Autonomous Vision-Based Target Tracking

Kavin Ravie
kravie@andrew.cmu.edu
Carnegie Mellon University

Naman Menezes
nmenezes@andrew.cmu.edu
Carnegie Mellon University

Saudamini Ghatge
sghatge@andrew.cmu.edu
Carnegie Mellon University

***Abstract*— This study introduces a system enabling a robotic manipulator, specifically a 7 DOF Franka Emika robot, to interact with dynamically moving objects displayed on a screen. The objects, Aruco markers, are tracked by an Intel Realsense RGBD camera mounted on the arm. Visual tracking control is achieved by aligning the axes of the marker pose with the robot pose. We utilize the Frankapy library, developed by IAM-Lab [1], to publish ROS messages for position control. Additionally, a verification system comprising a monocular camera employs classical computer vision approaches to determine if the object has been engaged by the robot. The verification node operates asynchronously, facilitating a closed-loop system. The proposed system is assessed in real-world scenarios, where a display screen with randomly moving Aruco markers is placed in front of the robot arm using a Pygame simulator. The system's performance was evaluated with five markers, achieving successful engagement with the objects 80% of the time, as verified by the verification system. The system is capable of handling multiple targets within the robot's workspace, presenting a solution for tracking and sequentially engaging with dynamically moving objects on a screen. Following is the github link to our repository AVTT.**

I. INTRODUCTION

Vision-based tracking of dynamic objects has found diverse applications in robotic systems such as Unmanned Aircraft Systems (UASs), Autonomous Mobile Robots (AMRs), and Mobile Manipulators. Vision serves as a primary perception source in many of these applications due to the small form factor of cameras, lightweight construction, and high image resolution. Target tracking has been extensively applied across industries, ranging from UAVs tracking moving objects [2] to robotic manipulators predicting the trajectory of objects in space and attempting to catch or hit them [3]. Moreover, researchers have explored applications in medical robotics, where manipulators assist in surgeries, necessitating visual servoing methods for precise operation at specific locations [4].

Our work introduces the use of a manipulator to track and engage randomly moving objects in its vicinity. The aim is to showcase the enhanced tracking capabilities of dexterous manipulators through visuo-motor control. We have utilized a laser to illustrate the ability of an n-DOF arm to precisely engage with desired objects by orienting itself accordingly. This study can serve as a benchmark for applications requiring such precise tracking to execute desired actions. Furthermore, this application can be extended to manipulators used in manufacturing industries, such as the vehicle manufacturing industry, where they are employed to assemble parts or apply paint, necessitating precise tracking of goal locations.

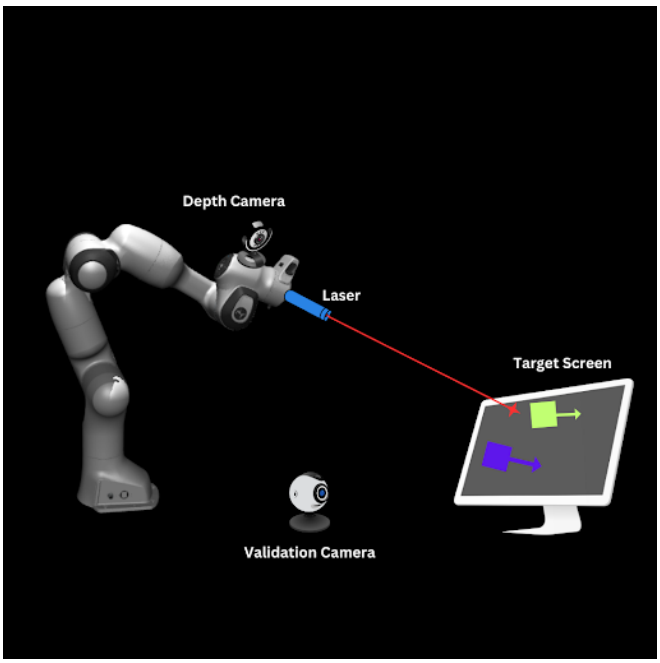

Fig. 1. Vision of our System Implementation

We have developed a rudimentary target-tracking system that showcases the capabilities of a 7-DOF arm to precisely track fast-moving targets using feedback from an RGBD camera mounted on its hand. We explored various methods for generating continuous motion of the arm using the Frankapy library [1].
Additionally, we are incorporating a secondary camera system to ensure accurate engagement with objects and to avoid false triggers **Fig 1**. Our experiments involved multiple targets at varying speeds, achieving an overall accuracy of 80%. Once all targets are hit, the arm returns to its home position and awaits new targets to spawn on the screen. Our contributions to this project can be summarized as follows:

1) Integration of the Frankapy library developed by IAM-Lab for robot position control
2) Development of a GUI that spawns Aruco markers randomly moving on the screen
3) Integration of a secondary monocular camera to verify whether targets have been hit or not

II. RELEVANT WORK

After conducting an extensive literature survey, we found no direct applications of our work. However, there has been extensive research in the field of object tracking using camera feeds. Our work can be easily extended to track random objects instead of aruco markers. Various vision-based tracking methodologies exist, ranging from classical computer vision approaches to learning-based methods. One such implementation using classical computer vision techniques is presented in [5] which employs a resilient feature detection strategy for tracking. This work has been extended to unmanned aerial system (UAS) applications. Another relevant study, [6] is similar to our proposal, with the distinction that the manipulator also aims to grasp the object. This work focuses on aligning a dynamic target frame to the onboard gripper using visual servoing. However, unlike our system, they have prior knowledge of the target pose.

III. METHODOLOGY

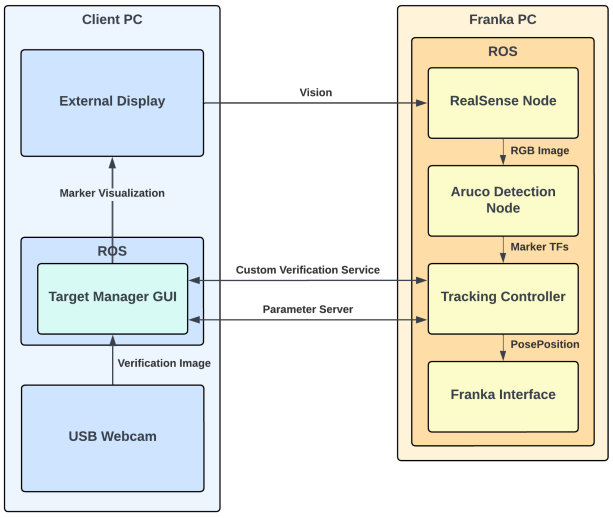

Fig. 2. System Architecture

To efficiently track targets and evaluate performance quality, we implemented two subsystems for target tracking and verification, along with a Pygame simulator to simulate moving objects in a virtual environment. Our workspace consists of the following: a) A monitor placed at a distance of _ meters from the base of the robot arm. b) The arm's home position is configured to point towards the screen, ensuring that the end-effector housing the laser points directly at the screen. c) A secondary verification camera placed on the side table, facing towards the screen. Refer to **Fig 3** for the complete system setup and **Fig 2** for our system architecture.

A. Graphical User Interface

We have developed a Python Pygame simulator capable of spawning Aruco Markers generated from the Original ArUco marker dictionary. These markers are spawned at random locations and move in random directions. Additionally, we have included functionality to adjust Aruco marker parameters such as size, velocity, and number of markers. This system receives input from the target tracking node and places the nearest marker to the current arm position on top of all other markers. In summary, the robot arm locates the nearest marker to its current position and tracks it. When the end-effector pose aligns with the marker pose, the target tracking node sends a flag to the Pygame simulator, which blanks out the screen and highlights only the active marker being tracked by the robot arm **Fig 4**.

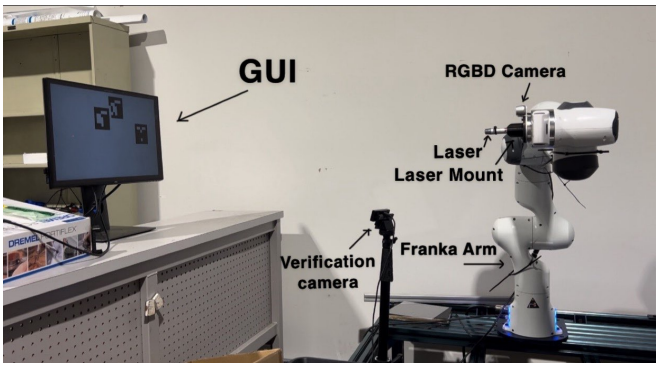

Fig. 3. System Setup

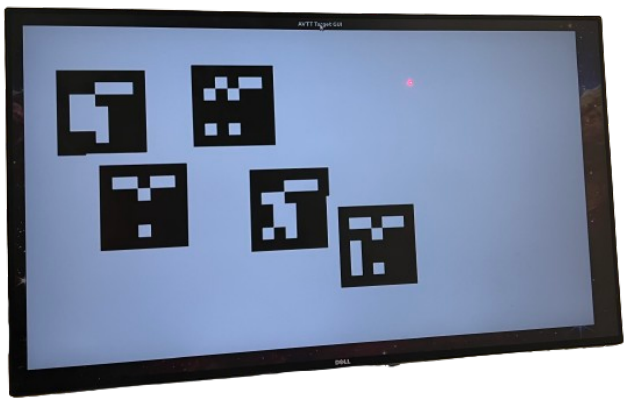

Fig. 4. GUI

B. Target Detection

We employ the ROS aruco_detect package to detect all the visible images from **ARUCO_ORIGINAL** dictionary of markers. The Marker sizes were set to 0.07m to emulate long-range scenarios. The image feed from the RGB camera of the RealSense camera (hand-eye calibrated) was passed to the detection package.

The detection package detects all visible markers in the scene belonging to the pre-defined dictionary and publishes their pose on the ROS TF server.

C. Target Tracking

This is the key contribution of work which is the development of a visual-servoing controller. Due to the limitations posed by the FrankaPy library, which limited us from directly commanding joint torques, the prior controller strategy which we had validated in MuJoCo simulation [7] and showcased in our Mid-term evaluation Presentation could not be used. Instead, we had to rely on the trajectory interface provided by Franka-Interface using which we published Pose-Position commands in the task space.

The Algortihm:

1) Get all the latest marker poses from the TF server (we use an async loop for this lookup at 100Hz)
2) **Controller Loop (async @ 100Hz)**
   a) Select the optimal target (if not already selected) based on Euclidean distance metric
   b) Track this object until engaged
   c) Call verification service provided by the manager for external validation when we are sufficiently close to the target position
   d) Continue tracking the same ID if validation was unsuccessful or else re-select the optimal target and remove the engaged target from our active list.
   e) Repeat all the above steps until we have engaged all the targets (defined by the /target_manager/num_targets ROS parameter provided by the target manager)

   For tracking the target we manually align the TV/Display to be planar to the XZ plane. This enables us to define a virtual offset target on the XY plane centred at the base of the manipulator. So any and all motion would be confined to this plane and ideally, there should not be any motion in the Y-direction (for safety).

   We keep the orientation of the EEF fixed as with complete 6DOF control, we faced issues with the manipulator taking aggressive and risky motions.

IV. EVALUATION

To evaluate our system, we tested it by spawning 5 targets on the screen. We used a Red color Laser to engage with the virtual objects. Since we are also integrating a verification system to verify whether the target was engaged or not, we could accurately verify the working of our system. Overall our system was able to track and verify object engagement 80% of the time successfully. Since our reset condition is defined as restarting the target manager and the controller nodes, which means generating new targets at any random position, the arm successfully performs the whole operation.

From the plot above we can note the good tracking performance achieved by our system. With joint torque control strategies, this error can be further reduced as we would be using the most optimal joint state changes using the jacobian.

This leads us to a major limitation of our present implementation which arises from the fact the EEF has to traverse the same distance as the target due to the pure position control approach. Using Orientation should remove this drawback as now the robot could simply change the orientation of the EEF to make the EEF-Z/Laser axis intersect with the target.

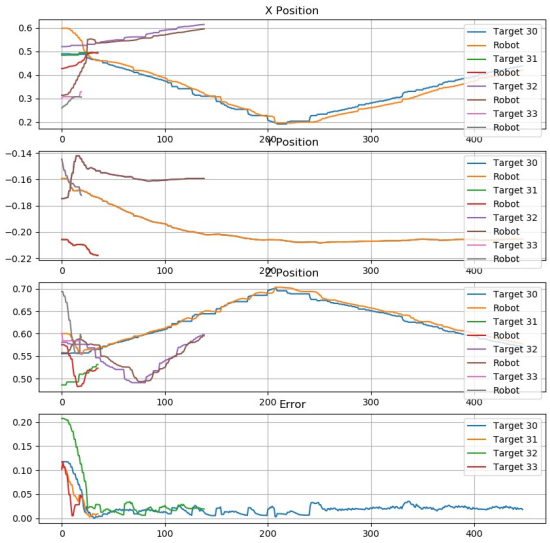

Fig. 5. Position Error

V. CHALLENGES

There were many challenges faced during this project, the most significant ones are listed below:

1) **Absence of Torque Control:** While the frankapy API has extensive coverage of different control mechanisms, there was a significant challenge encountered when the torque control was not available for the franka arm. Our initial proposal aimed to use torque control of the joints to quickly reposition the end effector of the arm. To overcome this problem the franka python API was replaced with a ROS message based controller in order to implement position control of the arm. This allowed for much smoother dynamic motion compared to the Franka gotoPose functions. By adjusting the stiffness.
2) **Full Pose Control:** Commanding full-6DOF poses to the manipulator using the trajectory interface led to the issues mentioned in the previous sections. Having orientation control would greatly reduce the time to intercept each target.
3) **Distributed Systems Integration:** The software architecture was the most complicated part of our whole project. These were:
   - Generating metric-accurate Aruco markers on a digital display.
   - Multi-threading-related issues stemming from our parallelized software architecture

- Syncing the Manager and the Controller states using minimal data flow.

VI. FUTURE WORK

The system exceeded the functional requirements that were set up for it. There are a few possible directions in which future work can improve the system. Firstly, the system does not implement a good way to do orientation tracking especially in the general case, there is an offset that is caused by the position of the laser which may not always be aligned to the end effector position. In order to overcome this variability, it is possible to create a calibration sequence that estimates the end effector position. Another possible direction of future work is to implement position control on the laser position, currently no feedback of the laser position is used in the pipeline of positioning the arm, this was done as the time that the laser was being 'fired' was minimized. However, in an ideal case, the few frames that the laser is visible should be used to correct minor errors in the end effector position to improve the accuracy of the arm.

VII. CONCLUSION

This report demonstrates a methodology to autonomously track and engage multiple moving targets with a laser mounted on a Franka Emika arm. A robust methodolgy is used to switch between appropriate targets and a novel verification system is proposed to ensure proper engagement of the targets. We demonstrate the ability of successfully tracking and engaging five unique targets moving with unique velocities in distinct directions. We open source our code to run smooth detection and tracking of the targets and also the target manager GUI which can be used to evaluate the performance of the system. We hope that our work will be useful for anyone doing visuo-motor control using a manipulator.

VIII. ACKNOWLEDGEMENTS

The authors would like to express their immense gratitude to Prof. Oliver Kroemer for his guidance with the project. Additionally we are grateful to our TA's Abhinav Gupta and Vibhakar Mohta for their help in implementing our project. Finally, we would like to acknowledge our fellow MRSD students for all the help extended to us.

REFERENCES

[1] https://github.com/iamlab-cmu/frankapy
[2] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for uavs: Current developments and trends," Journal of Intelligent & Robotic Systems, vol. 87, no. 1, pp. 141–168, 2017.
[3] S. -T. Kao, Y. Wang and M. -T. Ho, "Ball catching with omni-directional wheeled mobile robot and active stereo vision," 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), Edinburgh, UK, 2017, pp. 1073-1080, doi: 10.1109/ISIE.2017.8001395. keywords: Cameras;Mobile robots;Trajectory;Projectiles;Stereo vision;Visualization;Visual Servoing;Wheeled Robots;Visual Tracking
[4] https://mrsdprojects.ri.cmu.edu/2024teame/
[5] P. Karmokar, K. Dhal, W. J. Beksi and A. Chakravarthy, "Vision-Based Guidance for Tracking Dynamic Objects," 2021 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 2021, pp. 1106-1115, doi: 10.1109/ICUAS51884.2021.9476712. keywords: Atmospheric modeling;Cameras;Aircraft;Open source software,
[6] Arora, P., Papachristos, C. (2020). Mobile Manipulator Robot Visual Servoing and Guidance for Dynamic Target Grasping. In: Bebis, G., et al. Advances in Visual Computing. ISVC 2020. Lecture Notes in Computer Science(), vol 12510. Springer, Cham. https://doi.org/10.1007/978-3-030-64559-5_17
[7] https://mujoco.readthedocs.io/en/stable/overview.html