Object Correspondence in digital twin terrain

Vijayramsriram Sathanandhan
1233713197

Harshvardhan Sivasubramanian
1233780186

Teja Vishnuvardhan Boddu
1234122970

[1]Arizona State University.
[2]MS Robotics and Autonomous Systems.

***Abstract*— In this work, we present an integrated simulation-based framework for detecting and quantifying rock displacement in a digital twin terrain using autonomous aerial robotics. The framework utilizes ROS 2 for robotic middleware, PX4 for flight control, Gazebo Sim for high-fidelity simulation, and the YOLOv8 deep learning model for real-time object detection. The simulated environment is generated using photogrammetric reconstruction of real-world terrain and rocks, which are imported into Gazebo as mesh-based objects. A PX4 drone equipped with an RGB-D camera flies over the environment, capturing images for dataset generation and live inference. The trained YOLOv8 model detects the rocks in real-time from onboard camera feeds. Depth data and camera intrinsic parameters are used to estimate the 3D coordinates of the detected rocks in the camera frame, which are subsequently transformed into the world frame using TF2 and odometry. By comparing the world-frame positions of the rocks across different terrains (e.g., pre- and post-displacement), we effectively analyze object correspondence and motion. This project provides a foundational prototype for applications in geospatial change detection, site monitoring, and planetary terrain exploration.**

Keywords*— *ROS 2, PX4, Gazebo Sim, YOLOv8, TF2, Object Detection, Photogrammetry, Meshroom, Blender, UAV, Visual Odometry, Rock Correspondence

Introduction

The detection and quantification of environmental changes, especially the displacement of objects in a physical landscape, is a vital capability in fields such as environmental monitoring, construction validation, planetary science, disaster response, and archaeological surveying. Traditional manual methods of surveying such terrains are not only time-consuming but also prone to inaccuracies and may pose significant risks in hostile or remote environments.

To address these challenges, autonomous UAVs equipped with computer vision systems provide a scalable and efficient solution. When deployed in conjunction with high-resolution terrain maps and robust simulation environments, these systems can model, observe, and analyze object changes with a high degree of precision.

In this project, we focused on simulating such a system using the PX4 flight stack in Gazebo Sim, with a goal to identify and track rock displacements in a digital reconstruction of real-world terrain. We began by reconstructing a terrain using photogrammetry techniques from a series of overlapping images. These were processed through Meshroom to generate a textured 3D mesh, which was cleaned and optimized in Blender. The resulting model, along with individually reconstructed rock models, was imported into Gazebo to form a controllable and repeatable virtual testbed.

Our UAV platform of choice was the PX4-supported x500 drone with a depth camera. The drone captured RGB and depth data in the simulated terrain. Using manually collected and labeled image datasets, we fine-tuned a YOLOv8 model to detect four distinct rock types. Real-time inference was performed during flight to detect and locate these rocks. Coordinate estimation was done using depth data and pinhole projection, followed by TF2-based transformation to compute world-frame positions. A comparison of rock coordinates between two terrains—before and after simulated displacements—enabled us to track movements with measurable accuracy.

This end-to-end system reflects a robust and extensible approach to object correspondence in dynamic scenes and can serve as a foundation for future autonomous mapping and monitoring systems.

I. Methodology

The methodology comprises several key stages, each addressing a vital component of the system: terrain modeling, simulation environment setup, UAV deployment, data collection, model training, real-time inference, coordinate estimation, and displacement analysis.

A. Terrain and Object Modeling

We began by capturing approximately 150 photographs of a small terrain area with sufficient overlap (20–30%) between successive frames to ensure effective photogrammetric reconstruction. These images were imported into **Meshroom**, an open-source structure-from-motion (SfM) and multi-view stereo (MVS) pipeline. The output was a textured 3D mesh in .OBJ format, along with associated UV maps.

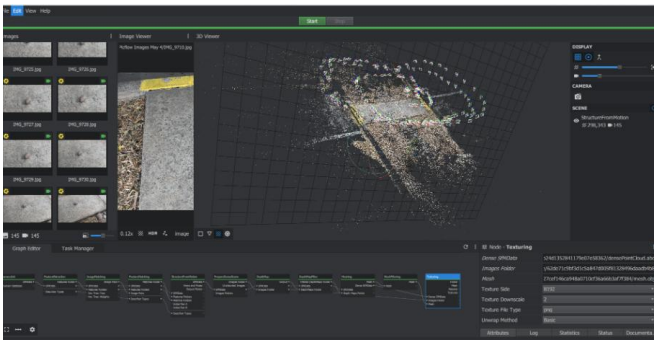

Fig. 1. Meshroom 3d reconstruction

The mesh was imported into **Blender** for cleanup. Artifacts such as floating vertices and non-manifold edges were removed. The terrain was re-centered, scaled, and flattened appropriately to ensure correct integration into the Gazebo physics environment. Textures were retained and remapped as needed. Similarly, individual rocks were reconstructed and processed to serve as detectable objects.

B. *Simulation Setup in Gazebo*

The final terrain and rock meshes were incorporated into a **custom SDF world** file. Each rock was placed as a <model> element at a known pose, and the terrain mesh was marked as static. To enhance realism, lighting, material properties, and collision geometries were configured.

The **PX4 x500_depth drone** was spawned using a ROS 2 launch file, which initialized:

- The Gazebo simulation engine with the correct world.
- The ROS-Gazebo bridge for sensor and control topic translation.
- The keyboard teleoperation node for manual drone control.
- A static TF from base_link (drone body) to camera_link.
- A dynamic TF broadcaster that publishes the transform from world to base_link using PX4 odometry data.

Environment variables such as PX4_GZ_WORLD and PX4_GZ_POSE were configured to dynamically select the world file and drone spawn location.

C. *Data Collection and Labeling*

Using the keyboard control node, the drone was manually flown around the terrain to capture diverse views of all rocks. The RGB images were saved in batches, and corresponding camera info and depth data were stored for future processing.

Fig. 2. Yolo training dataset.

Each image was manually labeled using **Roboflow**, annotating bounding boxes for each rock as one of four classes (rock1 to rock4). The labeled dataset was split into training and validation sets in YOLO format.

D. *YOLOv8 Training and Optimization*

We utilized the **Ultralytics YOLOv8** framework, beginning with a pretrained yolov8n.pt model (nano variant). The training configuration included:

- Epochs: 100
- Image size: 640x640 pixels
- Batch size: 8
- Early stopping: patience=20\

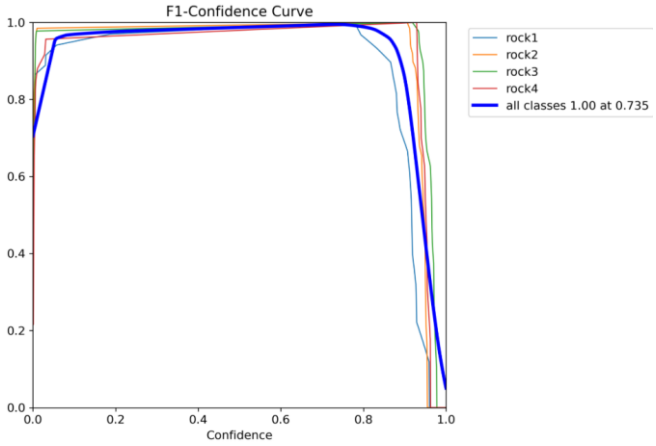

Fig. 3. F1-Confidence curve.

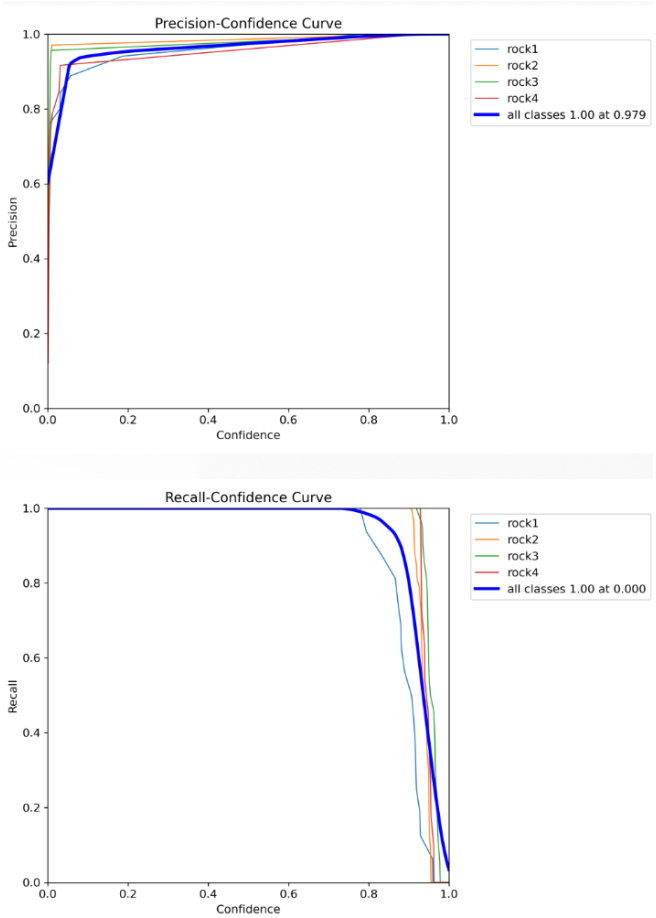

Fig. 4. Precision & recall confidence curve.

Training was performed on GPU. The final model achieved a loss of ~0.01, with a validation mAP@0.5 of 0.995. The PR curves, confusion matrix, and F1-Confidence plots indicated a stable and high-performing model with negligible overfitting.

E. *Coordinate Estimation and Transformation*

To compute the 3D location of detected rocks, we used the center pixel of each YOLO bounding box. The corresponding depth value was extracted from the aligned depth image. Using the intrinsic camera parameters from the

/camera_info topic, the 3D position in the camera frame was computed via the pinhole projection model:

$$X = \frac{(u - c_x) \cdot Z}{f_x}, \quad Y = \frac{(v - c_y) \cdot Z}{f_y}, \quad Z = \text{depth}(u, v)$$

These coordinates were then transformed to the world frame using the TF2 tree:

- A dynamic transform (world → base_link) based on PX4 odometry
- A static transform (base_link → camera_link)

The final world coordinates of each rock were logged in a CSV file and visualized in **RViz2** using marker topics.

Fig. 5. RQT graph.

II. Experiments and Results

A. Detection Performance

The YOLOv8 model achieved highly reliable results:

- **Confusion Matrix:** Perfect classification with no class overlap.
- **PR Curve:** Maintained precision and recall above 0.99 across thresholds.
- **F1-Confidence:** F1 scores above 0.98, indicating balanced precision and recall.
- **Inference Time:** ~15–20 ms per image on GPU, suitable for real-time operation.

B. Localization Accuracy

World-frame coordinates estimated using depth + TF2 were visualized in RViz2. The detected rock positions were stable across multiple runs with standard deviation within ±0.2 meters, validating the consistency of the camera model and depth estimation.

C. Rock Displacement Detection

To simulate terrain changes, a second world was created with the same terrain mesh but repositioned rocks. Running the same detection pipeline yielded new coordinate sets, which were compared to the originals.

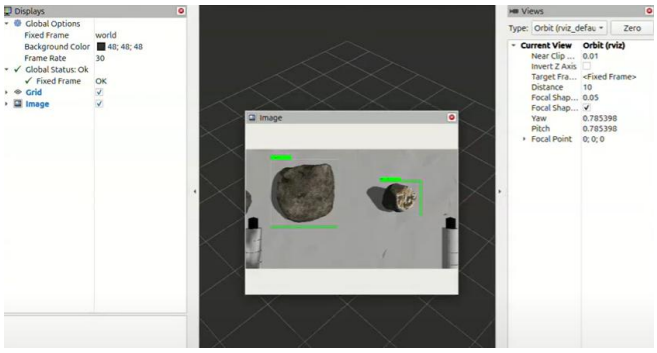

Fig. 6. Rviz2 visualization.

The system successfully detected rock displacements ranging from **0.2m to 0.4m**, with visual confirmation in RViz2 and numeric differences logged in CSV files. This validated the system's ability to track object correspondence over time and across scenes.

III. Proposed System

The complete pipeline consists of five ROS 2 nodes:

1. **control.py** – Publishes velocity and arming commands from keyboard inputs.
2. **yolo.py** – Loads YOLOv8 model and performs live inference from the RGB feed.
3. **coordinates.py** – Computes camera coordinates, applies TF transformations, and logs world-frame rock positions.
4. **tf_broadcaster** – Broadcasts the dynamic odometry-based world → base_link TF.
5. **ros_gz_bridge** – Handles bridging for RGB, depth, camera_info, and other Gazebo topics.

These nodes work together to form a real-time loop: detect → localize → transform → visualize → compare. The system is modular, extensible, and ready for both simulation and real-world UAV deployment.

IV. Conclusion

This project demonstrates a fully integrated and simulation-driven framework for detecting and analyzing object displacement in digital twin environments using UAVs, computer vision, and robotics middleware. By combining photogrammetry-based terrain reconstruction with PX4-based drone simulation, ROS 2 infrastructure, and deep learning-powered object detection via YOLOv8, we have developed an end-to-end pipeline capable of identifying specific objects (in this case, rocks), estimating their spatial coordinates in real-world frames, and comparing them across different environments to infer displacement.

One of the significant achievements of this project is the successful integration of multiple advanced technologies—from 3D terrain modeling to robotic control, and from real-time visual inference to 3D coordinate transformation using TF2. The fine-tuned YOLOv8 model performed exceptionally well on our custom dataset, achieving near-perfect class separation and low inference latency, validating its suitability for UAV-based real-time object detection tasks. The accurate conversion of pixel-space detections into

world-frame coordinates using depth data and odometry-enabled transforms showcases the robustness of the system's perception pipeline.

Our methodology not only provided quantifiable insights into object movement but also demonstrated the power of simulation in testing complex robotic workflows without the constraints or risks of real-world deployment. This capability is particularly relevant in contexts such as planetary exploration, disaster site assessment, archaeological monitoring, or remote infrastructure surveillance—domains where physical access is limited, and precise change detection is essential.

Additionally, the modular and extensible nature of our ROS 2 package makes it adaptable to other applications or sensor configurations. The use of TF2 and standardized ROS interfaces ensures compatibility with existing SLAM, mapping, or control algorithms, thereby facilitating future enhancements and real-world deployment.

However, while the simulation-based results are promising, several limitations were identified. The absence of SLAM means odometry drift could affect long-term localization. The system currently requires manual matching of detected objects across different worlds. Furthermore, the reliance on a static camera-to-body transform may reduce adaptability in dynamic environments or if the camera is gimballed.

To address these limitations, future work will focus on:

- Integrating ORB-SLAM3 or Cartographer for drift-free localization.
- Automating correspondence matching using descriptor-based object re-identification or nearest-neighbor matching of coordinate logs.
- Incorporating 3D point cloud comparison tools such as CloudCompare for full-scene displacement analysis.
- Adapting the system to run onboard real UAVs equipped with stereo or RGB-D cameras for field validation.

In summary, this project provides a strong proof of concept for vision-based object correspondence and displacement detection using simulated autonomous drones. It demonstrates how computer vision, robotics simulation, and AI can be harmonized into a powerful toolset for spatial analysis and monitoring in challenging environments. The framework serves as a solid foundation for both academic research and real-world exploration systems, and opens up opportunities for further innovation in intelligent robotic perception and autonomous mapping.

REFERENCES

[1] G. Jocher, A. Stoken, J. Chaurasia, et al., “YOLO by Ultralytics,” *GitHub Repository*, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[2] PX4 Autopilot Development Team, “PX4 Open Source Flight Control Stack,” *PX4 Documentation*, 2023. [Online]. Available: https://docs.px4.io

[3] Open Robotics, “Gazebo Sim: Robot Simulation Made Easy,” *Gazebo Documentation*, 2023. [Online]. Available: https://gazebosim.org

[4] AliceVision Development Team, “Meshroom: A 3D Reconstruction Software Based on Photogrammetry,” *AliceVision Project*, 2023. [Online]. Available: https://alicevision.org

[5] Open Source Robotics Foundation, “Robot Operating System 2 (ROS 2): A Flexible Framework for Writing Robot Software,” *ROS Documentation*, 2023. [Online]. Available: https://docs.ros.org

[6] ROS 2 Developers, “TF2 Transform Library: Coordinate Framesfor Robot Systems,” *ROS 2 Intermediate Tutorials*, 2023. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Intermediate/Tf2/

[7] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[8] A. Redmon and S. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint*, arXiv:1804.02767, 2018.

[9] M. Cordts et al., "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Blender Foundation, “Blender – Free and Open 3D Creation Software,” *Blender.org*, [Online]. Available: https://www.blender.org

[11] M. Quigley et al., "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009.

[12] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” in *European Conference on Computer Vision (ECCV)*, 2014.

[13] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[14] T. Whelan et al., “ElasticFusion: Real-time dense SLAM and light source estimation,” *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[15] P. Besl and N. McKay, “A Method for Registration of 3D Shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. (for future extension using ICP in point cloud comparison)

[16] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation,” in *Robotics: Science and Systems (RSS)*, 2015.