2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

System Integration of Controlling Multi-vehicles by Manipulating a Renderer as a Recognition Tool

Park Kunbum[1] and Tsuchiya Takeshi*

***Abstract*— This study proposes to employ a rendering engine as a tool for recognition and control. The example of the study primarily demonstrates controlling two types of vehicles: a rover and a drone. The controls require recognizing a three-dimensional space and simulating movement in advance, and a renderer embedded in the controller enables more intuitive recognition and control. For example, it is possible to match the geometry and camera images obtained from the rover by switching them to the drone's point of view. The renderer is newly implemented and embedded in the controller to avoid the heavy computational demands of commercial engines. Consequently, this paper presents the corresponding experiments in the experimental section, with explanations based on simulations discussed in the implementation section. In conclusion, this study proposes a renderer that has been thoroughly researched as a recognition tool and shows an example.**

I. INTRODUCTION

Environmental recognition through sensors is very important in robotics, autonomous driving, and drones. The author of this paper has studied how intuitive recognition and control are when a renderer is built into the controller. In most cases, a renderer has been used primarily for visualization. By embedding a renderer in the controller, it becomes possible to sample beyond the sensor's current scope or create alternative views by resampling objects in a virtual space rather than just relying on direct sensor feedback. However, a dedicated renderer was introduced since it takes too much computing power for a commercial engine. In conclusion, utilizing a renderer as a recognition tool enables the intuitive development of control and recognition algorithms by reproducing the simulation's benefits and the sensor's extracted features.

Figure 1 shows the rover system's structure[1]. The features are extracted from the sensor, represented in the renderer, and resampled. In previous studies, this system produced SLAM-like applications.

Figure 2 shows a drone control(Tello EDU) system with a single camera, the Attitude Reference Sensor (ARS), and an altitude sensor that measures the distance to the floor. The odometry tracker uses image processing, feature prediction, and location information from an external tracking of its location as the drone moves. The take-off and landing operator is provided with the initial feature at the time of takeoff through the image synthesis of the rover, and by estimating the location at the time of landing, it can land on the rover again.

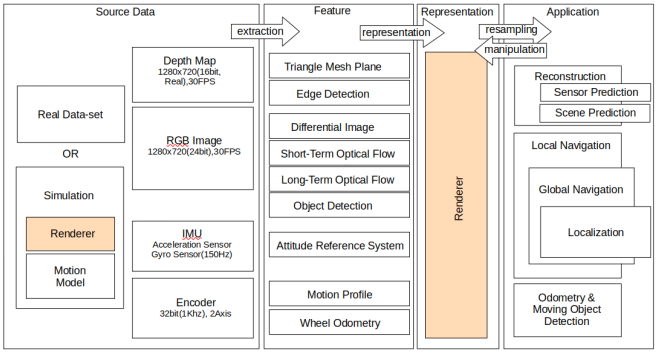

Fig. 1. Structure of the proposed software platform for active sensing robots: Rover

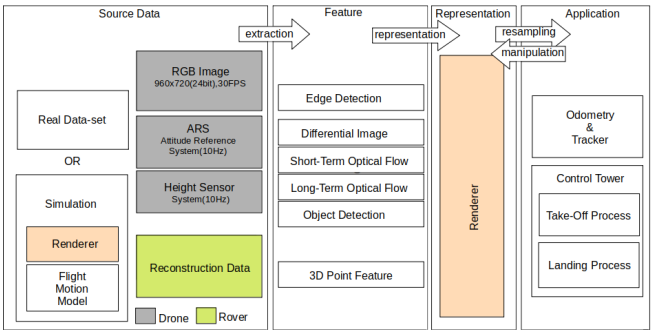

Fig. 2. Structure of the proposed software platform for active sensing robots: Drone

To prove the effectiveness of this study, it will show the cooperation of heterogeneous devices. The rover can operate for a relatively long time and has a large payload with multiple sensors. Moreover, the computing power is greater than that of other types of vehicles. However, in the case of drones, only a single camera, ARS, and altitude sensors are installed due to payload limitations. Still, they can fly to places the rover cannot go or recognize the environment at a higher altitude. Because the ARS is not synchronized with the monocular cameras and is somewhat limited in localizing within the environment, the rover's spatial information is crucial for tracking the location and guiding take-off and landing. This process rendered centralized data and resampled with different view rovers to the drone.

This work was not supported by any organization

[1]Kunbum Park received his bachelor's and master's degrees from Korea Aviation University (03-11). He then worked as a researcher at Samsung Electronics, Mando-Hella Electronics, and Hyundai Mobis for seven years (11-18). The author is currently affiliated with the University of Tokyo and Groove-X, Tokyo, Japan. (e-mail: asd441@naver.com)

*Corresponding author Takeshi Tsuchiya is the first author's supervisor and a professor in the Department of Aeronautics and Astronautics at the University of Tokyo. (e-mail:tsuchiya@g.ecc.u-tokyo.ac.jp)

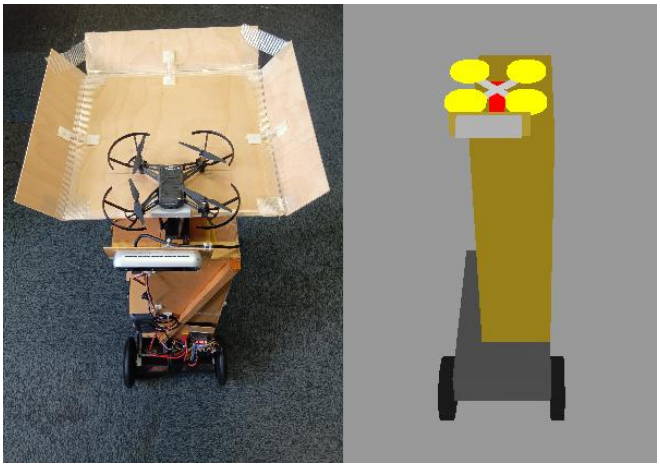

Fig. 3. Left: Actual platform (Rover and Drone: Tello EDU)), Right: Simulation Environment

In conclusion, this study aims to show how important the embedded renderer is in recognition and control to carry out a mission that complements the two types of vehicles with actual platform and simulation like Figure 3.

II. RELATED WORK

The author of this study conducted a series of studies in which a renderer is manipulated to recognize and argues that this is the novelty of this research. If we have technologies that have been thoroughly researched so far, not necessarily end-to-end, the studies are being conducted to utilize them better.

A. Computer Vision Process, Computer Graphics, Projection Geometry

In this study, textbook-level technologies in three intertwined fields were constantly used. Computer graphics is a rendering technology [2] that projects three-dimensional into two-dimensional space. In projection geometry[3], for example, three-dimensional data are obtained in two-dimensional features in areas such as Structure for Motion(SfM)[4]. Moreover, Computer vision[5] provides several signal-processing grounds for image processing.

B. Simultaneous Localization And Mapping (SLAM), Visual odometry(VO), Iterative Closest Point(ICP)

The field of tracking the movement of an object is called odometry. For example, wheel odometry, which counts wheels with encoders; visual odometry or visual-inertial odometry [6], which tracks the trajectory with cameras; and ICP[7], [8], which tracks with point clouds throughout LIDAR, can be listed. Since this odometry has errors such as drift, which eventually accumulate, techniques such as loop closure, filter, and relocalization are needed to revise errors. SLAM[9], [10], [11], the collective name for these techniques, provides the theoretical basis for this study, which requires tracking the location of vehicles.

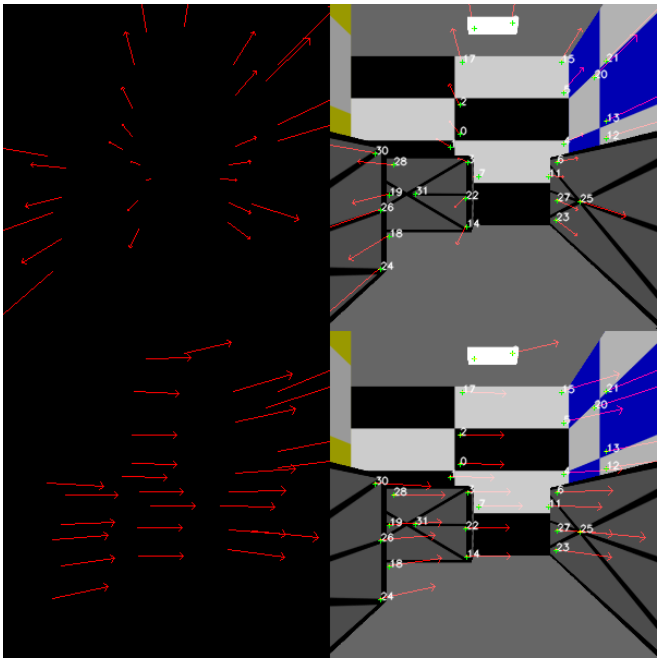

Fig. 4. Prediction for moving forward and rotating left: The tail of the red arrow represents the starting point of the motion, whereas the head represents the endpoint of the predicted motion.

C. Prior Study

The prior study[1]'s implementation controlled a rover by developing SLAM-like technology with a controller equipped with a rendering engine. Among them, this subchapter lists the parts used in this study.

- Prediction of the movement of feature points on odometry with the driving model: Since the rover rotates and moves forth or back, it is possible to predict the moving trajectory if the 3D point of the feature point is known. The renderer draws the trajectory with 3D points before and after movement. If there is a feature point on this trajectory, it can reduce computing power because it operates like an initial guess like Figure 4.
- Predictive scene generation: The rover can use the depth camera to reconstruct the scene's geometry and texture with a sequence of images on movement or interpolation. The agent can generate scenes with different poses and positions with a view change.
  The left picture in the figure is obtained by the rover's RGBD camera, and the right is obtained after moving forward the view and rotating it to the left.
  Figure 5.1 is the scene obtained from the rover after moving around more than 10 meters. While the odometry calculates the rover's position and pose, errors gradually accumulate. Figure 5.2 was generated by moving the view to the position and the position of the odometry. These two figures can be matched, and the rover's position can be corrected. That is, the accumulated error in the odometry can be reduced to the degree of measurement error in the RGBD camera.

As an extension of this study, the goal is to control the

Fig. 5. Scene generation and matching : 1: image after traveling 2: generated scene on position and pose where odometry indicates 3: matching two scenes with SIFT algorithm

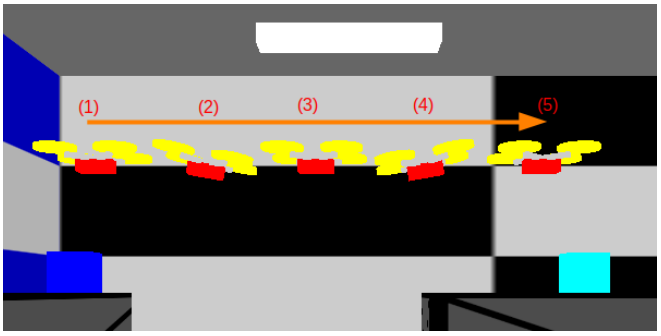

Fig. 6. Locomotion Sequence: (1)Start (2)Acceleration (3)Cruise (4)Deceleration (5)Arrival

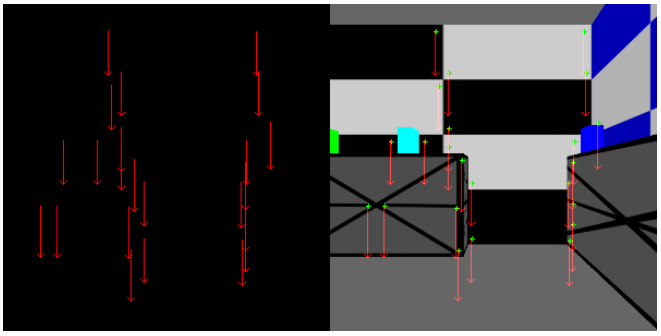

Fig. 7. Feature prediction on moving up

odometry, tracking, and landing of a drone that took off from the rover and to obtain high-altitude scenes that could not be obtained from the rover.

III. IMPLEMENT

A. Locomotion Model

The drone used in this study supports position control mode. It supports transition x(left/right), y(up/down), z(forward/backward) movement and yaw rotation. Since performing control at a lower level than the position level in a non-real-time operating system is inappropriate, the simulation and the experiment are conducted with position control mode. A key difference from the rover is that, due to the drone's movement characteristics, a change in pose is inevitable, as illustrated in Figure 6. For example, if you move forward, it is accompanied by a change in the pitch. Therefore, the model is modified by giving the angle of the pose to the acceleration/deceleration section of the transition.

$$x_{position} = \sum Speed \cdot \cos(yaw_{pose})\Delta t$$

$$y_{position} = \sum Speed \cdot \sin(yaw_{pose})\Delta t$$

$$Attitude_{pose} = \alpha \cdot MovingAverage(Acceleration)$$

The attitude means pitch in the case of z-locomotion and roll in the case of x-locomotion. The alpha of the formula was measured with the maximum point of the movement result of the optical flow in the actual image.

The odometry is performed with the yaw value and the position of the command. Still, the measurement of the single action (rotation, transition, hovering) is measured in the experimental chapter to see how much error there is. This is the value that the drone system has when the performance value of this odometry cannot be corrected from external data.

B. Odometry and Tracking

Since errors accumulate in odometry, correction must be performed. The location of the odometry and the relative location seen from the rover are updated by applying a linear Kalman filter. The prediction of the feature, the position seen from the rover, is considered an observation, and the prediction of the feature is used as a model.

Trajectory prediction of feature points for the moving model: It is possible to predict where the feature will move before the vehicle moves like Figure 7. 3D coordinates are not required in the case of rotation, but the translation must obtain the first 3D point from the rover. This process takes place before takeoff.

When the rover can see the drone, the location of the drone is specified by combining the results of object detection[12], difference image, and moving object detection, and the drone's location is updated to the smallest value in the depth map. This is because there are many non-drone pixels in the bounding box, like Figure 8, and the drone size is small.

$$\bar{P}_{k+1} = P + Q \tag{1}$$

P is a covariance matrix, and Q is the noise of the depth camera.

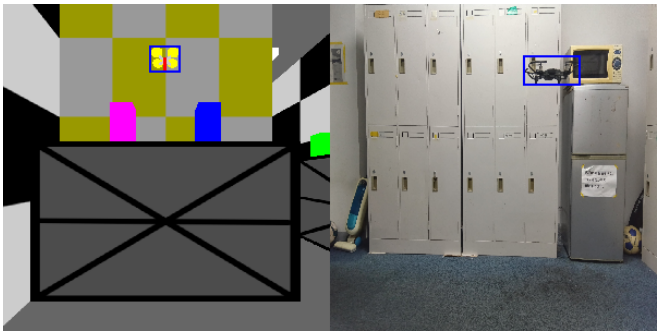

Fig. 8. Object Detection- Left: Simulation, Right: Experiment

$$z = Pos_{VO} \tag{2}$$

$$Pos_{VO}, Pose_{VO} = \min\left(\sqrt{(O_{Init} \cdot R \cdot T - O_{Opt})^2}\right) \tag{3}$$

z is a visual odometry result. VO minimizes error rotation matrix R and translation matrix T.

$$K = \bar{P} \cdot H^T \cdot (H \cdot \bar{P} \cdot H^T + R)^{-1} \tag{4}$$

$$\bar{x} = N_{rover} \cdot Depth_{rover}, x = \bar{x} + K \cdot (z - H \cdot \bar{x}) \tag{5}$$

$\overline{x}$ is the drone's position of YOLO detection from the rover. N is a normalized plane vector of the rover's camera and a minimum depth of the bounding box.

$$P = \bar{P} + K \cdot H \cdot \bar{P} \tag{6}$$

With a Kalman filter of equations (1) to (6), the position observed in the rover and the position estimated by the drone from the three-dimensional data received from the rover are fused. The yaw is depend on visual odometry.

C. Taking off and Landing Process

The drone must return to the rover at the end of the mission because the flight time limit is shorter than the rover's operating time. Since takeoff presupposes that the camera and the drone are set in the same direction, they can share the same field of view and obtain 3D coordinates of the feature points once they take off.

Figure 9 is a flowchart of the procedure for taking off, exploring the surroundings, and landing back on the rover. In the 3D points-shared state, the drone takes off. It is recognized that the distance to the ground has deviated from the rover as it moves. The rover moves while carrying out the mission and then returns on the rover. At this stage, it is recognized that the drone is hovering on the rover when the distance to the ground has decreased significantly. Match the x and z positions with the rover's screen and the drone's position, and land to retrieve the drone.

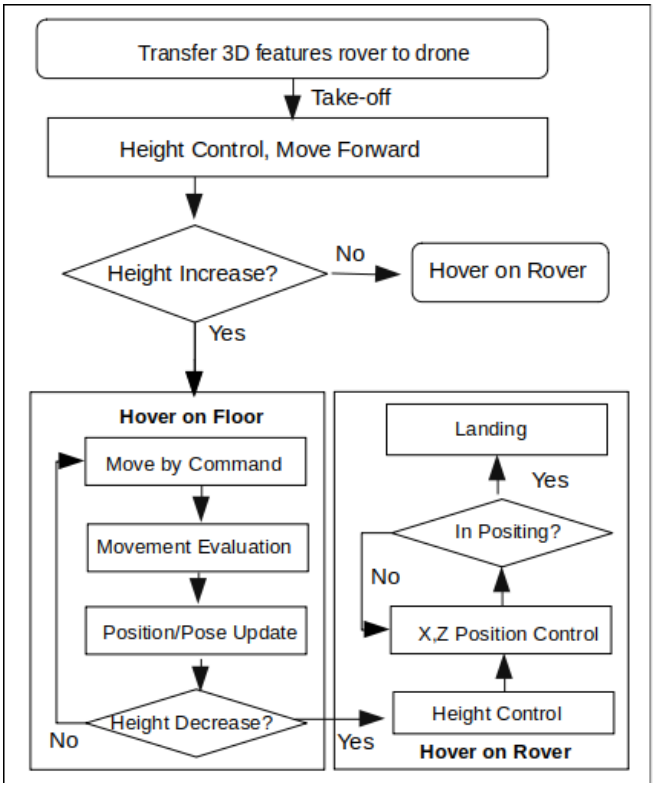

Fig. 9. Prediction for moving forward and rotating left: The tail of the red arrow represents the starting point of the motion, whereas the head represents the endpoint of the predicted motion.

IV. EXPERIMENT

A. Drone Camera Scene Generation and Matching Test

Before takeoff, the drone obtains the position of the feature point visible from the drone from the reconstruction obtained from the rover. After calibrating the drone camera[13], the agent produces the scene again with the obtained drone's intrinsic parameter. Assuming that the starting position is always the same, only the relative pose for the rover is obtained. When the error in the coordinates of the SIFT[14] matching algorithm is the smallest within the range of the 2-sigma standard deviation, the matching group is obtained like Figure 11.

The initial drone setting position is manually set the drone to (0 cm, 4 cm, 11 cm) compared to the camera, as shown in Figure 3. After SIFT matching is performed, the matching points are not horizontal and have different positions on the pixel frame, as shown in Figure 10; in addition, the field of view(FOV) is different, and the x and y coordinates of the matched points are different, so even if there is an error in matching, it cannot be detected. Therefore, the FOV is set equally through scene generation, the position is known, the pose is assumed to have an offset, and the optimal matching position is found by changing the pose to be brute-force.

B. Singe Action Experiment Evaluation of Locomotion

Table I shows the result of receiving 3D coordinates from a rover, performing only one action after takeoff, landing,

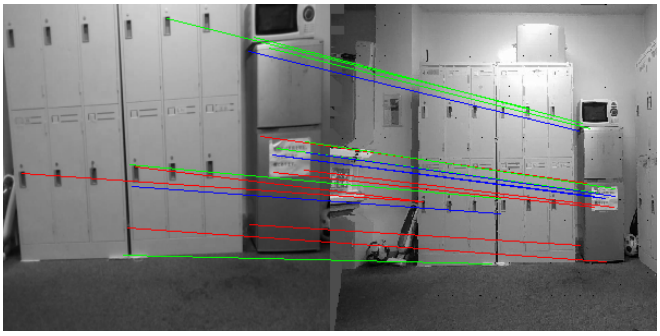

Fig. 10. SIFT Matching without Scene Generation: Left(Drone's camera image), Right(Rover's view)

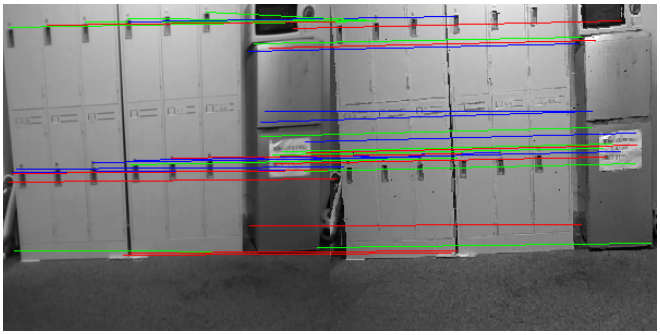

Fig. 11. SIFT Matching result: Left(Drone's camera image), Right(Generated image by the Renderer)

and manual measurement. Since all motions are composed of these single actions, it is possible to predict drift before external information corrects the accumulation of errors. Each scenario was conducted and measured five times.

TABLE I

ERROR RATE ON SINGLE LOCOMOTION

Movement	Error
Move Forward 0.5 meter	15.2%
Move Forward 1 meter	14.7%
Move Backward 0.5 meter	15.3%
Move Backward 1 meter	13.8%
Move Left 0.5 meter	12.1%
Move Left 1 meter	11.3%
Move Right 0.5 meter	13.2%
Move Right 1 meter	11.3%
Move Rotate 30 degrees	9.2%
Move Rotate 45 degrees	7.9%
Move Rotate 90 degrees	8.6%
Move Rotate -30 degrees	8.9%
Move Rotate -45 degrees	8.4%
Move Rotate -90 degrees	9.5%

The author's lab has several of the same models, so when tested, there are individual differences and biases. For example, forward movement has only positive errors and backward movement has only negative errors. Most of them estimate that yaw rotation is about 10% per single action and movement is about 15-20%.

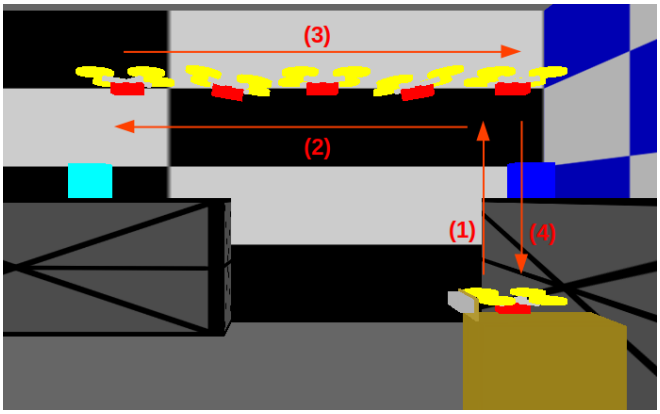

Fig. 12. Test Sequence: (1)Take-off (2)Moving-out, Taking-out a scene (3)Moving-in (4)Landing

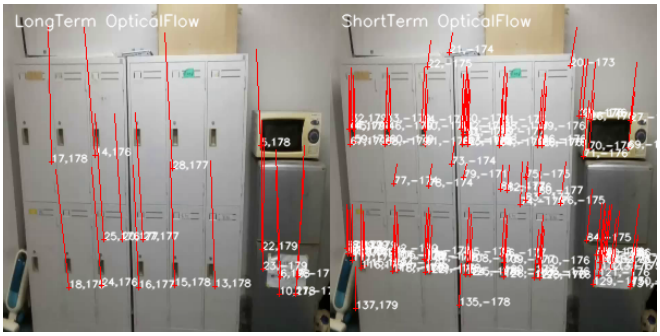

Fig. 13. Drone Optical flow on Odometry: Left(Long-Term), Right(short-Term)

Fig. 14. Left: Scene of the rover's camera, Right: Scene updated by the Drone

C. Take-off, Flight and Landing Sequential Test

As the final test, as shown in Figure 12, after taking off and moving to leave the rover, obtaining a scene that cannot be seen in the rover, it returns to the top of the rover and lands. Since the base for landing the drone is 30cm by 30cm like Figure 3, it can be landed if there is a precision of approximately +-15cm.

To calculate the odometry, optical flow features are received from the SW platform. The long term used good FeaturesToTrack[15], and the short term used the Features from Accelerated Segment Test(FAST)[16] algorithm like Figure 13.

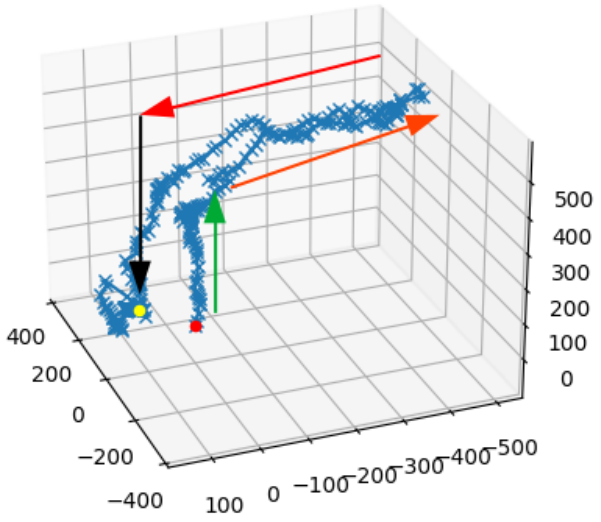

Fig. 15. Estimated Trajectory: Take-off(green),Moving forward(Orange), Moving backward and Landing process(Read), Landing(Black)

Figure 14 is a picture taken by the drone outside the view of the rover. It is written "DRONE TEST SII 2025". The test trajectory is shown in Figure 15. Errors that landed at this time have an average of 13.2 cm and a standard deviation of 5.1 cm in 10 repetitions. Since the landing base has guards to protect, the case where it landed on the guard was treated as 15 cm.

V. CONCLUSIONS

A. Discussion

As a result of this study, the agent can transfer geometry information by converting it into a viewpoint viewed by other types of robots through geometry taken from one robot and camera images by its embedded renderer. Therefore, this study reveals that it can be used to control small agents that do not have geometry. This type of utilization can be used for heterogeneous equipment to share the same three-dimensional space to integrate, resample, and control data.

B. Conclusion and Contribution

A series of studies[1], [17], [18], [19], including this study, contribute to using it for spatial recognition of a renderer. In particular, in this study, collaboration with other types of vehicle drones was implemented using data obtained from the rover. As vehicles with two complementary characteristics collaborate, a process for operating and gains was proposed. Through this study, it is proposed that the vehicle controller reconstructs and maintains a three-dimensional space in the future and performs control while re-sampling there.

C. Future-work

The drones used in this study are commercial and too small to install manually. Therefore, it is an inconvenience that a person has to charge and position it again after each landing. It would have been a more attractive project with more custom drones using wireless charging devices. If multiple drones can repeat taking off and landing while charging themselves, the rover's limitations and small indoor drones' limitations can be supplemented.

REFERENCES

[1] P. Kunbum and T. Takeshi, "Employing an embedded renderer as recognition tool for odometry, map-building, navigation, and localization on active sensing robotics," *IEEE Journal of Indoor and Seamless Positioning and Navigation*, pp. 1–18, 2024.

[2] J. F. H. et al., "Computer graphics: Principles and practice 3rd edition," pp. 387–668, 2013.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[4] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day," *2009 IEEE 12th International Conference on Computer Vision*, pp. 72–79, 2009.

[5] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed., ser. Texts in Computer Science. Springer, 2022.

[6] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[7] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." *Robotics: Science and Systems*, vol. 2, no. 9, pp. 1–9, 2014.

[8] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, 2018.

[9] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[10] 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.

[11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[12] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[13] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1330 – 1334, 12 2000.

[14] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–, 11 2004.

[15] J. Shi and C. Tomasi, "Good features to track," Cornell University, Tech. Rep. TR 93-1399, 1993.

[16] Y. Biadgie and K.-A. Sohn, "Feature detector using adaptive accelerated segment test," vol. 33, 05 2014, pp. 1–4.

[17] P. Kunbum and T. Tsuchiya, "Rendering involved and machine-learning-based environment interpretation," *2023 IEEE/SICE International Symposium on System Integration (SII)*, pp. 1–5, 2023.

[18] K. Park and T. Tsuchiya, "3d scene description by pretrained features and icp-based odometry," *Dimensional Optical Metrology and Inspection for Practical Applications XI*, vol. 12098, 2022.

[19] P. Kunbum and T. Tsuchiya, "Rendering involved and machine-learning-based environment interpretation applying depth estimation," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2023, pp. 2483–2488.