Optimal visual servoing of a ground robot following an aerial object using a Pan-Tilt-Zoom (PTZ) camera

AAAI Press
Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

Abstract

In this paper, we describe a novel method of visually following/tracking a flying target by a ground robot, equipped with a Pan-Tilt-Zoom (PTZ) camera. The local planning task is formulated as a nonlinear optimization problem. A major contribution of this paper is that obstacle avoidance, robot motion and camera orientation are all solved in a single optimization problem. The entire code architecture has been described. Extensive results have been demonstrated in a ROS-based simulation. Finally, a real-world implementation is described along with some experimental results. The described architecture is shown to be performing satisfactorily in both cluttered and non-cluttered environments.

1 Introduction

Vision-based robotics has been one of the major research areas for the last few decades. A vision sensor accounts for a very versatile feedback instrument for a wide variety of robot control and manipulation problems. There are a large number of applications involving vision-based systems, ranging from self-driving cars to home automation. A very important subdomain of these applications is dynamic scene perception, which is a key challenge for all kind of autonomous robot operations in dynamic environments. Robust following of moving objects in scenes is vital for several applications, for example: a robot butler serving a person (Srinivasa et al. 2012) etc. The problem is particularly challenging because of randomness in motion of the target, which may lead to sub-optimal tracking performance or even absolute failure in many cases of robot's motion. In addition, the environment for such an application is usually not free of obstacles, which means the local planner needs to model them as well. The problem becomes even more complex when the motion of the object to be tracked is in 6 dimensions (x,y,z, yaw, pitch, roll), for example: a flying quadrotor that needs to be followed by a ground robot. Motion in a plane (x, y, θ) is usually not enough to feasibly follow such an object.

We propose the use of a pan-tilt-zoom (PTZ) dome camera for tracking such an object, which is a regular camera mounted on a 2DoF rotating platform, with capability of zooming into targets and autofocus. Such cameras are usually used for CCTV and surveillance purposes. However, if rigidly attached to a mobile ground robot, such a camera adds a great deal of flexibility to the tracking system, alleviates the effect of slow ground robot dynamics and expands the field-of-view (FOV) of the robot by a huge factor (Rohmer et al. 2010). However, inclusion of additional degrees-of-freedom in the vision system means that local planning and control of the robot becomes much more pronounced and challenging as a problem.

In real world situations, the environment around the robot tracking the object is usually not empty, rather it is populated by obstacles. An algorithm that might perform really well in absence of obstacles, let's say in simulation, may not work at all in real life due to the ever-present risk of a collision during tracking. Hence, inclusion of dynamic obstacle avoidance using LiDAR (Light Detection And Ranging) scanner range data in the aforementioned optimization problem is proposed, to enable realistic tests in cluttered environments.

Thus, the contributions of this paper can be summarized as threefold: 1) Formulation of the moving object tracking/following problem as a single nonlinear optimization problem which models both camera and vehicle dynamics; 2) Dynamic obstacle avoidance for experiments in real life environments; 3) Proposal of a new method of UGV (Unmanned Ground Vehicle) teleoperation in coordinated UAV(Unmanned Aerial Vehicle)-UGV experiments. To the best of our knowledge, this is the first work that solves all the aforementioned problems.

The paper is organized as follows: In section 3, the conventions and symbols used throughout the paper are listed. The entire optimization problem is described in sections 4 and 5. Results of the algorithm on a simulated robot, across various different kinds of environments, are shown in section 6. Finally, the real-life test setup and corresponding observations are noted in section 7 and finally conclude the paper in section 8.

2 Related Work

A lot of work has been done on visual object tracking using mobile robots. In (Jung and Sukhatme 2004) and (Jung and Sukhatme 2010), the authors have proposed an algorithm for tracking objects by simultaneously estimating robot egomotion using image features, and detecting moving parts in

image using a particle filter. This leads to very robust tracking performance for objects in the ground plane. The algorithm is demonstrated on multiple types of vehicles. But, the methodology will not succeed when the object to be tracked is airborne, as the robot has no means of changing its orientation about the pitch and roll axes.

Pan-Tilt cameras have been used for tracking objects in many previous works. In (Park et al. 2011), the authors used a pan-tilt camera for visual servoing of a mobile robot towards a static target while moving on an inclined surface. In (Yu, Wu, and Lin 2010), the authors do track a moving object using a robot equipped with a PTZ camera. The robot motion command assumes that the object is visible in the ground plane, and no generalizations have been proposed for tracking a moving object not on the ground plane.

There are several works that implement tracking/following of ground robots using aerial vehicles (), or formation control (Kumar and Michael 2012) of aerial robots. In fact, the problem statements of autonomous quadcopter landing on a mobile robot (Falanga et al. 2017) and coordinated UAV-UGV exploration (Delmerico et al. 2017) (Michael et al. 2014) are extensively studied. However, in all these works, a UAV is the follower, and not a kinematically constrained UGV. This is because the 3 degrees of freedom of the UGV is not enough to track a UAV feasibly.

3 Notations and Assumptions

Throughout this paper, ${}^{w}(.)$ is considered as the world frame, ${}^{b}(.)$ as the body frame and ${}^{c}(.)$ as the camera frame. ${}^{n}_{m}T$ represents the 4x4 transformation matrix from frame m to frame n in homogeneous coordinate system. ${}^{m}P$ represents a position (usually of the target in this paper) in coordinate frame m, and the corresponding position in frame n is given by ${}^{n}P = {}^{n}_{m}T \times {}^{m}P$, where ${}^{m}P$ and P are both in homogeneous coordinates. ${}^{m'}(.)$ refers to frame m updated after an optimization step.

A right handed coordinate system has been used at all times. The direction the robot is facing, is considered as its x-axis, the direction perpendicular to the ground plane is considered as the z-axis and the y-axis is given by the direction to the left of the robot. This is demonstrated in figure 1. All counterclockwise rotations are taken as positive.

A right handed camera coordinate system is assumed. A pinhole camera model is assumed, and it is assumed that all lens distortions are taken care of after calibration and undistortion steps before the main algorithm.

The following simplifying assumptions have been made during this work:

- The ground plane is flat and the ground robot never leaves the ground plane.
- Reliable odometry information about the robot's position with respect to its starting position is available.
- Accurate extrinsic transformation matrices between sensor frames and intrinsic calibration of the camera parameters are available.

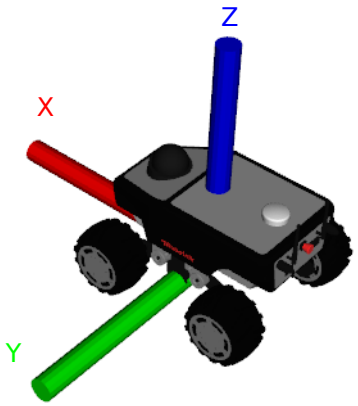

Figure 1: The coordinate system for the ground robot

- The focus of the PTZ camera always stays at the same point, regardless of orientation. Thus the rotation is always about the focus. This simplifies the problem a lot, by converting all pan-tilt motions to pure rotations about the fixed focus.

4 Optimization Problem Formulation

A ground robot which has a LiDAR scanner and a PTZ camera is considered, both of which are the data input sources to the optimization problem. The presence of an odometry source (which does not have to be error-free) is assumed. The odometry souce constantly gives the robot's orientation in ground plane, with respect to a fixed direction (considered as the north direction in this paper). All the errors caused due to the curvature of the Earth and the shift of the magnetic pole have been neglected. Most Inertial Measurement Units (IMUs) that have a magnetic compass, usually provide reliable data about the robot's orientation in the ground plane with respect to north direction, through hardware-based sensor data fusion. The frame of reference of the odometry algorithm is chosen to coincide with the body frame of the robot.

4.1 Data Preprocessing

Range Data A LiDAR scanner outputs data in the form of a 2-dimensional or 3-dimensional point cloud, which is a collection of data points in space. In this paper, only the 2-dimensional data of the LiDAR is used. The data points' distances and angles are measured from the centre of the sensor.The data points are first transfomed from LiDAR coordinate system to body coordinate system. This accounts for a pure translation transformation. The large number of points in the point cloud, if all directly added as residuals in the optimization problem, will slow it down quite a bit. Thus, the raw 2-dimensional point cloud is first segmented into meaningful line segments and arcs. The raw point cloud is first converted into Euclidean clusters following the method described in (Rusu 2010), using an implementation in the

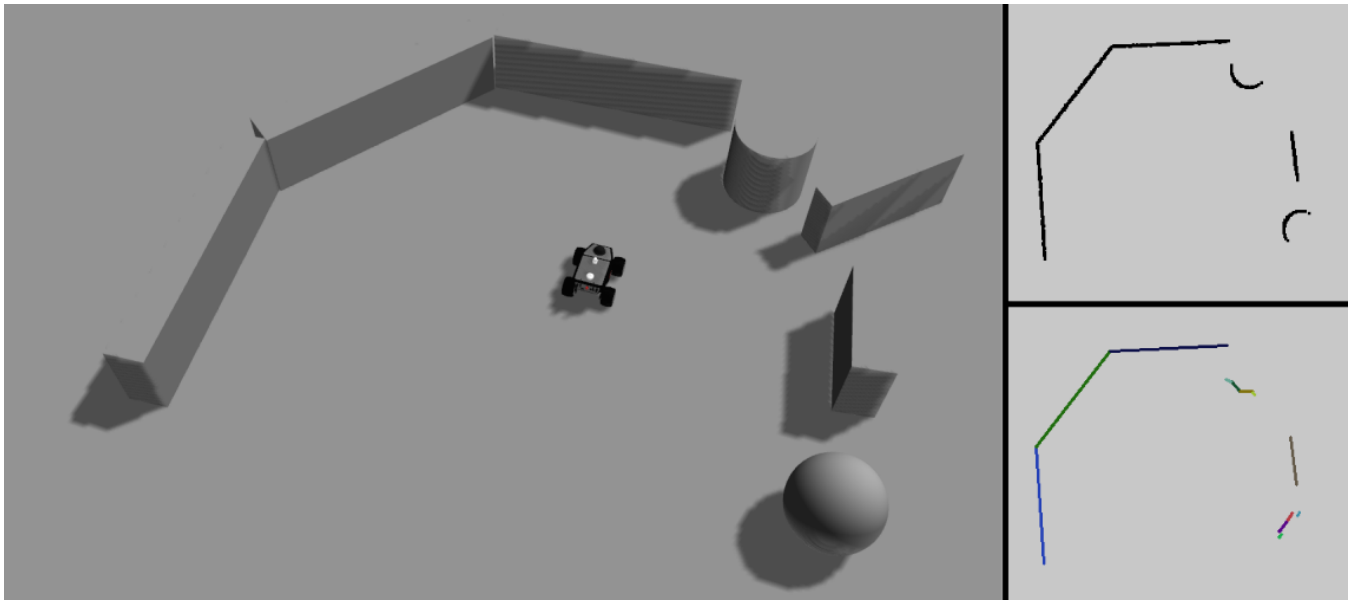

Figure 2: The raw 2D point cloud (top right) is segmented into line segments and clusters. Notice the arcs (which represent the clusters) leaving out sparse data points (bottom right).

Point Cloud Library (PCL) (Rusu and Cousins 2011). The algorithm returns a list of clusters of a minimum size, all of which are further processed using the steps described in Algorithm 1

Algorithm 1: Line segments and arcs extraction

Data: A set of N cluster points
Result: List of line segments and arcs
if *cluster size* $\leq minimumsegmentclustersize$ **then**
  Extract the arc parameters of the cluster;
Find the equation of line connecting the extreme points of the cluster;
Find the the point in the cluster farthest from the line and count inliers;
if *proportion of inliers* $\geq threshold$ **then**
  Store line segment;
else
  Create two new clusters separated by the farthest point from the line;
  Run the same algorithm recursively on the two new clusters;

The result of the segmentation algorithm is shown in figure 2. The extracted m line segments and n arcs are then passed as data to the optimization problem, in the form $[L_1, L_2, \ldots, L_m, C_1, C_2, \ldots, C_n]$.

Each L_i consists of the 2D coordinates of the start and end points of the line segment, in the body frame. Since the small-sized clusters are modeled as arcs of a circle, each C_i consists of a radial distance, width of the arc representing the cluster and its bearing angle in the body frame.

Image Data After the image data is received from the PTZ camera, it is rectified according to known distortion parameters. The image is then passed through the detection pipeline. The focus and zoom levels of the camera are kept as constant. The optimization problem is not sensitive to the type of object being tracked, but it is assumed that the size of the object is known and hence also its 3D coordinates camera frame, ${}^cP$ can be estimated from the image processing pipeline. For testing, a fiducial marker was used as a target, because not only does it facilitate high detection accuracy, but it also enables finding the 6DoF pose of the target, which will be beneficial for all future work. ArUco (Garrido-Jurado et al. 2014) library in OpenCV is chosen to perform the detection and pose estimation of the target. The marker size is chosen to be 6x6 and was generated using the library itself. The output of the detection step is the 3D coordinates of the target in camera coordinates. This front-end can easily be replaced for any type of target, be it a coloured ball, a human, or any other object whose size is known. Another noteworthy point is that the algorithm is not very sensitive to the accuracy of detection.

4.2 Optimization Problem Variables

The robot's target 2D position (dx,dy,$d\theta$) in current body frame, and the absolute pan-tilt angles (p,t), are chosen as the optimization variables in this problem. The pan and tilt angles are restricted to be in range $(-\frac{\pi}{2}, \frac{\pi}{2})$ and (0,π) respectively. All the residuals and Jacobians are calculated with respect to these 5 variables. The residuals used are described in the next few sections.

4.3 Range Data Residuals

Two separate type of residuals are calculated for line segments and arcs.

Line Segment Residual The target of this block of optimization is to incentivize the robot to stay away from obstacles. Positions close to obstacles should be heavily penalized, and also positions *behind* the line segment. By behind, it is meant that the point is on the opposite side of the line segment as the robot, but in the same cone subtended on the line by (0,0). Since a LiDAR scanner only calculates distances based on the first contact of the LASER beam with a surface, there is no way of knowing what is behind the obstacle. Hence, to avoid ambiguity, all points shadowed by the line segment are considered as inaccessible areas, which are heavily penalized. The error should ideally reduce really fast as the point moves away from the obstacle, so an exponential kernel is chosen. A plot of the error can be seen in figure 3. As evident from the plot, the function has a constant high value when the distance is negative, and decreases exponentially as distance increases. Here $distance$ is defined as the minimum euclidean distance between the point (dx, dy) and the line segment (see equation 1)

$$line\_distance = signed\_euclidean\_dist((dx, dy), median(linestart, lineend, proj(dx, dy))) \quad (1)$$

where $proj(dx, dy)$ is the perpendicular projection of the point (dx, dy) on the line segment. The median operation refers to finding the point that lies between the other two points. Signed euclidean distance denotes the euclidean distance with a sign being negative if (dx, dy) lies in the cone subtended by (0,0) on the line segment and on the opposite side of the line segment as (0,0), positive otherwise. Finally, the residual can be written using equation (2)

$$residual = \begin{cases} 10^7, & line\_distance \leq 0 \\ 10^7 \times e^{-5 \times line\_distance)} & \text{otherwise.} \end{cases} \quad (2)$$

Arc Residual Once again, it is desirable to be as far away from the arcs as possible. In the case of arcs, *behind* means points which are in the same sector as the arc but have more radial distance from the centre of the circle which the arc is part of. The same cost function as figure 3 is chosen, with a change in definition of distance, defined by equation (3)

$$arc\_distance = signed\_euclidean\_dist((dx, dy), arc) \quad (3)$$

Signed euclidean distance denotes regular euclidean distance, with a sign being negative if (dx, dy) lies in the sector subtended by the arc on (0,0) and on the opposite side of the arc as (0,0), positive otherwise. Finally, the residual can be written in the same fashion as equation 2, with $line\_distance$ being replaced by $arc\_distance$.

4.4 Image Reprojection Residual

The image reprojection residual minimizes the distance of the projection of the target in the PTZ camera image plane from the optical centre of the image. Based on the motion of the robot and PTZ camera, equations (4) through (8) can be written

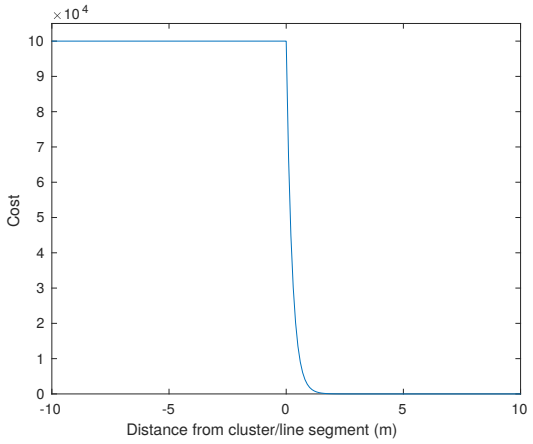

Figure 3: Cost function for line segment and arc data residuals

$$^{c'}P = {}^{c'}_{b'}T \times {}^{b'}P \quad (4)$$

$$^{b}P = {}^{b}_{c}T \times {}^{c}P \quad (5)$$

$$^{b'}P = {}^{b'}_{b}T \times {}^{b}P \quad (6)$$

$$^{b'}P = {}^{b'}_{b}T \times {}^{b}_{c}T \times {}^{c}P \quad (7)$$

$$^{c'}P = {}^{c'}_{b'}T \times {}^{b'}_{b}T \times {}^{b}_{c}T \times {}^{c}P \quad (8)$$

The final projection residual is written as

$$proj\_residual = \frac{1}{P_c.z} \times K \times {}^{c'}_{b'}T \times {}^{b'}_{b}T \times {}^{b}_{c}T \times {}^{c}P - \begin{bmatrix} c_x \\ c_y \\ 1.0 \\ 1.0 \end{bmatrix} \quad (9)$$

where $P_c.z$ represents the z component of P_c and K is the camera matrix for homogeneous coordinates, defined by equation (10)

$$K = \begin{bmatrix} f_x & 0.0 & c_x & 0.0 \\ 0.0 & f_y & c_y & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix} \quad (10)$$

where f_x and f_y represent focal lengths in x and y direction in pixels respectively, and (c_x,c_y) is the optical center of the image in pixel units.

The camera in body is transformation is characterized by a pure rotation from camera frame to the sensor link where the PTZ camera is attached to the robot, and a pure translation from the sensor link to the body frame. Although the translation part will vary from robot to robot, the matrix can be written as equation (11)

$$
{}^{b'}_{c'}T = \begin{bmatrix} -sin(p) & sin(t) \times cos(p) & cos(p) \times cos(t) & 0.19 \\ -cos(p) & -sin(t) \times sin(p) & -cos(t) \times sin(p) & 0.0 \\ 0.0 & -cos(t) & sin(t) & 0.395 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{11}
$$

The transformation of the body frame after motion in old body frame is defined in equation (12)

$$
{}^{b}_{b'}T = \begin{bmatrix} cos(d\theta) & -sin(d\theta) & 0.0 & dx \\ sin(d\theta) & cos(d\theta) & 0.0 & dy \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{12}
$$

4.5 Target Distance Residual

The target distance residual is to minimize the distance between the projection of the target on the ground plane and the robot. The residual can be denoted by equation (13)

$$
\begin{aligned} dist\_residual.x &= dx - {}^{b}P.x \\ dist\_residual.y &= dy - {}^{b}P.y \end{aligned} \tag{13}
$$

where ${}^{b}P.x$ and ${}^{b}P.y$ denote the x and y components of the target's position in body frame.

4.6 Robot Reachability Residual

The optimization is run at a constant rate of 10Hz (which is constrained by the data rate of the LiDAR sensor, and not by the processing time of the optimization). Thus, the robot has 0.1 seconds to execute the trajectory command sent to it. If the target (dx, dy, $d\theta$) values are set to be very large, the robot may not be able to reach the target pose in time. Thus, it is required to constrain the target pose command based on the robot's state (its current linear and angular velocities) and dynamics (maximum and minimum possible linear and angular accelerations). The target pose positions can be present only in the interior of a reachability curve, as shown in figure 4. Thus, a high residual value is assigned to all points outside the reachability curve, and 0 residual is assigned to all values inside the curve. To generate the reachability curve, linear velocities between minimum and maximum possible velocities are sampled, and 10 small integration steps with time period set to 0.01 seconds ($1/10^{th}$ of the time period of update) are performed.

5 Problem Solution & Robot Commands

The optimization problem is solved using the Ceres Solver nonlinear least squares library (Agarwal, Mierle, and Others). A sparse Cholesky solver is used for solving linear equations. Auto differentiation is used for calculating Jacobians for all the residuals, although Jacobians for most residuals were also manually evaluated on a MATLAB testbed to check the correctness of the error functions being used. As mentioned in section 4.6, the optimizer runs at a constant rate of 10Hz. A tuple of camera and LiDAR data is extracted using an ApproximateTime synchronizer in ROS. The data is preprocessed using steps described in section 4.1. The robot's current odometry estimate is extracted from the odometry source, as described in section 4. The data tuple and the odometry are both sent to the optimizer, which calculates an optimal value of the target state [dx,dy,$d\theta$,p,t] that minimizes the sum of squares of all residuals. The pan and tilt commands are sent to the camera directly. The robot motion commands are sent to a Timed Elastic Bands (TEB) local motion planner (Rösmann, Hoffmann, and Bertram 2015) implementation in ROS, which then gives a target command velocity (linear velocity and angular velocity) for the robot control. The motion control of the ground robot is handled using a PID control strategy.

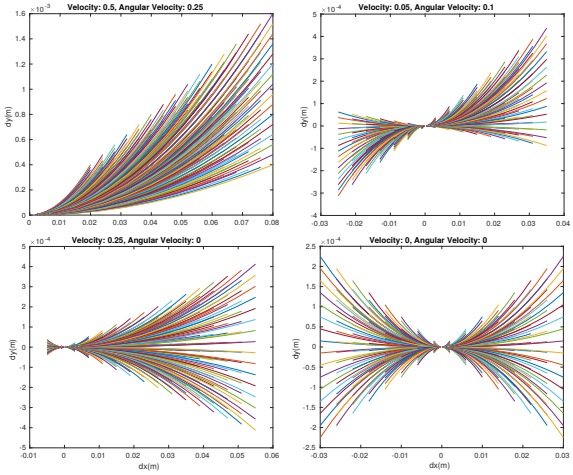

Figure 4: Reachability plots for different starting velocities

6 Simulation Results

The simulation environment used is a the open source ROS-based environment Gazebo. The robot used in simulation is the Summit XL by Robotnik Automation, which comes with a PTZ camera attached. The code is written in C++ and communicates with the simulator via ROS. The ground truth of the robot's orientation and position is used for feedback to the optimization problem. The default Summit control package is used for robot's control, and the PID gains are tuned experimentally. For experiments in simulation the following tests, incorporating realistic test conditions are defined:

- **Camera tracking target with no robot motion** (Figure 5): In this test, there's no kind of obstacle avoidance and no motion command is given, rather only the

Table 1: Tracking Performance

Target Speed(m/s)	Number of Runs	Successful Runs
1	6	6
2	7	7
5	10	9
10	8	4
15	4	1

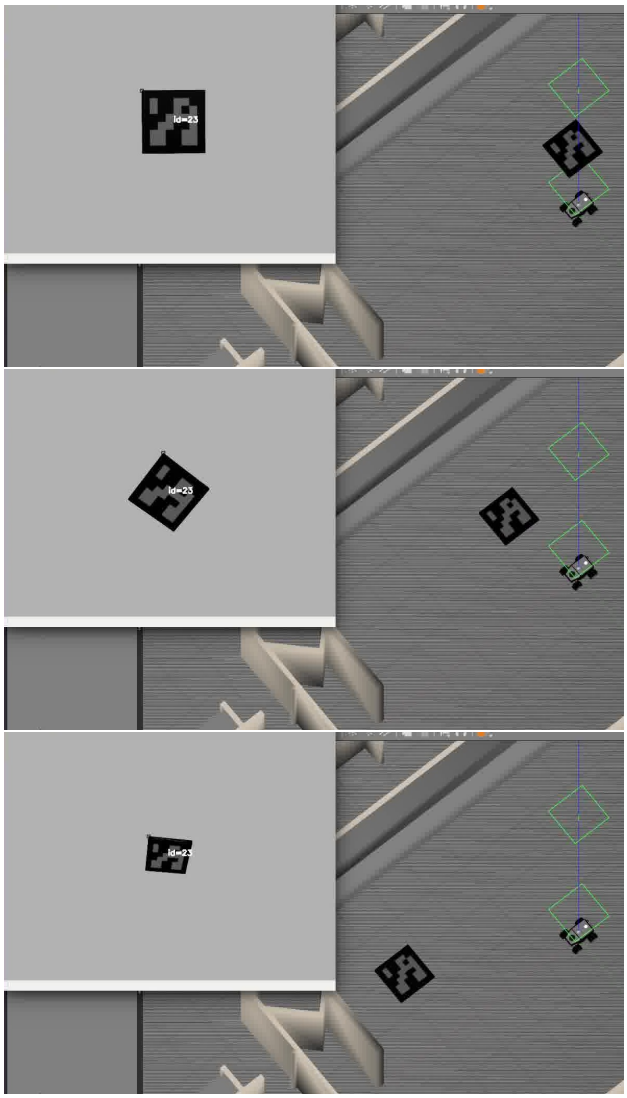

Figure 5: Tracking target with no robot motion. Top: t = 1s, Middle: t=3s, Bottom: t=7s

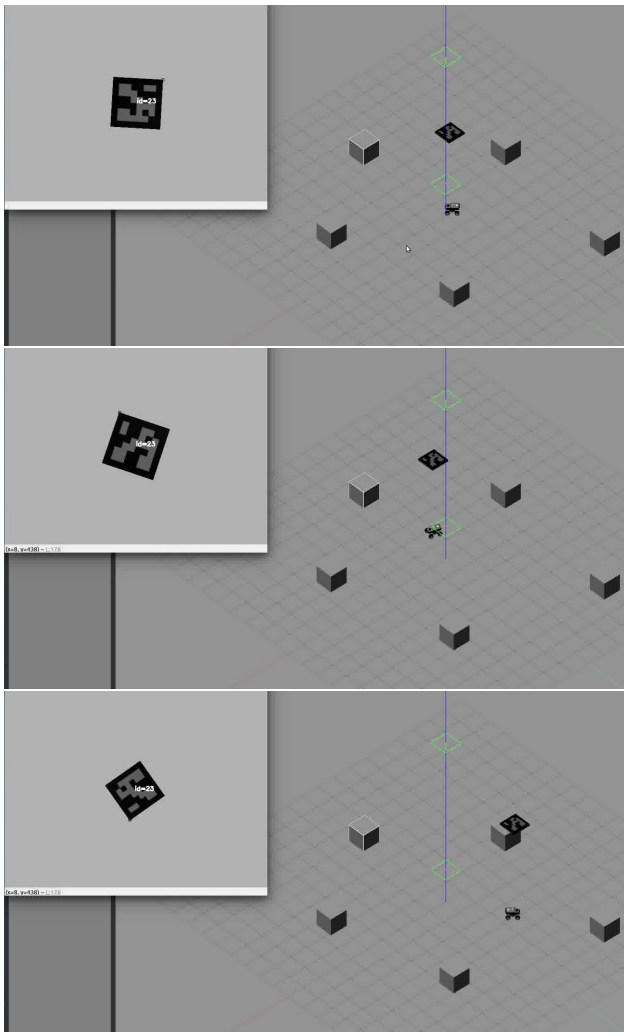

Figure 6: Following target in non-cluttered environment. Top: t = 0.5s, Middle: t=2.5s, Bottom: t=6s

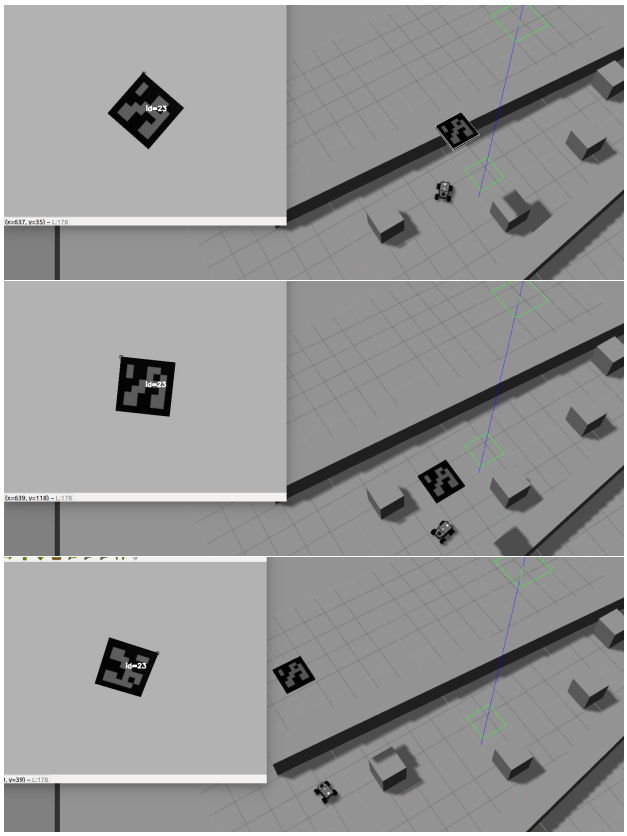

Figure 7: Following target in cluttered environment. Top: t = 2s, Middle: t=5s, Bottom: t=10s

image reprojection residual is used with the motion variables $(dx, dy, d\theta)$ set to zero. The target was given a random trajectory to traverse, which is generated using Bézier curves. The speed of the target's motion is gradually increased and results are noted. The camera is able to track targets moving up to the speed of 5m/s satisfactorily. (Table 1) The tracking failures usually occur when the pan link has to rotate by 180 degrees after reaching its limit on one end.

- **Robot following a target in a non-cluttered environment** (Figure 6): In this test, all the residuals are enabled, but it is performed in a sparse environment, so the obstacle avoidance residual does not contribute much. The target was moved at 5m/s. The robot was successfully able to track the target till the end of the trajectory in all 10 runs.
- **Robot following a target in a cluttered environment** (Figure 8): In this test, the same settings as the previous test were maintained, but the experiment was conducted in a relatively cluttered environment. The robot was successfully able to track the target till the end of the trajectory in 8 out of 10 total runs. The first failure was when the robot came too close to a wall while avoiding another obstacle and the target became occluded. The second failure was due to the pan link being forced to rotate 180 degrees.

7 Real-World Experimental Results

For real world tests, a Clearpath Husky robot was used. The PTZ camera used was the Axis P5514-E dome camera. The ArUco marker was mounted at the end of a pole and the robot was asked to follow the target. A Velodyne PUCK VLP-16 LiDAR was used for range data. Although the data comes in form of 16 rings, only the middle ring was used. A Core-i5 2.4GHz laptop running Ubuntu 16.04 with ROS Kinetic was used for the processing. In 3 separate runs, the robot successfully followed the target, while avoiding obstacles, in a mildly cluttered environment. The motion of the person holding the pole, while standing close to the robot, interfered with the results a bit, leading to an absurd rotation executed by the robot in one of the runs, but the algorithm successfully recovered the tracking after that particular failure.

8 Conclusions

Through this work, the capability of pan-tilt cameras as enabler of aerial target following by a mobile robot was demonstrated. Tracking at moderate speeds was satisfactory. Optimal performance of the visual servoing operation was demonstrated in both cluttered and non-cluttered environments.

Future Work The aim of this project is to enable cooperative exploration using UAVs and UGVs, with either acting as leader. An improvement that is currently required is handling of cases when the pan link rotates by a full 180 degrees. More tests with targets mounted on an actual flying vehicle will be performed in near future.

Figure 8: Real world tests

References

Agarwal, S.; Mierle, K.; and Others. Ceres solver. `http://ceres-solver.org`.

Delmerico, J.; Mueggler, E.; Nitsch, J.; and Scaramuzza, D. 2017. Active autonomous aerial exploration for ground robot path planning. *IEEE Robotics and Automation Letters* 2(2):664–671.

Falanga, D.; Zanchettin, A.; Simovic, A.; Delmerico, J.; and Scaramuzza, D. 2017. Vision-based autonomous quadrotor landing on a moving platform. In *Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics, Shanghai, China*, 11–13.

Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F. J.; and Marín-Jiménez, M. J. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* 47(6):2280–2292.

Jung, B., and Sukhatme, G. S. 2004. Detecting moving objects using a single camera on a mobile robot in an outdoor environment. In *International Conference on Intelligent Autonomous Systems*, 980–987.

Jung, B., and Sukhatme, G. S. 2010. Real-time motion tracking from a mobile robot. *International Journal of Social Robotics* 2(1):63–78.

Kumar, V., and Michael, N. 2012. Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research* 31(11):1279–1291.

Michael, N.; Shen, S.; Mohta, K.; Kumar, V.; Nagatani, K.; Okada, Y.; Kiribayashi, S.; Otake, K.; Yoshida, K.; Ohno, K.; et al. 2014. Collaborative mapping of an earthquake damaged building via ground and aerial robots. In *Field and Service Robotics*, 33–47. Springer.

Park, J.; Hwang, W.; Bahn, W.; Lee, C.-h.; Kim, T.-i.; Shaikh, M. M.; Kim, K.-s.; and Cho, D. 2011. Pan/tilt camera control for vision tracking system based on the robot motion and vision information. In *IFAC World Cong*, volume 44, 3165–3170.

Rohmer, E.; Yoshida, T.; Ohno, K.; Nagatani, K.; Tadokoro, S.; and Konayagi, E. 2010. Quince: A collaborative mobile robotic platform for rescue robots research and development. In *The Abstracts of the international conference on advanced mechatronics: toward evolutionary fusion of IT and mechatronics: ICAM 2010.5*, 225–230. The Japan Society of Mechanical Engineers.

Rösmann, C.; Hoffmann, F.; and Bertram, T. 2015. Planning of multiple robot trajectories in distinctive topologies. In *Mobile Robots (ECMR), 2015 European Conference on*, 1–6. IEEE.

Rusu, R. B., and Cousins, S. 2011. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, 1–4. IEEE.

Rusu, R. B. 2010. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz* 24(4):345–348.

Srinivasa, S. S.; Berenson, D.; Cakmak, M.; Collet, A.; Dogar, M. R.; Dragan, A. D.; Knepper, R. A.; Niemueller,

T.; Strabala, K.; Weghe, M. V.; et al. 2012. Herb 2.0: Lessons learned from developing a mobile manipulator for the home. *Proceedings of the IEEE* 100(8):2410–2428.

Yu, M.-S.; Wu, H.; and Lin, H.-Y. 2010. A visual surveillance system for mobile robot using omnidirectional and ptz cameras. In *SICE Annual Conference 2010, Proceedings of*, 37–42. IEEE.