Optimization-based Estimation of Environment Obstacles from Human Demonstration Using Control Lyapunov Function and Control Barrier Functions

MEAM 5170 Final Project, Fall 2023

Shafagh Keyvanian
14493437

Ho Jin Choi
24661677

Minku Kim
27107660

***Abstract*—This project tackles the challenge of enabling a sensor-free robot to identify obstacles based on expert demonstrations. The proposed methodology utilizes the Control Lyapunov Function (CLF), Control Barrier Function (CBF), and the CLF-CBF-QP optimization algorithm to acquire knowledge about obstacle position and size within an environment. The method introduces two contributions: (a) modeling human behavior around obstacles by probabilistically learning parameters α and γ of the CLF-CBF-QP formulation from expert demonstrations, and (b) learning the position and radius of obstacles (x_{obs} and r_{obs}) from all human demonstrations. The project assumes spherical objects in the environment and stable target trajectories from expert demonstrations.**

I. Introduction

Navigation through complex environments with multiple obstacles requires a sophisticated trajectory-planning approach. Human planners, skilled in crafting smooth trajectories, take into account various factors such as distance, effort, time, and jerk. The intricate and subjective nature of human trajectory planning highlights the necessity for an optimized and adaptable imitation learning model. Our project addresses this challenge by capturing the subjective essence of human decision-making, with the goal of providing robots with both adaptability and collision avoidance capabilities.

Our project contributes by modeling essential parameters and learning obstacle properties, such as position and radius. This approach enables the robot to identify and navigate around multiple obstacles. Harnessing the capabilities of the Control Lyapunov Function (CLF) and Control Barrier Function (CBF), in conjunction with the CLF-CBF-QP formulation, our methodology establishes a framework for obstacle estimation without relying on perception, which can be used for learning human preference or non-physical obstacles.

A. GMM and GMR Formulation

The applications of the Gaussian Mixture Model (GMM) and Gaussian Mixture Regression (GMR) [1, 2] are employed for modeling human-like trajectories around obstacles. GMM represents a probabilistic model by expressing the scatteredness of the data points as a weighted summation of Gaussian components. Using this approach for modeling the acquired data, the discrepancies of the demonstrations can be accounted for as the covariance of the Gaussian kernels in GMM, and the most optimal trajectorycan be derived out of the given demonstrations using GMR. In GMR, as opposed to GMM where all dimensions of the dataset is considered to be probabilistic, a subset of data dimensions are considered deterministic to calculate what would be the most probable (expected) value for the probabilistic dimensions. This means that after fitting a GMM over the collected dataset for demonstrations in two dimensions of X and Y, by knowing the value of X, what would be the most expected Y value achieved from the GMM.

The probability density function of GMM of a 2-D dataset would be written as the following:

$$P\Big(\begin{bmatrix}x\\y\end{bmatrix}\Big) = \sum_{k=1}^{K} \pi_k \mathcal{N}\Big(\begin{bmatrix}x\\y\end{bmatrix}, \begin{bmatrix}\mu_{k,x}\\\mu_{k,y}\end{bmatrix}, \begin{bmatrix}\Sigma_{k,x} & \Sigma_{k,xy}\\\Sigma_{k,yx} & \Sigma_{k,y}\end{bmatrix}\Big)$$

subject to:

$$\sum_{k=1}^{K} \pi_k = 1$$

where K is the number of Gaussian clusters in the GMM, π_k is the weight of the k^{th} Gaussian cluster, μ_k is the associated mean vector and Σ_k is the associated positive-semi-definite covariance matrix.

After fitting the GMM onto the dataset, we are left with a weight (π), mean vector (μ), and a covariance matrix (Σ) for each of the Gaussian clusters of the GMM. Then the most probable y values, knowing x values of the dataset, can be derived using the following equations of GMR:

$$y(x) = \sum_{k=1}^{K} \beta_k(\mu_{k,y} + \Sigma_{k,yx}(\Sigma_{k,x})^{-1}(x - \mu_{k,x}))$$

subject to:

$$\beta_k = \frac{\pi_k \mathcal{N}(x, \mu_{x,k}, \Sigma_{x,k})}{\sum_{i=1}^{K} \pi_i \mathcal{N}(x, \mu_{x,i}, \Sigma_{x,i})}$$

These techniques enable the project to learn and replicate human-like trajectories around obstacles from expert demonstrations, providing a foundation for subsequent steps in the methodology.

B. CLF-CBF-QP

Control Lyapunov Function (CLF) and Control Barrier Function (CBF) establish a theoretical basis for devising control strategies that guarantee stability and obstacle avoidance in robotic systems [3]. The CLF-CBF-QP formulation integrates these functions into a quadratic program, generating instantaneous control inputs for each state to ensure stability

and safety. In this project, the emphasis is on utilizing this control methodology to empower a robot to recognize obstacles through expert demonstrations and navigate an environment using predictions about obstacles.

$$u(x) = \arg\min_{u,\delta} \ \frac{1}{2}u^T H(x)u + p\delta^2 \qquad (\text{CLF} - \text{CBF} - \text{QP})$$
$$\text{s.t.} \ \ L_f V(x) + L_g V(x)u \leq -\gamma(V(x)) + \delta$$
$$L_f h(x) + L_g h(x)u \geq -\alpha(h(x))$$

II. Methodology

A. Problem Formulation:

Problem Assumptions:

- Expert trajectories lead to a stable target without collision, following the CLF-CBF-QP formulation.
- Obstacle shapes can be decomposed into multiple spheres, creating multiple CBF constraints.
- No dynamic constraint: $u_k = x_{k+1} - x_k$ ($x \in \mathbb{R}^m$) for $k = 1, 2, ..., T-1$ where T is the last time step of a trajectory
- Lyapunov function: $V(x) = (x_{target} - x)^2$ where x_{target} is the target state.
- Barrier function: $h(x) = (x - x_{obs})^2 - r_{obs}^2$ where x_{obs} and r_{obs} are the obstacle's position and the radius.
- α and γ functions are positive linear functions: $\alpha(h(x)) = \alpha h(x)$ and $\gamma(V(x)) = \gamma V(x)$
- $H(x) = I$

The CLF-CBF-QP formulation is utilized to estimate the velocity input, $u_{k,\text{est}}$, which is then compared with the demonstrated velocities, u_k.

$$u_{k,est} = \arg\min_{u,\delta} \ \|u\|_2^2 + p\delta^2$$
$$\text{s.t.} \ \ 2(x_{target} - x_k)u \leq -\gamma(x_{target} - x_k)^2 + \delta$$
$$2(x_k - x_{obs,i})u \geq -\alpha((x_k - x_{obs,i})^2 - r_{obs,i}^2)$$

for $i = 1, 2, ..., N$ where N is the number of obstacles. Here, differentiable QP is employed to obtain gradients with respect to the obstacle's position and radius [4]. The cost or loss function is defined as

$$\text{Loss: } \frac{1}{T-1}\sum_{k=1}^{T-1}\left(1 - \frac{u_k \cdot u_{k,est}}{\|u_k\|_2\|u_{k,est}\|_2}\right)$$

Since CLF-CBF-QP can be formulated for each state and the corresponding target, the learning process of obstacles does not have to rely on a single trajectory but can be derived from multiple trajectories with different targets in the same environment. Therefore, $T-1$ is adjusted to represent the number of data points. The rationale behind the loss function is to position obstacles in a way that generates repulsive forces, altering the direction of the vector field. To capture this, a cosine similarity loss is employed instead of a mean squared error. Additionally, given the knowledge that demonstrated trajectories avoid collisions with obstacles, soft constraints are incorporated into the loss as follows:

$$-ReLU(-(\|x_k - x_{obs,i}\|_2 - r_{obs,i}))$$

for $i = 1, 2, ..., N$ and $k = 1, 2, ..., T$. While manual calculation of gradients is possible, the project leverages the auto-differentiation feature of PyTorch for this purpose.

B. Project Steps:

Step 1: Learning α and γ: Modeling Human-like Obstacle Avoidance Behavior

1) Employ GMM-GMR to model human-like trajectories around obstacles based on expert demonstrations.
2) Apply the CLF-CBF-QP formula with initial assumptions for α and γ to estimate velocities (u).
3) Compute the loss by comparing estimated velocities with known velocities.
4) Iteratively refine α and γ until convergence.

Step 2: Obstacle Estimation

1) Utilize all demonstration data (expert and non-expert) to extract position (x_k) and velocity (u_k) information.
2) Implement the CBF-CLF-QP optimization algorithm to estimate u_k while ensuring collision avoidance.
3) Formulate a loss function by comparing estimated velocities with actual velocities.
4) Adjust x_{obs} and r_{obs}, and iterate until convergence.

This two-step methodology allows the robot to identify obstacles, emulate human-like behavior, and estimate obstacle properties either independently or in conjunction with sensors. It offers an approach for robot navigation in cluttered environments, showcasing the potential for effective obstacle recognition and avoidance.

C. Evaluation Metric for 2D case

A robust criterion for evaluating the accuracy of estimated obstacles compared to real obstacles is the Intersection over Union (IoU), which quantifies the degree of overlap between estimated and real obstacles. Another metric is the Intersection over Ground Truth (IoGT), a conservative measure allowing for more estimated obstacles than the actual ones. While measuring these metrics is straightforward when the number of obstacles is known, challenges arise when their numbers are unknown or establishing direct correspondence is difficult. For the 2D case, both predicted and real obstacles are rendered into pixel masks, and the number of pixels is calculated to determine intersection or union. Given A_{obs} and $A_{\text{obs,est}}$ as the 2D areas occupied by real and estimated obstacles, the evaluated measures include:

$$\text{IoU: } \frac{A_{obs} \cap A_{obs,est}}{A_{obs} \cup A_{obs,est}} \quad , \quad \text{IoGT: } \frac{A_{obs}}{A_{obs} \cup A_{obs,est}}$$

III. Data Construction

To generate expert trajectories, we design three distinct scenarios, each characterized by unique behaviors and corresponding α and γ parameters.

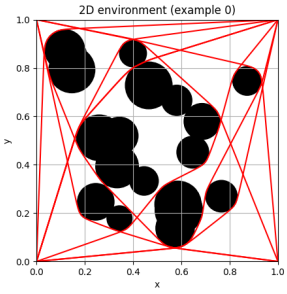

Fig. 1. Example of a 2D simulated demonstration (Red: demonstrated trajectories, Black: obstacles)

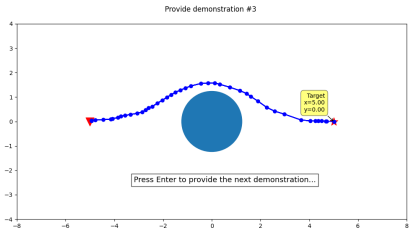

Fig. 2. Example of a 3D demonstration (Red: demonstrated trajectories, Black: table, Coordinate frame: target, Others: obstacles)

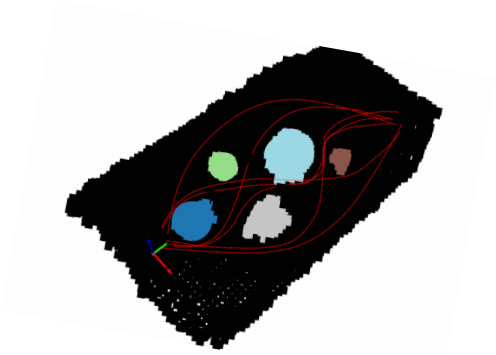

Fig. 3. Example of a 3D demonstration (Red: demonstrated trajectories, Black: table, Coordinate frame: target, Others: obstacles)

A. 2D optimization-based simulation

We employ non-linear optimization-based trajectory generation with input constraints and cost considerations for velocity and acceleration using the Drake framework. Each trajectory generation is initialized with randomly placed obstacles, two distinct targets, and sampled trajectories that are unique for the same environment, as illustrated in Figure 1. Given the non-linear nature of the optimization, it produces several non-optimal trajectories, serving as suboptimal demonstrations.

B. 2D human-like demonstrations with mouse

We construct a 2D environment featuring obstacles of varying radii and generate demonstrations using a mouse interface, as depicted in Figure 2.

C. 2D and 3D real human demonstrations

We position sphere-like obstacles on a table, which is captured by an RGBD camera and processed using the SLAM algorithm RTAB-Map [5] to obtain point clouds representing the table, as illustrated in Figure 3. Subsequently, we apply RANSAC to identify both the table and the obstacles. To capture demonstration trajectories, we utilize the Optitrack motion tracking system, calibrated with the RGBD camera, to track the position in 3D. We also assess its relative position concerning the target.

IV. Results

A. Learning α and γ from expert demonstrations

1) 2D human-like demonstrations with mouse: Ten mouse trials are conducted for obstacles of ten different radii. In each trial, three demonstrations are presented, and the GMM-GMR algorithm is employed to derive the regression function, as depicted in Figure 4. Subsequently, the data points of the regression function are utilized to learn the α and γ constants, with the obstacle position and radius being known. The learned values for α and γ for each trial are documented in Table 1.
2) 2D real human demonstrations: The actual demonstrations utilized for learning α and γ are derived from two planar trials, each featuring a single obstacle. Figure 5 illustrates the three demonstrations for each trial, along with the GMM kernels and the GMR function. The resulting α and γ values for the real human demonstrations are reported in Table 2.

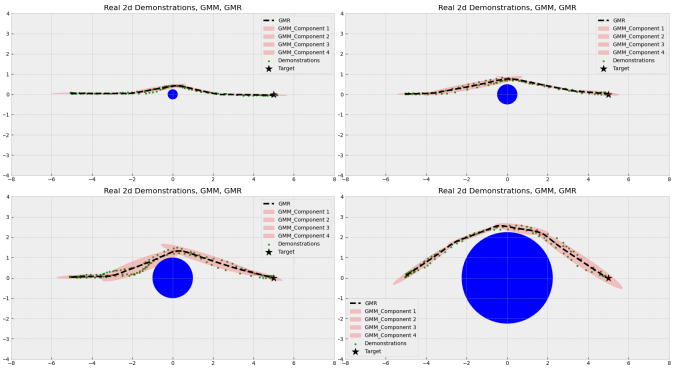

Fig. 4. GMM-GMR results for $r_{obs} = 0.25, 0.5, 1, 2.25$ (top-left to bottom-right)

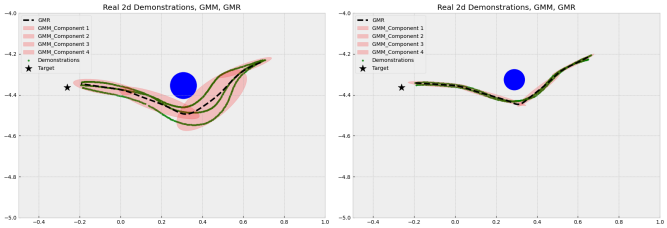

Fig. 5. GMM-GMR results for $r_{obs} = 0.0.0657, 0.0525$

B. Learning obstacles properties from learned α and γ

We leverage the α and γ parameters that were obtained from the demonstrations in the previous section to acquire the position and radius of the obstacle(s) in the real 2D demonstration environment. In Figure 6, we present a visual representation of the 2D environment, illustrating the visualized results of the learned obstacle positions and radii alongside the evolving loss and metric across iterations. The α and γ parameters used in each 2D environment are α = [0.1751, 0.2390, 0.0178] and γ = [0.1416, 0.1003, 0.2811] respectively.

TABLE I
α AND γ FOR 2D MOUSE DEMONSTRATIONS

	r_{obs}	α	γ	Loss
Row 1	0.25	0.121	0.0163	0.0402
Row 2	0.5	0.0992	0.0221	0.036688
Row 3	0.75	0.743	0.01	0.06389
Row 4	1	0.0985	0.0212	0.044818
Row 5	1.25	0.0907	0.03	0.059265
Row 6	1.5	0.0754	0.0299	0.044088
Row 7	1.75	0.0682	0.0221	0.066847
Row 8	2	0.0773	0.029	0.080959
Row 9	2.25	0.0741	0.0269	0.059892

TABLE II
α AND γ FOR 2D REAL DEMONSTRATIONS

	r_{obs}	α	γ	Loss
Row 1	0.0525	0.239	0.1003	0.048046
Row 2	0.0666	0.1751	0.1416	0.048615

V. DISCUSSION AND FUTURE WORK

From the results presented earlier in this report, the following observations were made:

1) **Variability of** α: As evident in Table 1, the α variable exhibits variation with changes in obstacle size. This suggests the need for formulating α as a function of r_{obs} or alternatively as a function of the barrier function $h(x)$ rather than a constant value, as also discussed in [3]. Additionally, there's an intuitive belief that α might be a function of $\dot{h}(x)$, considering that individuals tend to exercise more caution in avoiding obstacles when moving at higher speeds.
2) **Estimation of Additional Obstacles**: In scenarios with only one obstacle in the environment, the algorithm tends to estimate additional obstacles to justify curved trajectories close to the start point and target. This behavior may be attributed to the fact that during demonstrations, the demonstrator aimed to reach the target in a horizontal orientation. To address this, further experiments will be designed to analyze and understand this behavior or dynamic model. One potential solution for planar experiments is to model human dynamics while considering the yaw angle of the motion as an additional state, akin to a unicycle.

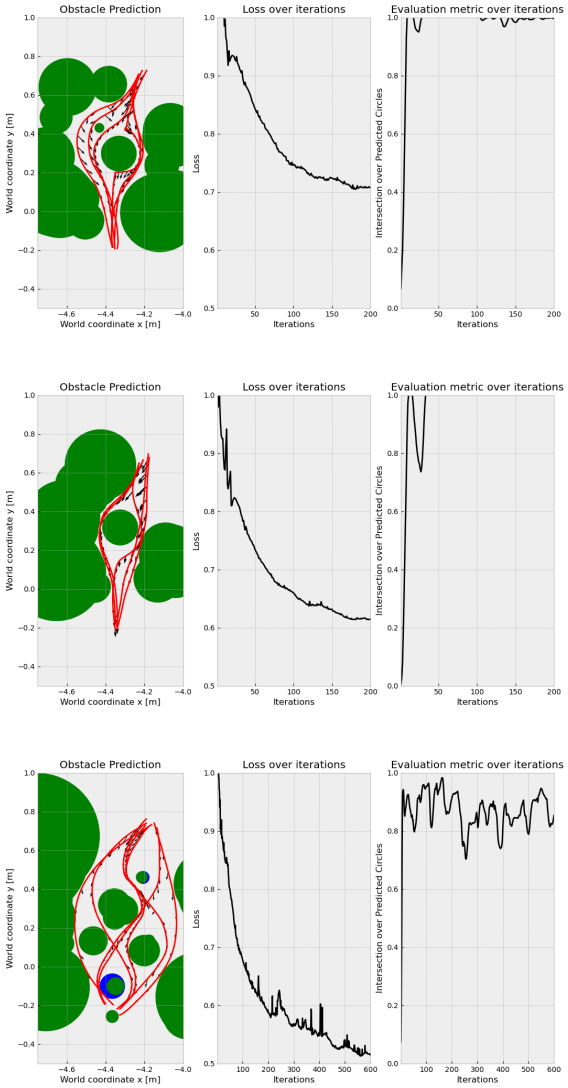

Fig. 6. Plot of predicted obstacle(s) and Loss / Metric across iterations

REFERENCES

[1] Emmanuel Pignat and Sylvain Calinon. "Bayesian Gaussian mixture model for robotic policy imitation". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4452–4458.

[2] S Mohammad Khansari-Zadeh and Aude Billard. "Learning stable nonlinear dynamical systems with gaussian mixture models". In: *IEEE Transactions on Robotics* 27.5 (2011), pp. 943–957.

[3] A. Ames et al. "Control Barrier Functions: Theory and Applications". In: *2019 18th European Control Conference (ECC)* (2019), pp. 3420–3431. URL: https://api.semanticscholar.org/CorpusID:85530121.

[4] Brandon Amos and J. Zico Kolter. "OptNet: Differentiable Optimization as a Layer in Neural Networks". In: *International Conference on Machine Learning*. 2017. URL: https://api.semanticscholar.org/CorpusID:1791473.

[5] Mathieu Labbé. "RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation". In: 2018. URL: https://api.semanticscholar.org/CorpusID:209527025.