraph database (courtesy of TrianGraphics) is used. vsTASKER can build a map out of this format and display it in 3D. This will be convenient for mission preparation and real time monitoring. The position of the target on the map can be checked in 3D as line of sight is an important part of the detection process.

The digital elevation twin of the terrain is not provided to the drone and will be used on the second part of this project. The first approach is to solve the absence of the terrain profile.

Targets

Three targets has been positioned on the terrain. They are stationery ground vehicles: one helicopter, one Humvee and one missile launcher.

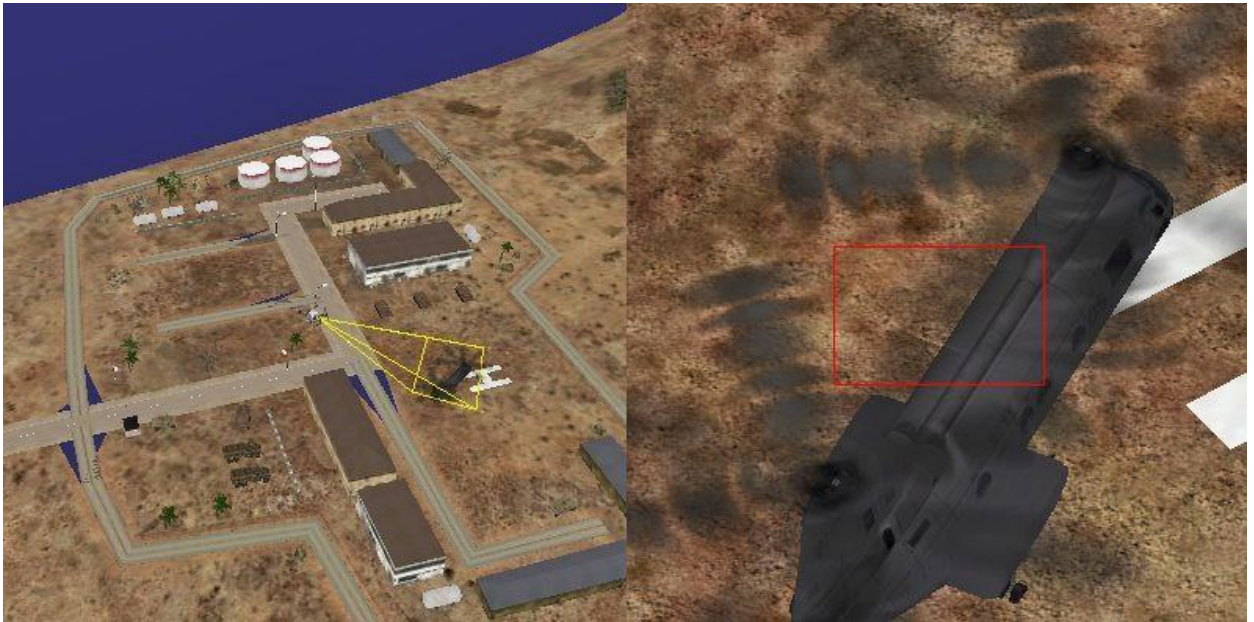
Each of these targets have a visual signature which allows the Image Analyzer (based on machine learning) to recognize them. It may take some time according to the angle of view and the distance. Enough pixels should be provided to trigger the point of interest (Poi).



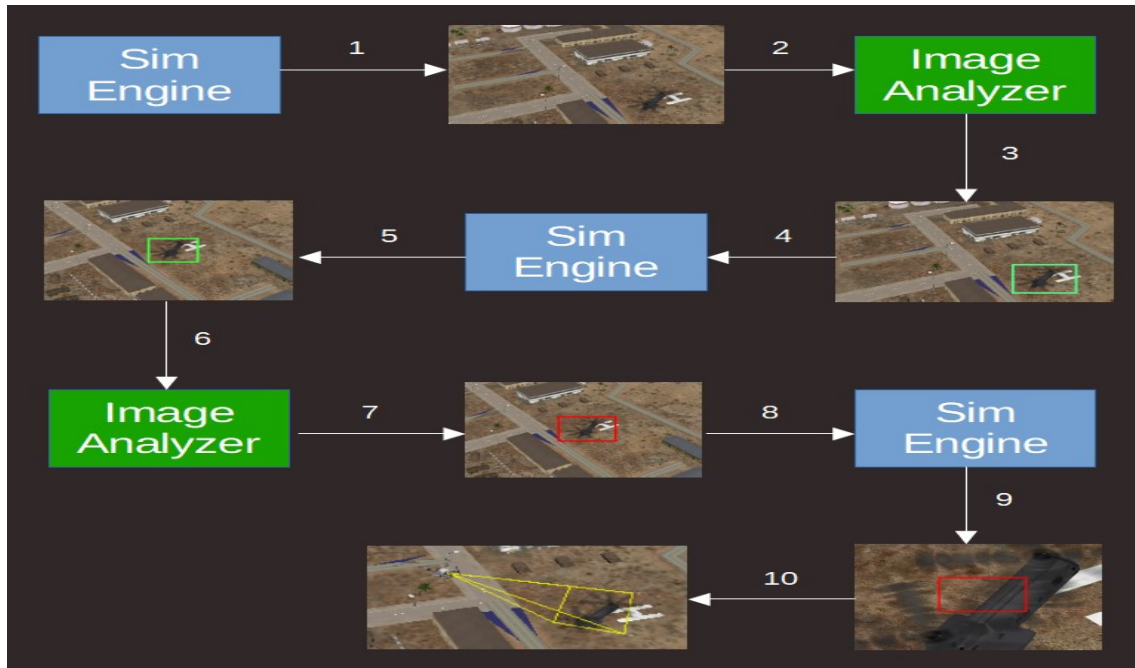
When something is detected, the drone receives the (x,y) coordinates of the Poi on the image. Knowing the Field of View (Fov) and the payload orientation at the time of the image (the detection may come with a delay since the image analyzer has its own processing time), the drone is capable of zooming toward the estimated Poi. With a higher resolution image, the analyzer can confirm or invalid the Poi as a target.

When something is detected, the drone receives the (x,y) coordinates of the Poi on the image. Knowing the Field of View (Fov) and the payload orientation at the time of the image (the detection may come with a delay since the image analyzer has it's

own processing time), the drone is capable of zooming toward the estimated Poi. With a higher resolution image, the analyzer can confirm or invalid the Poi as a target.

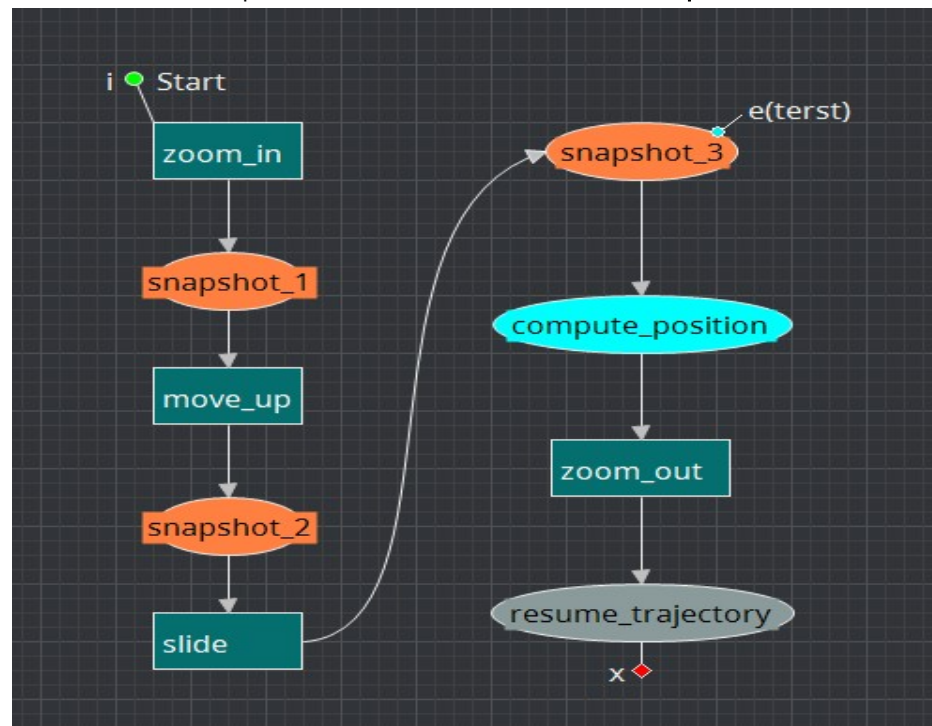


If the Poi is recognized as a valid target, the drone will suspend the trajectory and start a specific pattern to determine the target geodetic position. During this process, the drone is continuously using the feedback of the Image Analyzer to center the camera focus on the Poi (red square on image).



- ✚ The Simulation Engine (SE) rotates the drone payload to scan the area around it, at high Fov. Every second, a snapshot is taken and sent to the Image Analyzer.
- ✚ The Image Analyzer (IA) processes the received snapshot to identify a potential threat.
- ✚ When a point of interest (Poi) is found on the image, the IA mark it with a green square.
- ✚ The coordinates on the image of the square center is sent to the SE.
- ✚ The SE stops scanning the area and moves the payload to center the Poi coordinates.
- ✚ Snapshots are sent to the IA continuously to correct the centering from the updated Poi coordinates.
- ✚ If the IA identifies a potential threat, the mark becomes red.
- ✚ The SE stops the drone motion and starts the triangulation maneuver.
- ✚ Camera zooms on the Poi to allow more accurate recognition. Snapshots are continuously sent to the IA to maintain the centering of the target using the mark position acknowledge.
- ✚ The payload angles and drone coordinates will be used to estimate the target geodetic location.

The two most important ones are: **search** and **acquire**:



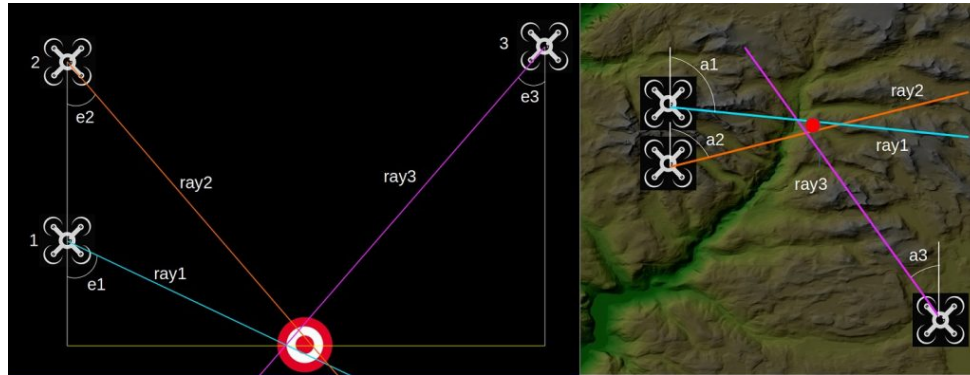
Acquire logic sequence

The **Search** logic is rotating the payload (azimuth) and controls its elevation at each full revolution (360°), from -20° to -40°, by step of 10°. A snapshot is taken every second and sent to the Image Analyzer. The logic is left as soon as a Poi is notified.

The **Acquire** logic is controlling the copter motion for each Poi detection. Snapshot *actions* are performing ray shooting as explained below. The difference between a *task* (green rectangle) and an *action* (ellipse) objects is that a *task* is an iterative process which takes time (such as a C++ procedure) and quits when the job is done, while an *action* is a one cycle job (like a function), which quits immediately.

Triangulation

Since the only known values are the drone geodetic position and the absolute angle values of the payload in terms of azimuth (0° = north) and elevation (0° = down), it is necessary to make several measures before extracting the target position.



Sliding Rings Algorithm

This algorithm gives the two closest points on two non-intersecting 3D segments. The principle is to slide two rings along each segment until the distance is minimal. The process is iterative and stops when the step falls below a certain small value. The step starts at $1/3$ of the smallest segment then is divided by 2 at each iteration. An iteration works around the ring position found at the previous iteration until the step is small enough to give up.

Future Work

This prototype is the first stage of a more complete project. Better Image Analyzer is on work. More realistic scenery will be used based on IG like Unreal. A more accurate copter dynamic model could be used although it is not that important as long as the payload is stabilized and images are not too fuzzy.

The following topics will be covered on due time and the document updated accordingly.

Statistics

Once a scenario is defined, vsTASKER can use its batch capability to chain thousands of run at high speed (the maximum the Image Analyzer can support), while changing some parameters at each run (like scanning Fov, copter speed, payload rotation speed, scanning pattern, vibrations, quality of the camera, etc.) and record the probability and time of detection, for later analysis.

Moving targets

The previous triangulation computations are based on stationary targets. When the Poi is moving, the time between the ray shots is reducing greatly the accuracy of the localization. This can be reduced by only throwing 2 rays for a quick approximation, then making the copter stationary and then refine the position according to payload adjustment angles to keep the target centered. The first problem to fix is to determine if the Poi is stationary or moving. This could only be provided by the Image Analyzer.

Using DEM terrain

If the terrain elevation is already known and loaded into the copter, the Poi localization shall become straightforward as only one ray shot would be sufficient to determine the ground coordinates (intersection of the ray with the terrain surface). With such digital terrain onboard, coordinates could be computed on the fly, at real-time, for any part of the image without making the copter change its course. Multiple Poi could then be localized at the same time.

Conversely, the copter could also focus and zoom on any provided Lat Lon coordinates, as the altitude would be extracted from the digital terrain, the vector could be computed and payload would be orientated accordingly to obtained angles.

OS & HARDWARE SUPPORT	INDUSTRY APPLICATION

Note: ** Specifications are copied from manufacturer data. Please contact before buying.