ELG5163 – Machine Vision
UOttawa, April 2022

Inter-vehicle Distance Estimation Using Displaced Stereo Vision

Alfa Budiman, Mathieu Falardeau
University of Ottawa, Faculty of Engineering

Abstract *– In this paper, we propose a vision-based inter-vehicle distance estimation algorithm by combining inputs from an overhead camera mounted on an Unmanned Arial Vehicle (UAV) with a ground-based mobile robot acting as an Unmanned Ground Vehicle (UGV). The two vision inputs were combined in a manner like stereoscopic vision to estimate the distance between the ground based mobile robot and a moving target. Two triangulation methods were used to estimate the distance: Closest-Points method and Sine Law method. Each method was tested over two scenarios in a simulated environment and returned mix results.*

Keywords – Inter-vehicle distance, Stereoscopic vision, Robot Operating System, Unmanned Aerial Vehicle, Unmanned Ground Vehicle

I. INTRODUCTION

Inter-vehicle distance measurement is very important for self-driving vehicles as it ensures that the vehicle maintains a safe distance with other vehicles and avoids accidental collisions. Different technologies such as laser and radar are available, but they are usually very expensive and difficult to procure/integrate. Alternatively, cameras are much cheaper and accessible options. Over the years, multiple inter-vehicle distance algorithms using cameras were proposed to meet this requirement. One solution was to detect edge features of the vehicle and estimate the inter-vehicle distance from the vehicle rear-shadows [1]. Stereo camera was also proposed which determines the distance between the camera and an object by detecting its feature points on the images [2] and calculating the object's image disparity between two cameras with known baseline distance [3]. Over-head cameras are also applied in determining the location of the vehicles [4] which can then determine the distance between the two UGV.

II. PROBLEM DESCRIPTION

Each of the previously mentioned methods have weaknesses. For stereo vision, this only works if there are a minimum of two cameras on each robot. Installing two cameras on each robot might be too expensive for companies who wish to reduce costs. Plus, UGV must maintain visual contact with each other which makes the method pointless if an obstacle is obstructing the cameras. The overhead camera can only measure distance in a 2D plane. This means that the distance measured will be inaccurate on uneven ground.

In this paper, a proposed solution is to combine both stereo vision and an overhead camera which is called displaced stereo vision. The concept is to take image captures from cameras attached to a UAV and a UGV and detect the desired object(s).

(a) (b) (c)
Fig. 1. Robot 3D model. (a) Quadcopter. (b) Wafflebot. (c) Burgerbot

Then, the method generates two vectors from the camera's input using non-inverted image projection. Using triangulation, the method calculates the distance between the UGV and the desired target object.

III. SYSTEM DESCRIPTION

The method was developed and tested in a simulated environment. All robots were designed using the Robotic Operation System (ROS) which is an open-source robotic middleware suite that is widely used in robotic applications. The simulated environment was generated in Gazebo simulator which is an open-source 3D robotics simulator available for ROS. The robots were programmed using the Python programming language, and the OpenCV and Numpy libraires were used for image processing and calculations respectively. Three robots were used to evaluate the displaced stereo vision method: Hector Quad Rotor (quadcopter Fig. 1(a)), Turtlebot 3 Waffle (wafflebot Fig. 1(b)) and Turtlebot 3 Burger (burgerbot Fig. 1(c)). Both the quadcopter and wafflebot acted as the UAV and UGV with cameras with known pose while the burgerbot was the target UGV with unknown pose. Additionally, the wafflebot's integrated lidar was used to measure the real distance between the two UGV.

IV. METHODOLOGY

The goal is to estimate distances between a target and a UGV. This will be done by combining monocular vision on the UGV's camera with monocular vision on an overhead camera in a similar manner to [4]. In this case, a quadcopter will act as that overhead camera. This method will occur over 4 steps: object detection, direction calculation, position calculation, and distance calculation. The last two steps are combined in each method.

A. Object Detection

The overhead camera can detect the UGV and the target object. While many methods exist for object detection using

(a) (b)

Fig 2. (a) Quadcopter camera image. (b) Wafflebot camera image

vision, for this application, color was used to facilitate ease of detection of the target [5], [6]. The target was colored in solid red while the wafflebot was colored in solid black.

The overhead camera sees the burgerbot as a group of red pixels. It also sees the wafflebot as a group of black pixels. The red pixel closest to the group of black pixels is the location of the burgerbot from the quadcopter's perspective. This pixel is annotated with a "+" on Fig. 2(a).

The wafflebot camera also sees the burgerbot as a group of red pixels. The center of these red pixels is the target's location from the perspective of the wafflebot. This is shown as the "+" on Fig. 2(b). These pixel coordinates, the locations of the "+" on Fig. 2(a) and 2(b), are extracted and then used to calculate the direction from the respective observer to the target as unit vectors.

B. Direction Calculation

After the desired pixel position (p_x and p_y) are found, the algorithm calculates the unit vector by applying the inverse of the camera projection matrix on the pixel position to determine the 3D ray vector [7] which is shown in Fig. 3. The simulated camera lenses do not have any distortion, so calibration is not required for this method. The camera's focal lengths on the x and y axis, f_x and f_y, are equal (1), and because the cameras are monocular, the translation terms T_x and T_y are equal to 0 (2). As such, the new projection matrix was simplified by using matrix (3) where c_x and c_y are the center pixel position of the camera.

$$f_x = f_y = f, \quad T_x = T_y = 0 \tag{1),(2}$$

$$P = \begin{bmatrix} f_x & 0 & c_x & T_x \\ 0 & f_x & c_y & T_y \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The desired vector v is calculated by applying the inverse of the projection matrix to the pixel coordinate using (4). The unit vector u is then calculated by dividing the vector v by its magnitude as shown in (5).

$$v = P^{-1} \cdot \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{f} & 0 & \frac{-c_x}{f} \\ 0 & \frac{1}{f} & \frac{-c_y}{f} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{p_x - c_x}{f} \\ \frac{p_y - c_y}{f} \\ 1 \end{bmatrix} \tag{4}$$

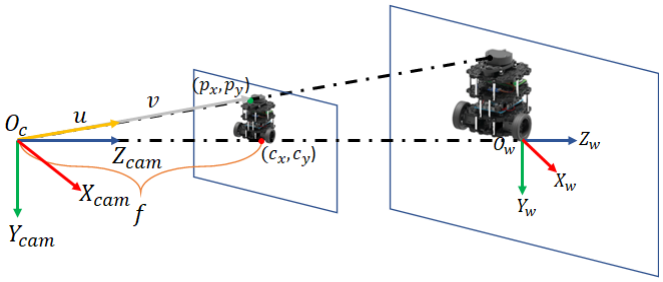

Fig. 3. Camera projection of UGV from real to image plane

$$u = \frac{v}{\|v\|} \tag{5}$$

Afterwards, the unit vectors must be re-oriented from their respective camera's reference frames to the world reference frame. Because the quadcopter's and wafflebot's poses are known as well as the poses of their cameras in reference to themselves, the unit vectors' orientation was rectified by using quaternion multiplication [8].

In ROS, the orientation is represented in quaternions. In (6), q is the quaternion orientation of the camera in reference to the world frame with vector part v_q and scalar part q_s. To rotate the unit vector in reference to the camera frame, u_{cam}, the method applies the conjugation of two quaternions onto u_{cam} shown in (8). This produces the unit vector in reference to the world frame, u_w, which is shown in (9).

$$q = (v_q, q_s) = [q_x \quad q_y \quad q_z \quad q_s]^T \tag{6}$$

$$(u, 0) = [u_x \quad u_y \quad u_z \quad 0]^T \tag{7}$$

$$(u_w, 0) = q(u_{cam}, 0)q^{-1} \tag{8}$$

$$u_w = ({q_s}^2 - \|v_q\|^2)u_{cam} + 2({v_q}^T \cdot u_{cam})v_q + 2q_s(v_q \times u_{cam}) \tag{9}$$

C. Distance Calculation - Closest Points

Using stereoscopic vision, an object's location can be calculated if given the direction from two cameras and the locations of these cameras [3]. In Fig. 4, L_q and L_w are the lines in 3D space from the quadcopter to the target (10) and the wafflebot to the target (11), respectively:

$$L_q = P_q + s\vec{v}_q, \quad L_w = P_w + k\vec{v}_w \tag{10),(11}$$

P_q and P_w are the locations of the quadcopter and wafflebot in reference to the world frame. These locations are obtained from the observers' onboard sensors. $\vec{v}_q$ and $\vec{v}_w$ are unit vectors from the quadcopter to target and wafflebot to target, respectively. These were obtained from direction calculation.

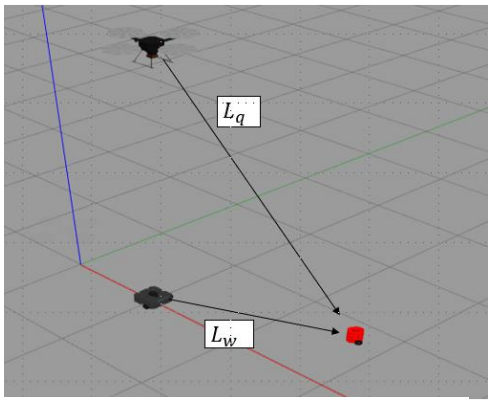

Fig. 4. Wafflebot and Quadcopter observing target

$$P_q = \begin{bmatrix} x_q \\ y_q \\ z_q \end{bmatrix}, \quad P_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \tag{12), (13}$$

$$\vec{v}_q = \begin{bmatrix} \Delta x_q \\ \Delta y_q \\ \Delta z_q \end{bmatrix}, \quad \vec{v}_w = \begin{bmatrix} \Delta x_w \\ \Delta y_w \\ \Delta z_w \end{bmatrix} \tag{14), (15}$$

Ideally, lines L_q and L_w would intersect, but due to the imperfection of the feature matching process in object detection and direction calculation, this will rarely occur. Therefore, the two closest points on non-intersecting lines must be found. The middle between these two points is the location of the target. The parameters s and k that yield the closest points are found when the following conditions are satisfied:

$$\left(L_q - L_w\right) \cdot \vec{v}_q = 0, \quad \left(L_q - L_w\right) \cdot \vec{v}_w = 0 \tag{16), (17}$$

Expanding these dot products yields a system of linear equations that can be expressed in matrix form $AX = B$ (18), (19). This system yields the following values of s and k (20).

$$A = \begin{bmatrix} \left(\Delta x_q{}^2 + \Delta y_q{}^2 + \Delta z_q{}^2\right) & -\left(\Delta x_w \Delta x_q + \Delta y_w \Delta y_q + \Delta z_w \Delta z_q\right) \\ \left(\Delta x_w \Delta x_q + \Delta y_w \Delta y_q + \Delta z_w \Delta z_q\right) & \left(\Delta x_w{}^2 + \Delta y_w{}^2 + \Delta z_w{}^2\right) \end{bmatrix} \tag{18}$$

$$B = \begin{bmatrix} -(x_q - x_w)\Delta x_q - (y_q - y_w)\Delta y_q - (z_q - z_w)\Delta z_q \\ -(x_q - x_w)\Delta x_w - (y_q - y_w)\Delta y_w - (z_q - z_w)\Delta z_w \end{bmatrix} \tag{19}$$

$$X = \begin{bmatrix} s \\ k \end{bmatrix} = A^{-1}B = \begin{bmatrix} X_{11} \\ X_{12} \end{bmatrix} \tag{20}$$

Therefore, the target's location, P_t can be calulcated by substituting the values of s and k from (20). The distance, d, between the wafflebot and the target is the magnitude of the difference between P_t and P_w (22).

$$P_t = \frac{1}{2}\left(L_q(s = X_{11}) + L_w(k = X_{12})\right) \tag{21}$$

$$d = \left|\left|P_t - P_w\right|\right| \tag{22}$$

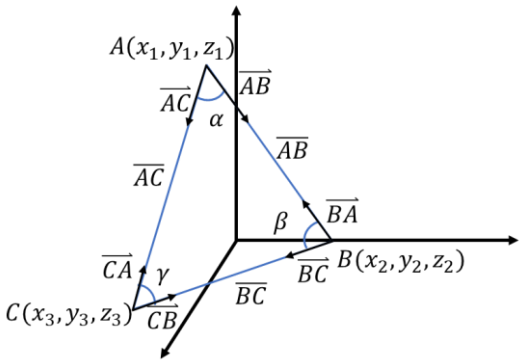

Fig. 5. Sine law triangulation method

D. Distance Calculation - Sine Law

The second method is based on triangulating the distance by using the law of Sine [3]. In Fig. 5, A represents the quadcopter, B represents the wafflebot and C represents burgerbot. By applying the law of Sine, the distance between the wafflebot and turtlebot, $\overline{BC}$ is calculated by determining α, γ, and $\overline{AB}$.

Because the quadcopter and wafflebot positions are known, vector $\overrightarrow{AB}$ and $\overrightarrow{BA}$ are calculated using (23) and (24). Using the unit vectors from the cameras ($\overrightarrow{AC}/\left\|\overrightarrow{AC}\right\|$ and $\overrightarrow{BC}/\left\|\overrightarrow{BC}\right\|$), $\overrightarrow{AB}$ and $\overrightarrow{BA}$, a vector multiplication on the vertices of A and B is used to determine α and β respectively, shown in (25), (26). γ is calculated by subtracting π by α and β (27). Finally, law of sine is used to determine $\overline{BC}$ which is the distance, d, between the two UGV (28).

$$\overrightarrow{AB} = B - A = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, \quad \overrightarrow{BA} = -\overrightarrow{AB} \tag{23), (24}$$

$$\alpha = \cos^{-1}\left(\frac{\overrightarrow{AB}}{\left\|\overrightarrow{AB}\right\|} \cdot \frac{\overrightarrow{AC}}{\left\|\overrightarrow{AC}\right\|}\right) \tag{25}$$

$$\beta = \cos^{-1}\left(\frac{\overrightarrow{BA}}{\left\|\overrightarrow{BA}\right\|} \cdot \frac{\overrightarrow{BC}}{\left\|\overrightarrow{BC}\right\|}\right) \tag{26}$$

$$\gamma = \pi - (\alpha + \beta), \quad d = \overline{BC} = \frac{\left\|\overrightarrow{AB}\right\| \sin(\alpha)}{\sin(\gamma)} \tag{27), (28}$$

Although the method is simple, it is not precise. This method assumes that the two unit-vectors from the cameras will always point to the same exact position of the burgerbot which is a false assumption. Position C represents an approximate position of the burgerbot. Depending on the desired error tolerance, this method would still work as a distance estimator.

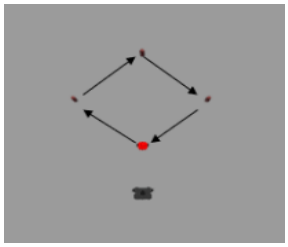

Fig. 6. Test Drive Waypoints

V. TESTING AND RESULTS

The distance calculation algorithms were tested by having the target drive through four waypoints as per Fig. 6. Two tests were conducted: test 1 with a stationary wafflebot and test 2 with the wafflebot following the target at a preset distance. The results of test 1 and 2 are shown on Fig. 7 and 8, respectively. They plot the calculated distances using closest points and sine law methods, as well as the measured distance from the wafflebot's onboard lidar.

For test drive 2 a simple proportional controller was used to control the wafflebot's position and orientation. This constituted a simplified form of visual servoing [9]. The wafflebot's linear speed, $\dot{x}$ was proportional to the distance between it and the target. Where K_1 is a control gain and d was calculated from (22)/(28).

$$\dot{x} = K_1 d \tag{29}$$

The wafflebot's rotational speed $\dot{\theta}$ was proportional to the angle between itself and the target. This was calculated as the angle between the vector of the wafflebot's direction $\vec{v}_{WB}$, and the unit vector pointing from the wafflebot to the target $\vec{v}_w$:

$$\dot{\theta} = -K_2 * \cos^{-1}\left(\frac{\vec{v}_{WB} \cdot \vec{v}_w}{||\vec{v}_{WB}|| * ||\vec{v}_w||}\right) \tag{30}$$

$\vec{v}_w$ is obtained from direction calculation and K_2 is a control gain. $\vec{v}_{WB}$ is known from the sensors on the wafflebot.

VI. DISCUSSION AND CONCLUSION

From Fig. 7, it can be observed that the calculated distances closely tracked the lidar distance in the stationary trial. However, for the leader follower trial in Fig. 8, the calculated distances diverge from the lidar distances.

During the leader-follower trial, the error (difference between the calculated and lidar distances) was less consistent. The error increased over the duration of the wafflebot's motion, but decreased when its motion was reduced.

This error was caused from the imperfection in the feature matching process. As the wafflebot moved, its pixel location on the quadcopter camera changed which may not be completely consistent/proportional with its motion. This change combined with a similar one in the burgerbot's pixel location resulted in a noisier signal. However, it should be noted that the overall distances are relatively small to the sizes

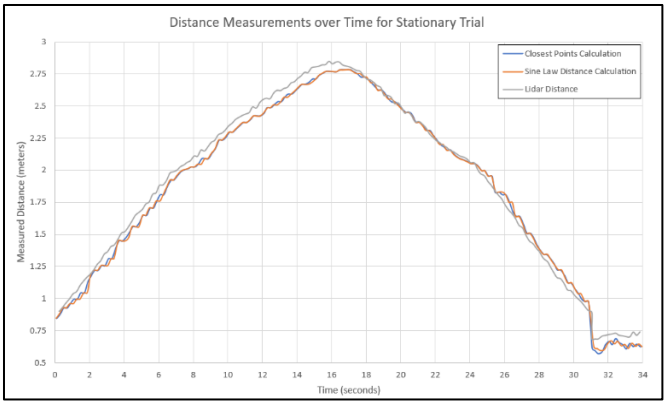

Fig. 8. Distance Measurements for Stationary Trial

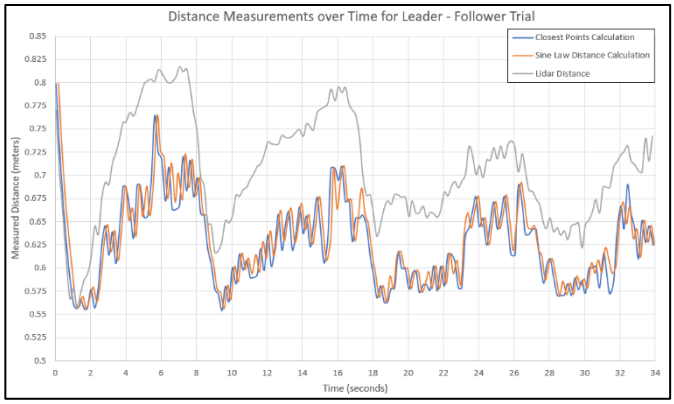

Fig. 7. Distance Measurements for Leader - Follower Trial

of the objects. If the overall distances were larger compared to the objects, then the relative error between the calculated and lidar distances would be significantly smaller.

Overall, vision data from both the quadcopter and the wafflebot were combined into a distance estimation solution which can be applied to provide additional information for coordination of multiple robots. For example, a group of robots may be required to move to a location while maintaining formation. While this form of platooning is possible using only monocular vision [10] it would require more data on object dimensions in its environment [1].

This maybe an issue when multiple different types of robots must cooperate with each other or with human operated elements such as in [11]. It may not be feasible to rely solely on monocular vision. Furthermore, while it is possible to add additional sensors, this still relies on maintaining line of sight to the targets / objects of interest.

An overhead view can track objects even if UGVs lose sight of the targets. This additional information can then be combined with previously collected data and then generate a path to the target. At the same time, relying solely on an overhead camera has limitations, namely, it requires additional information on the nature of the ground (changes in elevation, slope, camera height). Combining an overhead monocular camera with UGV mounted monocular cameras provides additional information to create a complete picture of the situation.

REFERENCES

[1] G. Kim and J. -S. Cho, "Vision-based vehicle detection and inter-vehicle distance estimation," 2012 12th International Conference on Control, Automation and Systems, 2012, pp. 625-629.

[2] Y. Shima, "Inter-vehicle distance detection based on keypoint matching for stereo images," 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017, pp. 1-6, doi: 10.1109/CISP-BMEI.2017.8302064.

[3] Abdelmoghit Zaarane, Ibtissam Slimani, et all, “Distance measurement system for autonomous vehicles using stereo camera,” Array Volume 5, 2020, 100016, ISSN 2590-0056.

[4] Z. Ziaei, R. Oftadeh and J. Mattila, "Vision-based path coordination for multiple mobile robots with four steering wheels using an overhead camera," 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2015, pp. 261-268, doi: 10.1109/AIM.2015.7222542.

[5] J. Li, J. Wang and J. Mao, "Color moving object detection method based on automatic color clustering," Proceedings of the 33rd Chinese Control Conference, 2014, pp. 7232-7235, doi: 10.1109/ChiCC.2014.6896196.

[6] A. N. Fitriana, K. Mutijarsa and W. Adiprawita, "Color-based segmentation and feature detection for ball and goal post on mobile soccer robot game field," 2016 International Conference on Information Technology Systems and Innovation (ICITSI), 2016, pp. 1-4, doi: 10.1109/ICITSI.2016.7858232.

[7] M. T. Bui, R. Doskocil and V. Krivanek, "Distance and angle measurement using monocular vision," 2018 18th International Conference on Mechatronics - Mechatronika (ME), 2018, pp. 1-6.

[8] Dr. P. Payeur. Machine Vision. (2022, Winter). ELG 5163. Ottawa, Canada: University of Ottawa.

[9] H. Mekki and M. Letaief, "Path planning for 3D visual servoing: For a wheeled mobile robot," 2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR), 2013, pp. 86-91, doi: 10.1109/ICBR.2013.6729262.

[10] P. Avanzini, B. Thuilot and P. Martinet, "Accurate platoon control of urban vehicles, based solely on monocular vision," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 6077-6082, doi: 10.1109/IROS.2010.5650018.

[11] C. J. R. McCook and J. M. Esposito, "Flocking for Heterogeneous Robot Swarms: A Military Convoy Scenario," 2007 Thirty-Ninth Southeastern Symposium on System Theory, 2007, pp. 26-31, doi: 10.1109/SSST.2007.352311.