DISSERTATIONES
TECHNOLOGIAE
UNIVERSITATIS
TARTUENSIS
73

HOUMAN MASNAVI

Visibility Aware Navigation



DISSERTATIONES TECHNOLOGIAE UNIVERSITATIS TARTUENSIS
73

DISSERTATIONES TECHNOLOGIAE UNIVERSITATIS TARTUENSIS
73

HOUMAN MASNAVI

Visibility Aware Navigation

UNIVERSITY OF TARTU
Press
1632

Institute of Technology, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Engineering and Technology on 07.07.2023 by the Council of the Institute of Technology, University of Tartu.

Supervisors

Prof. Karl Kruusamäe
University of Tartu
Tartu, Estonia

Prof. Arun Kumar Singh
University of Tartu
Tartu, Estonia

Opponent

Prof. Roel Stephan Pieters
Hervanta Campus
Tampere University
Tampere, Finland

The public defense will take place on 28.08.2023 at 14:15 in Nooruse 1-121.

The publication of this dissertation was financed by the Institute of Technology, University of Tartu.

Copyright © 2023 by Houman Masnavi

ISSN 2228-0855 (print)
ISBN 978-9916-27-255-8 (print)
ISSN 2806-2620 (pdf)
ISBN 978-9916-27-256-5 (pdf)

University of Tartu Press
www.tyk.ee

ABSTRACT

Motion planning is one of the core components of robotics and automation that involves determining a sequence of movements or actions to achieve a specific goal while satisfying various constraints. This thesis deals with a variant of motion planning problem called Visibility Aware Navigation (VAN), where the goal is to move between two points or track a reference trajectory while keeping certain objects in the robot's field of view at all times. VAN finds applications in different areas such as target tracking, aerial cinematography, marker-based localization, and cooperative warehouse inventory. In target tracking and aerial cinematography, it ensures clear visibility of the target while avoiding obstacles. In state estimation and localization, it improves accuracy and robustness by planning paths that maintain the visibility of markers like fiducial markers or beacons. VAN can also lay the foundation for cooperation between Unmanned Ground Vehicle (UGV) with more computing and sensing resources and a small-sized Unmanned Aerial Vehicle (UAV). Visibility Aware Navigation (VAN) can help a UGV to guide a UAV as long as their line of sight is not occluded.

However, there are two core challenges associated with VAN. First, robust algorithms should be developed that can handle complex constraints such as visibility, even in highly cluttered and dynamic environments. More specifically, the occlusions stemming from the obstacles that obstruct the robot's field of view need to be avoided during navigation. Second, the solutions should perform in real-time on embedded devices that can be carried by small robots like drones. This thesis proposes novel motion planning algorithms in the context of visibility-aware navigation that can handle complex environments with static and dynamic obstacles, while also achieving real-time performance on resource-constrained devices. The algorithms are extensively tested in both simulation and real-world environments and benchmarked against different state-of-the-art approaches. The proposed approaches outperform the state-of-the-art in different metrics such as occlusion time, control effort, and computation time. Furthermore, the algorithms run in real-time on embedded devices such as NVIDIA Jetson TX2 and Xavier.

CONTENTS

List of original publications **16**

1. Introduction **18**

2. Contributions and Scope of the Thesis **20**

3. Background **21**
3.1. Motion Planning 21
3.1.1. Sampling-Based 22
3.1.2. Search-based 23
3.1.3. Optimization-based 23
3.1.4. Gradient Descent (GD) 24
3.1.5. Sequential Quadratic Programming (SQP) 25
3.1.6. Sampling Based Gradient-Free Optimization 25
3.2. Visibility Aware Navigation (VAN) 28
3.3. Mathematical Preliminaries 29
3.3.1. Multi-Convexity and Alternating Minimization (AM) . 29
3.3.2. Trajectory Parametrization 30

4. Formulating Visibility Aware Navigation As an Optimization Problem **31**
4.1. Problem Formulation 31
4.1.1. Assumptions 31
4.1.2. Symbols and Notations 32
4.1.3. Trajectory Optimization 32
4.2. Related Works 33
4.2.1. Occlusion Model 33
4.2.2. Solution Process and Optimization Method 35

5. Visibility aware Navigation in Polygonal Environments **37**
5.1. Contributions 37
5.2. Trajectory Optimization 38
5.2.1. Reformulating Tracking Constraints 38
5.2.2. Combined Occlusion/Collision Avoidance Constraints . 39
5.2.3. Proposed Optimization Method 42
5.2.4. Solution by Split Bregman Iteration 43
5.2.5. Interpretation of the Occlusion Cost in the Proposed Formulation 46
5.2.6. Model Predictive Control (MPC) Through Real-Time Iteration 47
5.3. Optimizer Convergence 48
5.3.1. Robustness to Initialization 48

5.3.2. Comparisons with CCP 50
5.4. Validation and Benchmarking 51
5.4.1. Implementation Details 52
5.4.2. State-of-the-art and Comparison Metrics 53
5.4.3. Static Environment Tests 54
5.4.4. Dynamic Environment Tests 57
5.4.5. Additional Results 60
5.4.6. Real World Demonstrator 63

6. Visibility-aware Navigation in Arbitrary Environments 64
6.1. Contributions 64
6.2. Trajectory Optimization 64
6.2.1. Proposed Cross Entropy Method (CEM)-based Trajectory Optimizers 65
6.2.2. Efficient Batch Projection 67
6.2.3. Learned Occlusion Model 69
6.3. Validation and Benchmarking 73
6.3.1. Implementation Details 73
6.3.2. Point-to-Point Navigation 74
6.3.3. Static Environments 74
6.3.4. Dynamic Environment 76
6.3.5. Online Obstacle Adaptation 78
6.3.6. Real World Demonstrator 79

7. Cooperative Visibility-aware Navigation in Arbitrary Environments 80
7.1. Contributions 80
7.2. VACNA with Application in Warehouse Inventory 80
7.3. Trajectory Optimization for UAV-UGV Warehouse Inventory 81
7.3.1. Trajectory Parametrization 83
7.3.2. Proposed Optimizer 83
7.3.3. CVAE Initialization 88
7.4. Validation and Benchmarking 89
7.4.1. Implementation Details 89
7.4.2. Gaussian and CVAE Initialization 91
7.4.3. Effect of Convolutional Variational Autoencoder (CVAE) 92
7.4.4. Comparison with SOTA Sampling Based Optimizers . . 93
7.4.5. Comparison with A^* planners 94
7.4.6. Ablation Studies 95
7.4.7. CAVE Generalization to Unseen Enviornments 98
7.4.8. Real World Demonstrator 99

8. Conclusion **102**
8.1. Limitations 102
8.2. Future Work 103

Bibliography **104**

Acknowledgements **114**

Sisukokkuvõte (Summary in Estonian) **115**

Publications **117**
Real-Time Multi-Convex Model Predictive Control for Occlusion-Free Target Tracking With Quadrotors 119
Visibility-Aware Navigation With Batch Projection Augmented Cross-Entropy Method Over a Learned Occlusion Cost 145
VACNA: Visibility-Aware Cooperative Navigation with Application in Inventory Management 155
Multi-Modal Model Predictive Control Through Batch Non-Holonomic Trajectory Optimization: Application to Highway Driving 165

Curriculum Vitae **175**

Elulookirjeldus (Curriculum Vitae in Estonian) **176**

LIST OF FIGURES

1. Illustration of configuration space for a 3-DoF, planar, square-shaped robot. S and G are the start and goal points, respectively. The white area in (a) is the configuration space, and the white part and light-gray highlights in (b) represent the c_{free} and c_{obs}, respectively. 21
2. An example of regular navigation vs. visibility-aware navigation in an environment with one obstacle (red). 28
3. (a)-(c) show surface plots for $g(x_1, x_2, x_3)$, when x_3, x_1, and x_2 are held fixed, respectively and the function is evaluated for the other two variables. 30
4. (a) shows the visualization of the occlusion cost in [4] for a simple example with one obstacle. The target is a fixed point. As can be seen, the cost value increases radially from the center of the obstacle and away from the target. The cost surface does not explicitly depend on the geometry of the obstacle. Thus, some points which are occlusion-free are erroneously assigned high-cost values. This is exemplified in (b), wherein the point shown in green is occlusion-free with respect to the target (Line of Sight (LoS) (red) does not intersect with the obstacle (blue)) but the occlusion cost of this point is not zero. 34
5. The occlusion model used in the trajectory optimization in this thesis. It is defined using the distance $d_{oi}(t, u_j)$ between the points on the LoS connecting the quadrotor with the target at any given time instant t. A $d_{oi}(t, u_j) < 0$ implies that the LoS at time t intersects with the obstacle, and consequently results in occlusion. 35
6. The cost map resulting from an occupancy grid can be converted to a set of polygons as shown in (a). These can be further converted to ellipsoidal representations as shown in (b) that can be handled efficiently by the proposed optimizer. . . 37
7. The $3D$ polar representation of a target point with respect to the robot. The red arrow is the LoS vector between the robot and the target. $\alpha_r(t)$ and $\beta_r(t)$ represent the angle this vector makes with the X and Z axes, respectively. $d_r(t)$ denotes the magnitude of this vector. 39
8. Combined collision/occlusion avoidance model. d_{los} is the distance between a point on the LoS and the obstacle. The dashed ellipse signifies the size of the obstacle inflated by the size of the quadrotor. Avoiding collisions of each point on the LoS (u_j) with the obstacle at each time instant ensures collision/occlusion-free navigation. 40

9. (a) shows the visualization of the proposed occlusion cost given by (5.40) while (b) shows its heatmap representation. As can be seen, the cost-surface peaks at the center of the obstacle and gradually reduces to zero as we move away from it. Importantly, the exact geometry of the obstacle is reflected in the proposed occlusion cost. This is in sharp contrast to the occlusion model of [4] shown in Figure 4. 47
10. The convergence of the proposed optimizer for different initial guesses for a problem set-up where the quadrotor needs to move between a start and a goal position while keeping the LoS to the target (cyan) occlusion-free at all time instants. The obstacles are shown in blue and have been inflated with the size of the quadrotor. The heatmaps show the occlusion cost in different regions of the workspace. Note that if the occlusion constraint is ignored, the collision avoidance constraints are trivially satisfied by a straight line from the start to the goal point (red line). 49
11. (a) The sample trajectories obtained from the Convex Concave Procedure (CCP) [82] optimizer for the point-to-point navigation task presented in Figure 10(a). As shown in (b), the CCP optimizer is able to obtain a feasible solution in 3 iterations. However, as shown in (c) each iteration of CCP is computationally very expensive. (d) shows the optimal cost achieved by either of the optimizers. 50
12. The simulation pipeline. The main focus is on designing the high-level optimizer/MPC that provides a feasible feed-forward trajectory to an off-the-shelf low-level controller [89]. The proposed optimizer is guaranteed to provide a feasible trajectory to the low-level controller. To maintain clarity, only the x component of the state feedback is shown but the remaining y, z components are indeed integral parts of the state feedback. . . 52
13. The tracking performance of the proposed multi-convex MPC and the state-of-the-art AutoChaser [1] are qualitatively compared in (a) and (b) across two different simulation runs. While AutoChaser gets occluded at multiple instances (shown with yellow arrows), the proposed MPC can react fast to abrupt changes of the target's motion near the obstacles and avoid occlusions. It should be noted that AutoChaser has been provided with information about the global trajectory of the target in the form of intermediate waypoints. In contrast, the proposed MPC only needs information about the instantaneous position and velocity of the target. 55

14. The visibility score comparison between the proposed multi-convex MPC and the state-of-the-art algorithm AutoChaser [1]. The proposed MPC outperforms the state-of-the-art with visibility scores above zero across all the benchmarks. A visibility score of below zero implies occlusion or alternately non-satisfaction of the occlusion avoidance constraints. 56
15. Acceleration profiles and computation time of the proposed MPC and the state-of-the-art AutoChaser[1] are sown in (a) and (b), respectively. The proposed MPC achieves smoother acceleration profiles as well as lower computation times. . . . 57
16. The qualitative comparison between the proposed multi-convex MPC and [4]. The environment consists of a static target amongst three dynamic obstacles. The main task for the quadrotor is to continuously adapt its position to prevent occlusion from the dynamic obstacles. The proposed MPC maintains perfect visibility while the trajectories resulting from [4] get occluded at multiple instances. 58
17. The visibility scores achieved by (a) the proposed multi-convex MPC and (b) [4]. The proposed MPC outperforms [4] by maintaining a visibility score of above zero in all the benchmarks. The visibility score is a proxy for the satisfaction of occlusion avoidance constraints. In this particular experiment, the target is static and the simulation runs till the quadrotor reaches an occlusion-free region. Thus, the simulation times of the proposed MPC and that of the [4] are different. 59
18. The figure shows the comparison of acceleration profiles and computation time of the proposed MPC with [4]. The proposed MPC outperforms that of [4] in terms f trajectory smoothness by employing smaller acceleration profiles. The proposed MPC can run in real-time on embedded hardware such as Intel NUC and NVIDIA Jetson TX2. The latter is particularly crucial for quadrotors, given their lightweight structure. 60
19. Demonstration of the performance of the proposed MPC while tracking a dynamic target (red) amongst dynamic obstacles (grey). 61
20. The figure verifies how well the proposed MPC maintains the range specified for tracking constraints (5.2) in the benchmarks considered in Section 5.4.3. As the target trajectory is not known in advance, ensuring the satisfaction of range thresholds at all times is intractable. Thus, the evaluation is done for the extent to which the proposed MPC can minimize the violation of minimum and maximum specified distance for target tracking. 62

21. Snapshots from the hardware experiment of the proposed optimizer. The setup is similar to the one discussed in Section 5.4.4 with a static target amongst dynamic obstacles. The quadrotor needs to constantly change its position to avoid both occlusion and collision from the dynamic obstacles. The first column shows the third-person view while the second column depicts the view from the quadrotor's onboard camera. 63
22. Occlusion heatmap. (a) is the ground truth and (b) is predicted by the proposed learned occlusion cost. 72
23. The Figure shows the time taken to perform occlusion cost queries from an occupancy map using the SDF library compared with the proposed learned occlusion model. 72
24. (a) - (d) depict the visibility-aware trajectories generated using Alg. 1 while Figures (e) - (h) show the trajectories computed by Alg. 2. Although both algorithms work, from iteration 2 onwards, Alg. 2 ensures that all the sampled trajectories satisfy collision avoidance with the target. 74
25. A qualitative comparison of target tracking is conducted in an environment with 6 static obstacles. As observed, the baseline AutoChaser [1] is outperformed by both Alg. 1 and 2. Furthermore, Alg. 2 exhibits the best performance with near-perfect occlusion-free tracking. 75
26. A qualitative comparison between Alg. 2 and [4]. The environment consists of a static target amongst three dynamic obstacles. The task is for the quadrotor to continuously adapt its position to prevent occlusion from the dynamic obstacle. As can be seen, [4] gets occluded at several instances (shown in (a) - (b)) while the proposed Alg. 2 maintains perfect visibility (depicted in (c) - (d)). 77
27. Online obstacle adaptation of the proposed optimizer in Alg. 2. (a) shows the initial configuration with three obstacles. Two new obstacles are suddenly added to the environment (shown with yellows arrows in (b)) and occlude the quadrotor's LoS. The quadrotor moves to the bottom to clear its LoS from occlusions (shown in (c)). Notice how the quadrotor also avoids the new obstacle on the bottom (shown in (d)). 78
28. The qualitative comparison of a hardware experiment of the proposed optimizers. The environment consists of three obstacles and a moving human target. The human target rearranges the obstacles throughout the run and creates new configurations. (shown with yellow arrows) 79
29. (a) - (b) show top-down and third-person views of a typical warehouse inventory setting. 82

30. The pipeline for the learned CVAE. 88
31. The proposed CEM optimizer Alg. 2 with naive Gaussian and learned CVAE initialization. The latter produces distributions more focused around regions that are collision-free and maintain UAV-UGV and UAV-rack visibility. 92
32. The comparison of different metrics of the proposed optimizer when it is warm-started with Gaussian sampling and CVAE. Occlusion time refers to combined UAV-UGV and UAV-rack occlusion time. 93
33. The comparison of trajectories generated for UAV and UGV using A* and the optimizer presented in Alg. 2 95
34. An experiment demonstrating the importance of having an occlusion cost between the UAV and the rack. Figure (b) shows the result when the UAV's FoV is occluded due to absence of the occlusion cost. Figure (c) shows how adding the occlusion cost makes the vehicles overtake the obstacle from the bottom to maintain the rack's visibility (recall last term of (7.1a)). . . 95
35. (a) - (c) present the results of three ablations, where the responsibility of keeping the UAV-UGV LoS occlusion-free is handled by UAV, UGV or both. In Figure (a), the occlusion is handled by the UGV. Figure (b) depicts the results when the responsibility of avoiding occlusion is on the UAV. And finally, Figure (c) shows the result when occlusion is cooperatively handled by both. 96
36. The experiment depicted here illustrates the generalization of the proposed CVAE model to unseen scenarios. (a) and (c) present different parts of an inventory run where obstacles are introduced dynamically (shown as black circles) into the environment. (b) and (d) correspondingly indicate how the vehicles adapt to the new changes. The CVAE model adjusts the initialization trajectories to accommodate these changes, leading to trajectories that avoid collisions/occlusions stemming from the newly added obstacles. 98
37. The real-world experiment of Visibility-Aware Cooperative Navigation (VACNA) for scanning warehouse racks in a static environment. The fiducial markers visible in the UAV camera view resemble the items on the racks. 100
38. The figures show the real-world experiment of VACNA for scanning warehouse racks in a dynamic environment. The obstacles are displaced throughout the run (as can be seen in th external camera view) and UAV/UGV has to avoid collisions while maintaining the visibility of the rack and keeping UAV-UGV LoS free of occlusions. 101

LIST OF TABLES

1. Important Symbols 32
2. Occlusion Model Architecture 73
3. Total Target Trajectories: 10 / Sim. Time: 1290s 75
4. Total Obstacle Configurations: 8 / Sim. Time: 240s 76
5. PointNet Architecture 90
6. CVAE Encoder Architecture 90
7. CVAE Decoder Architecture 90
8. Overall Distance Traveled / Sim. Time: 1045.2m / 1628s . . . 93
9. Computation Time Comparison of A* Planners 94
10. Overall Distance Traveled: 1508.4m / Sim. Time: 1890s . . . 96
11. Overall Distance Traveled: 1508.4m / Sim. Time: 890s 97

LIST OF ABBREVIATIONS

Acronyms

ADMM Alternating Direction Method of Multipliers.
AM Alternating Mnimization.

BC Behavior Cloning.

CCP Convex Concave Procedure.
CEM Cross Entropy Method.
CMA-ES Covariance Matrix Adaptation Evolution Strategy.
CVAE Convolutional Variational Autoencoder.

DoF Degrees of Freedom.

FoV Field of View.

GD Gradient Descent.

LiDAR Light Detection and Ranging.
LoS Line of Sight.

MLP Multi-Layer Perceptron.
MPC Model Predictive Control.
MPPI Model Predictive Path Integral.

PRM Probabilistic Roadmap.

QP Quadratic Programming.

RGBD Red Green Blue Depth.
ROS Robot Operating System.
RRT Rapidly-Exploring Random Trees.

SB Split Bregman.
SDF Signed Distance Field.
SOTA State-Of-The-Art.
SQP Sequential Quadratic Programming.

UAV Unmanned Aerial Vehicle.
UGV Unmanned Ground Vehicle.

VACNA Visibility-Aware Cooperative Navigation.
VAN Visibility Aware Navigation.

LIST OF ORIGINAL PUBLICATIONS

Publications included in the thesis

1. **H. Masnavi**, J. Shrestha, M. Mishra, P. B. Sujit, K. Kruusamäe and A. K. Singh, "Visibility-Aware Navigation With Batch Projection Augmented Cross-Entropy Method Over a Learned Occlusion Cost," in IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 9366-9373, Oct. 2022, doi: 10.1109/LRA.2022.3190087.
2. **H. Masnavi**, V. K. Adajania, K. Kruusamäe and A. K. Singh, "Real-Time Multi-Convex Model Predictive Control for Occlusion-Free Target Tracking With Quadrotors," in IEEE Access, vol. 10, pp. 29009-29031, 2022, doi: 10.1109/ACCESS.2022.3157977.
3. **H. Masnavi**, J. Shrestha, K. Kruusamäe and A. K. Singh, "VACNA: Visibility-Aware Cooperative Navigation with Application in Warehouse Inventory," Under Review in Robotics and Automation Letters.
4. V. K. Adajania, A. Sharma, A. Gupta, **H. Masnavi**, K. M. Krishna and A. K. Singh, "Multi-Modal Model Predictive Control Through Batch Non-Holonomic Trajectory Optimization: Application to Highway Driving," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 4220-4227, April 2022, doi: 10.1109/LRA.2022.3148460.

Author's Contributions

In Publication I, the author developed a novel planning algorithm for visibility-aware navigation in environments with polygonal obstacles, developed the simulation setup, compared the developed method with various state-of-the-art approaches, and finally, deployed the results on actual hardware. The author was responsible for writing different sections of the paper.

In publication II, the author extended his previous work by developing a novel planning algorithm that can handle the task of navigation with visibility constraints in environments with arbitrary obstacles. The author developed the simulation setup, conducted the experiments, carried out the comparisons with the state-of-the-art, and implemented the results on actual hardware. The author was also responsible for the majority of the paper writing.

In publication III, the author developed a new planning algorithm for handling visibility constraints in cooperative navigation settings. The author developed a new simulation setup, bench-marked the method against different baselines, and implemented the algorithm on actual hardware. The paper also involved developing an imitation learning pipeline for further improving the computational performance of the developed planning algorithms. This was primarily done by the second author. The first author

contributed to the overall integration of the learned models into the planning pipeline. The author was also responsible for the full paper writing.

In contribution IV, a new trajectory planning algorithm was developed for autonomous driving. The author developed different experimental setups, compared the developed method against the state-of-the-art, and deployed the algorithm on actual hardware.

Other published work of the author

1. R. Valner, **H. Masnavi**, I. Rybalskii, R. Põlluäär, E. Kõiv, A. Aabloo, K. Kruusamäe, A. K. Singh, "Scalable and heterogenous mobile robot fleet-based task automation in crowded hospital environments-a field test," in Frontiers in robotics and AI, 9, 922835. https://doi.org/10.3389/frobt.2022.922835
2. S. Srikanth, M. Babu, **H. Masnavi**, A. Kumar Singh, K. Kruusamäe, K.M Krishna, "Fast Adaptation of Manipulator Trajectories to Task Perturbation by Differentiating through the Optimal Solution," in Sensors 2022, 22, 2995. https://doi.org/10.3390/s22082995
3. V. K. Adajania, **H. Masnavi**, F. Rastgar, K. Kruusamäe and A. K. Singh, "Embedded Hardware Appropriate Fast 3D Trajectory Optimization for Fixed Wing Aerial Vehicles by Leveraging Hidden Convex Structures," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 2021, pp. 571-578, doi: 10.1109/IROS51168.2021.9636337.
4. F. Rastgar, **H. Masnavi**, J. Shrestha, K. Kruusamäe, A. Aabloo and A. K. Singh, "GPU Accelerated Convex Approximations for Fast Multi-Agent Trajectory Optimization," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3303-3310, April 2021, doi: 10.1109/LRA.2021.3061398.
5. D. Aschenbrenner, JS. Rieder, D. Van Tol, J. Van Dam, Z. Rusak, JO. Blech, M. Azangoo, S. Panu, K. Kruusamäe, **H. Masnavi**, I. Rybalskii, et al., "Mirrorlabs - creating accessible Digital Twins of robotic production environment with Mixed Reality," 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), Utrecht, Netherlands, 2020, pp. 43-48, doi: 10.1109/AIVR50618.2020.00017.
6. F. Rastgar, A. K. Singh, **H. Masnavi**, K. Kruusamäe and A. Aabloo, "A Novel Trajectory Optimization for Affine Systems: Beyond Convex-Concave Procedure," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 1308-1315, doi: 10.1109/IROS45743.2020.9341566.

1. INTRODUCTION

Motion planning is a critical area of research in robotics and automation. It involves determining a sequence of actions or movements to achieve a desired objective or task while satisfying various constraints. In the context of mobile robots, the task, for example, can be simply to navigate between a start and a goal position. For manipulators, the task could be to pick and place an object. Common examples of the constraints include avoiding obstacles while respecting velocity and acceleration bounds. Despite these challenges, motion planning with various constraints is extensively used in several fields, including autonomous vehicles, industrial automation, and logistics, where a range of mobile robots are used to automate various tasks.

A specific type of motion planning, visibility-aware navigation, addresses the need to plan paths for mobile robots while maintaining the visibility of specific points of interest, such as landmarks or targets. In many applications, including target tracking, aerial cinematography [1], [2], and marker-based localization [3], keeping the target or object of interest in the field of view is a critical and challenging task.

In target tracking, visibility-aware navigation can be used to ensure that a tracking agent, such as a robot camera, maintains a clear view of the target [1][4]. For example, in a surveillance application, a robot may need to track a moving target while avoiding obstacles such as buildings or trees. By planning a path that maintains the visibility of the target, the robot can continue to track the target even as it moves and the environment changes.

Aerial cinematography is another application that benefits from visibility-aware navigation [1]. A camera mounted on a drone can be used to capture stunning aerial footage of landscapes, buildings, or events. However, to get the best shots, it is important to plan a path that maintains the visibility of the subject while avoiding obstacles such as buildings, trees, or other unforeseen obstacles [5]. By using visibility-aware navigation, a drone can capture high-quality footage while ensuring the safety of the subject and the surrounding environment.

State estimation and localization is another challenging component in the motion planning pipeline [6], where visibility-aware planning can play an important role. Localization involves determining the position and orientation of a robot or sensor relative to a known reference frame. This information is essential for accurate motion planning and control. In many cases, state estimation and localization rely on markers, such as fiducial markers or beacons, that can be easily detected and tracked by sensors [7][8]. However, maintaining the visibility of these markers can be challenging, particularly in environments with occlusions stemming from static and dynamic obstacles.

By incorporating visibility-aware planning into state estimation and localization, it is possible to improve the accuracy and robustness of these tasks [9]. For example, by planning paths that maintain the visibility of the markers, a robot can more accurately track its position and orientation even in complex environments and complete its task.

Besides the well-acknowledged applications of VAN discussed above, this thesis also leverages it for the cooperative navigation of a UAV and UGV. A potential utility of this cooperative navigation can be in inventory management. Herein, an UAV needs to scan the racks of the warehouses through its monocular camera while navigating through narrow, cluttered corridors [10]. There are two core challenges to it from the perspective of autonomous navigation. First, UAVs can carry limited computing power onboard. Second, mounting sensors like LiDAR and RGBD cameras on the UAV can increase its form factor and, at the same time, also reduce its flight time [11]. An increased UAV size may also make it unsuitable for navigation in tight indoor spaces. In such cases, a cooperative system can be introduced wherein a UAV is guided by a UGV. With its more significant form factor, the UGV can carry more extensive computing and sensing hardware and consequently plan motions for itself and the UAV. However, for the UGV to be able to plan their motions, UAV-UGV LoS should be clear from any occlusions. Using visibility-aware navigation the UAV and UGV can coordinate their motions not to lose each other's sight.

For the previously mentioned applications to benefit from visibility-aware navigation, two core challenges need to be addressed. First, robust algorithms should be developed to solve the problem of motion planning with visibility constraints in various environments (i.e. static or dynamic). Second, the solutions should perform real-time on embedded devices that can be easily carried by small robots like drones. This thesis aims to tackle these challenges by proposing innovative approaches to motion planning that can handle complex constraints such as visibility, while also ensuring efficient and real-time execution on resource-constrained devices.

2. CONTRIBUTIONS AND SCOPE OF THE THESIS

The overarching aim of this thesis is to develop novel motion planning algorithms in the context of visibility-aware navigation in highly cluttered and dynamic environments. Moreover, the thesis also aims to apply VAN to challenging problems such as target tracking in highly cluttered environments, aerial cinematography, and cooperative warehouse inventory with visibility constraints. Although not explicitly considered in this thesis, the proposed solutions can also be applied to scenarios of marker-based localization, where the visibility of certain markers for accurate state estimation is of paramount importance. Further, the proposed methods achieve high efficiency such that the algorithms can be deployed on embedded devices that can be carried by robots with limited payload and compute such as small UAVs. In light of the aforementioned objectives, the thesis contributions are summarized as follows:

- In Publication I, an MPC algorithm is developed for target tracking and aerial cinematography amongst static and dynamic obstacles [12]. A unified reformulation is presented that takes into account the tracking as well as collision/occlusion avoidance constraints. Thanks to this reformulation, the MPC runs real-time on embedded devices such as Jeston TX2. Furthermore, the developed method could be leveraged in maker-based localization applications.
- In Publication II, the author improves upon the previous work in terms of tracking targets with up to three times the speed as well as handling more obstacles in real-time. The analytical formulation for occlusion calculation developed in the previous work is replaced with a learned occlusion model. In addition, a new optimization approach is presented based on the CEM [13] to avoid getting stuck in local minima and further decrease the computation time. The new approach is an extension of the author's work in Publication IV, where the author presents a new approach to efficiently solve several optimization problems in parallel. Using this new method, the optimizer plans collision/occlusion-free motions purely based on online point clouds in completely unknown environments.
- In Publication III, the proposed visibility-aware formulation in Publication II is extended to incorporate cooperative navigation for a UAV and UGV performing a warehouse inventory task. Using this new formulation, a small-sized UAV using only a monocular camera can perform the task of scanning the racks in a warehouse while avoiding obstacles. The UGV, with its more significant form factor, can carry all the sensors and compute to plan the motion both for itself and the UAV as long as the UAV-UGV LoS is not occluded.

3. BACKGROUND

3.1. Motion Planning

Motion planning is a fundamental problem in robotics that involves finding a collision-free path for a robot to travel from its initial position to its desired destination. At its most basic level, this involves connecting the robot's starting and ending poses in an environment containing obstacles, using a trajectory that avoids collisions [14]. The spatial configuration of a robot is determined by its Degrees of Freedom (DoF), and the collection of all kinematically valid configurations that the robot can assume is known as its configuration space, or C-space [15]. This space can be divided into two subsets: the free space C_{free}, which consists of configurations where the robot can move without colliding with obstacles, and the obstacle space C_{obs}, which represents the regions occupied by obstacles where the robot cannot move without colliding. The configuration space is computed using the robot's geometry, its DoF, and the environment map that is fed into the planning algorithm. Figure 1 provides an illustration of C_{free} and C_{obs} configurations. Here, green is a 3-DoF, planar robot with its center shown in yellow. S and G are the start and goal configurations, respectively. The white area in (a) is the configuration space, while the free and forbidden configuration spaces are shown by the white segment and light-gray highlights in (b), respectively. More specifically, if the center of the robot overlays with the gray highlights at any time, it would result in collisions with either the obstacles or the walls (black edges). Notice that the trajectory (dashed-line) connecting start and goal configurations is entirely collision-free.

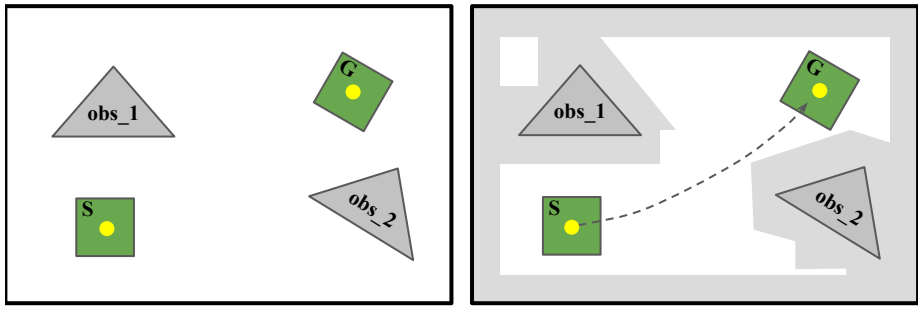


Figure 1. Illustration of configuration space for a 3-DoF, planar, square-shaped robot. S and G are the start and goal points, respectively. The white area in (a) is the configuration space, and the white part and light-gray highlights in (b) represent the c_{free} and c_{obs}, respectively.

Motion planning is critical for enabling robots to navigate through complex and dynamic environments and to perform tasks autonomously. This problem is challenging due to the high-dimensionality of the robot's configuration space and the need to avoid obstacles and other constraints while reaching the goal state. Therefore, motion planning algorithms must take into account the robot's kinematics and dynamics, environmental constraints, and goal specifications to find a feasible and efficient path for the robot to follow. These can include optimality, smoothness, and feasibility of the task. Broadly, the motion planning algorithms can be categorized into three different categories, namely Sampling-based, Search-based, and Optimization-based.

3.1.1. Sampling-Based

Sampling-based motion planning algorithms generate a set of random samples, called configurations or states, in the robot's configuration space (C-space). Sampling-based methods such as Rapidly-Exploring Random Trees (RRT) [16] and Probabilistic Roadmap (PRM) [17] use these configurations to build a graph, connecting the collision-free configurations to form a path between the initial and goal states. RRT and its variants grow a tree-like structure from the starting configuration while sampling new configurations in the configuration space. PRM, on the other hand, generates a graph by connecting configurations to their k-nearest neighbors. The graph can be searched for a path from the start to the goal configuration.

The primary advantage of these algorithms is that they are computationally efficient and work well in high-dimensional and complex environments [18]. They also have guarantees on finding a solution if one exists. However, sampling-based methods have some key shortcomings. One of the key problems with PRM and RRT is that they often produce collision-free trajectories that are sub-optimal. For example, the generated trajectory may include large detours around the obstacles. Thus, algorithms like PRM^* and RRT^* [19] introduce the notion of the shortest path within the PRM and RRT algorithms, respectively. Collision checking is another key bottleneck in sampling-based planners [20]. Sampling-based planners generate a large number of random points in the configuration space, which need to be checked for collision with the obstacles in the environment. This process involves performing collision detection algorithms, which can be time-consuming, especially if the environment has complex geometry or if the robot has many degrees of freedom [21]. Moreover, in many practical applications, the collision detection algorithms are not exact, and there is a trade-off between accuracy and computational efficiency. The planner must balance the number of collision checks performed with the level of accuracy required for the generated paths [22]. This balance is critical because too few collision checks may result in unsafe paths, while too many collision

checks may make the planner impractical for real-time applications [18]. Algorithms like Lazy PRM [23] solve this problem by performing collision checks on only a subset of sampled configurations. Some works like [24] accelerate the collision-checking process itself by using supervised learning.

3.1.2. Search-based

Search-based motion planning algorithms employ graph search algorithms such as A* [25] or Dijkstra's algorithm [26] to find a collision-free path from the start to the goal configuration. These algorithms are guided by a heuristic function that estimates the distance or more generally the cost to go between the current configuration and the goal configuration. The heuristic function guides the search to move towards the goal configuration while avoiding obstacles in the environment. These methods often use discretized representations of the configuration space to perform the search, such as grid-based or lattice-based methods [27]. Search-based algorithms are useful for environments with well-defined connectivity and narrow passages [28].

While search-based motion planning algorithms are effective in many scenarios, they do have some limitations. One of the main shortcomings of these algorithms is that they can be computationally expensive, particularly in high-dimensional spaces. This can make them impractical for real-time applications, where quick responses are necessary [29]. Additionally, search-based algorithms can be susceptible to getting stuck in local minima, which can result in suboptimal or inefficient paths [30]. Furthermore, these algorithms rely heavily on accurate models of the environment, which can be challenging to obtain in some scenarios, such as outdoor environments or areas with dynamic obstacles [29],[28].

3.1.3. Optimization-based

An alternate approach for motion planning is to formulate it as an optimization problem. This method, popularly known as trajectory optimization, has become the preferred motion planning approach for several robotic applications ranging from manipulation [31], [32], [33] to autonomous driving [34]. Succinctly, a trajectory optimization problem has the following generic mathematical form.

$$\begin{aligned} &\text{minimize} && f_o(\boldsymbol{\xi}), && && (3.1)\\ &\text{subject to} && g_i(\boldsymbol{\xi}) \leq b_i, && i = 1, ..., m && (3.2)\\ &\text{subject to} && h_j(\boldsymbol{\xi}) = \mathbf{0}, && j = 1, ..., n && (3.3) \end{aligned}$$

The variable $\boldsymbol{\xi}$ can be used to represent the various poses that the robot can take. Its definition can also be expanded to incorporate the various

velocities, accelerations as well as the control inputs required to move the robot. The f_o in (3.1) is called the cost function and models the notion of optimality in the motion planning. For example, a suitable f_o can be constructed to generate trajectories that connect the start and end pose of the robot with the shortest possible path. The function g_i in inequality constraints (3.2) can be used to model constraints such as collision avoidance. Similarly, the function h_j in (3.3) can be used to model constraints imposed by the dynamics of the robot or the task it is performing [33].

Trajectory optimization problems are computationally simple (i.e the global minimum can be computed in polynomial time) when they have a property called convexity wherein the functions f_o, g_i are shaped like a bowl and h_j has the shape of a hyper-plane [35]. Mathematically, the bowl shape of f_o, g_i results from their Hessian (or second derivative) being positive semi-definite. The function h_j in turn should have an affine form to have the hyper-plane shape. Unfortunately, most robot motion planning problems are non-convex wherein the cost and constraint functions have an arbitrary non-linear form (e.g product of trigonometric functions) [36],[37]. Such optimization problems have several local minimums besides the single global minimum [38]. Obtaining the global minimum in such problems is computationally intractable in most cases, especially if real-time performance requirements of motion planning are factored in. Thus, existing works focus on obtaining just the locally optimal solution. To this end, algorithms like Gradient Descent (GD)[31], Sequential Quadratic Programming (SQP) [33] and the CEM[13] are often used.

3.1.4. Gradient Descent (GD)

Gradient Descent [39] is a widely used first-order optimization technique for solving trajectory optimization problems. This approach works by iteratively updating the parameters to minimize the cost function. It computes the gradient of the cost function with respect to the parameters and updates them in the opposite direction of the gradient until the cost function is minimized [39].

However, gradient descent also has its own shortcomings. One of the main issues is that the algorithm can easily get stuck in local minima, which can prevent the robot from finding the global optimal solution [38]. This is because gradient descent only looks at the immediate neighborhood of the current solution, and if there is a deeper minimum nearby, it may not be able to find it. Additionally, gradient descent can be sensitive to the choice of the learning rate, which determines the step size taken at each iteration [40]. If the learning rate is too high, the algorithm can overshoot the minimum and fail to converge, while if it is too low, the convergence may be too slow.

3.1.5. Sequential Quadratic Programming (SQP)

SQP is a non-linear optimization technique used to solve constrained optimization problems [41]. The algorithm works by iteratively solving a Quadratic Programming (QP) sub-problem at each iteration. The QP sub-problem is formulated by approximating the objective function and constraints using a quadratic Taylor series expansion around the current solution point [42]. The solution of the QP sub-problem is then used to update the solution point and iterate the algorithm until a satisfactory solution is obtained. One of the main benefits of SQP is its ability to handle non-linear constraints efficiently and accurately, which is important in many engineering and scientific applications [43].

Another advantage of SQP is that it is a highly flexible and customizable optimization technique [44]. The algorithm allows for the incorporation of a wide range of problem-specific constraints and objective functions, making it suitable for a variety of applications. Additionally, SQP can handle problems with a large number of variables and constraints, making it suitable for complex systems with many interrelated parameters [44].

However, SQP also has several limitations that must be considered when using the algorithm. One of the main challenges with SQP is that it can be sensitive to the choice of starting point, which can affect the convergence and optimality of the solution [43]. Additionally, the algorithm can suffer from issues such as ill-conditioning, which can lead to numerical instabilities and difficulties in convergence [38]. Another potential limitation of SQP is that it requires the computation of Hessian matrices, which can be computationally expensive for large-scale problems [41].

3.1.6. Sampling Based Gradient-Free Optimization

a) ***Cross Entropy Method (CEM).*** CEM is a sampling-based optimization algorithm [13]. It generates a set of samples, also known as candidate solutions, and evaluates them based on a cost or objective function. The top-performing samples are selected, and their distribution is updated based on a weighted sum of the selected samples. The process is repeated iteratively until a satisfactory solution is found.

Consider the unconstraint objective function (3.1), the minimization for such a function using CEM boils down to steps presented Alg. 1. CEM is often used to solve optimization problems in high-dimensional spaces where traditional gradient-based methods may not be suitable [45]. It has been used in various applications such as reinforcement learning, machine learning, and control theory [46][47]. Although CEM is a sampling-based method, it differs from other sampling-based approaches like RRT or PRM in that it does not directly generate a graph or tree-like structure [48]. Instead, it optimizes the distribution of samples to find the optimal solution.

Algorithm 1: Cross Entropy Method for Minimizing $f(\boldsymbol{\xi})$

Input: Initial distribution parameters: mean vector μ, covariance matrix σ, sampling size n, elite set size m, convergence threshold ϵ, maximum number of iterations N

Output: Optimal solution: EliteSet with the lowest value for $f(\boldsymbol{\xi})$

for $i = 1$ **to** N **do**
  Sample n solutions from the current distribution
  Evaluate $f(\boldsymbol{\xi})$ for each solution
  Select the top m solutions (*EliteSet*)
  Update the distribution parameters using the *EliteSet*:
    - Compute the new mean for the *EliteSet*
    - Compute the new covariance for the *EliteSet*
  if *the average value of* $f(\boldsymbol{\xi})$ *for the EliteSet is less than or equal to* ϵ **then**
    Exit the loop and return the *EliteSet*
  end
end
Return the sample in the *EliteSet* with the lowest value for $f(\boldsymbol{\xi})$ as the optimal solution

Since CEM is an iterative algorithm that can quickly converge to a satisfactory solution, it makes ideal for real-time applications. Another advantage of CEM is that it can handle problems with stochastic and noisy objective functions by using weighted sampling, allowing it to find robust and reliable solutions [49].

However, like any optimization technique, CEM also has several limitations. One of the main challenges with CEM is that it requires a large number of samples to be generated to ensure that the optimal solution is found [13]. This can be computationally expensive, especially for large-scale problems with a high number of variables. Additionally, the performance of CEM can be sensitive to the choice of initial distribution and the selection of top-performing samples, which can affect the optimality of the solution. Finally, the algorithm is not guaranteed to find the global optimal solution, and it can converge to a local minimum instead [13].

Despite these limitations, CEM remains a popular and effective optimization technique that can handle complex, high-dimensional problems that are difficult to solve using other optimization methods [50].

b) ***Model Predictive Path Integral (MPPI).*** The Model Predictive Path Integral is a stochastic optimization algorithm that is used in control and robotics problems [51]. MPPI uses an MPC framework to generate a set of candidate trajectories for a system [52].

In each iteration, MPPI samples a set of control inputs and simulates the system under these inputs to generate a set of trajectories. The trajec-

tories are evaluated using a cost function that penalizes deviations from the desired state or behavior. The cost function is typically a sum of quadratic terms that penalize deviations from a desired state, as well as control effort. The best-performing trajectory is selected as the control input for the next time step, and the process is repeated [51], [53].

One of the main benefits of MPPI is its ability to generate optimal control trajectories for nonlinear and non-Gaussian systems by efficiently searching through a large space of control inputs [54]. Additionally, MPPI can handle constraints on the control inputs and states, making it useful in real-world applications where such constraints are present [55].

However, MPPI also has several limitations and shortcomings. It is sensitive to noise, and its performance may deteriorate significantly in the presence of modeling errors or disturbances [53]. Moreover, it has a limited horizon, and its performance is limited by the length of the horizon [56]. The planning horizon determines how far ahead in time the robot looks when making decisions.

MPPI has been successfully applied to a wide range of control and robotics problems, including quadcopter control and autonomous driving [56], [57].

c) ***Covariance Matrix Adaptation Evolution Strategy (CMA-ES)***. CMA-ES is a population-based stochastic optimization algorithm that is used for high-dimensional optimization problems [58]. It uses a multivariate normal distribution with a mean and covariance matrix to generate a population of candidate solutions. The mean and covariance are updated in each iteration based on the performance of the solutions to guide the search towards the optimal solution [59].

CMA-ES is able to handle non-convex and non-linear problems, making it useful in a wide range of applications [60]. Additionally, it is robust to noise and can handle constraints on the search space, which is important in many real-world optimization problems [61].

CMA-ES is particularly effective for problems with a large number of parameters because it adapts the covariance matrix to the shape of the objective function, which allows it to explore the search space efficiently [61].

However, CMA-ES also has some limitations and shortcomings. It can be computationally expensive, especially for high-dimensional problems, which can limit its use in real-time applications. It can also suffer from local optima, where it may get stuck in a suboptimal solution [59]. Furthermore, its performance can be highly dependent on the selection of the algorithm's parameters, such as the population size and the learning rate [58].

CMA-ES has been successfully applied to a wide range of optimization problems, including trajectory optimization, parameter tuning, and neural network training [60].

3.2. Visibility Aware Navigation (VAN)

VAN is a type of motion planning that takes into account the visibility constraints of the environment when generating motion plans for robots or other automated systems [62]. The goal of visibility-aware motion planning is to generate trajectories that maximize the visibility of the robot's sensors or cameras or to minimize occlusions while avoiding obstacles and satisfying other constraints. This problem is especially challenging because the collision-avoidance requirement is often in conflict with occlusion avoidance [4]. In other words, the trajectories which are collision-free are not necessarily occlusion-free. For example, consider the scenarios depicted in Figure 2. The task is for the quadrotor to reach the destination point (shown in black) while avoiding the red obstacle and maintaining the visibility of the human target. Although the quadrotor can choose to take the green trajectory (Figure 2(a)) and avoid the obstacle while reaching the destination point, its view of the human target would be occluded by the red obstacle. More specifically, its LoS (shown with magenta) intersects with the obstacle (highlighted with black circles). Therefore, it needs to take the blue trajectory (Figure 2(b)) to both avoid the obstacle and keep its LoS free of occlusions throughout the trajectory.

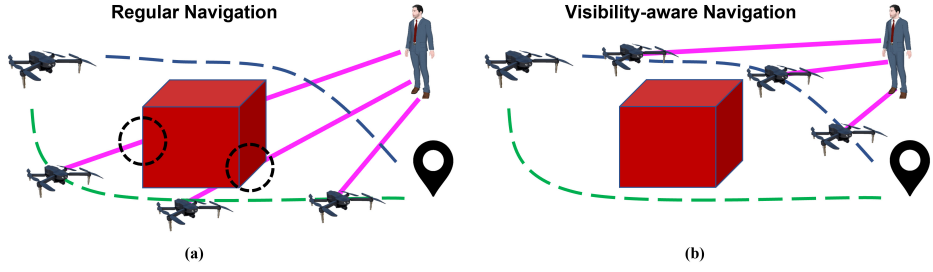


Figure 2. An example of regular navigation vs. visibility-aware navigation in an environment with one obstacle (red).

VAN forms a core component in autonomous target tracking and aerial cinematography pipelines [63], [64] and it is also important for improving state estimation [9].

Authors in [1], [5] present algorithms for single target tracking and aerial cinematography while minimizing occlusion in static environments. The work in [65] extends [1] by incorporating two targets for tracking. These works require a map of the environment, do not handle dynamic scenarios, and need the target's future trajectory for prediction.

The work in [4] leverages visibility-aware navigation for occlusion-free target tracking in environments with dynamic obstacles. However, due to the conservative method of handling occlusions (see Figure 4), it fails to

find an occlusion-free path in an environment with narrow passages (refer to the experiments in Section 5.4).

Authors in [66] present a visibility-aware approach for exploring the environment using two quadrotors. One quadrotor does the task of exploration while the other one keeps filming it and avoids occlusions stemming from the environment. However, it is shown in Section 7.4.6 that putting the responsibility of avoiding occlusion only on either of the vehicles does not suffice, and the vehicles would fail to keep their LoS unobstructed in more challenging, cluttered environments.

In the next few chapters, the thesis proposes a novel and comprehensive approach for solving VAN. A particular emphasis of the approach lies in avoiding occlusions from the environment while retaining real-time planning performance.

3.3. Mathematical Preliminaries

In this section, mathematical preliminaries such as Multi-Convexity, Alternating Minimization, and Trajectory Parametrization are covered. These are later used in solving the optimization problem (4.1a) - (4.1f).

3.3.1. Multi-Convexity and Alternating Minimization (AM)

a) ***Multi-Convexity.*** Consider a function $g(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$ whose arguments can be split into three separate blocks of variables. The function $g(.)$ is said to be multi-convex (multi-affine) if fixing two of the variable blocks makes it convex (affine) with respect to the remaining variables. For example, fixing $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2)$ makes g convex (affine) in $\boldsymbol{\xi}_3$ [67]. This definition trivially extends to an arbitrary number of blocks of variables.

b) ***Alternating Mnimization (AM).*** Alternating (or Gauss Seidel) minimization is an efficient technique for optimizing over multi-convex functions. For example, the iteration for minimizing multi-convex function $g(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$ has the following form [68]:

$$^{k+1}\boldsymbol{\xi}_1 = \arg\min_{\boldsymbol{\xi}_1} g(\boldsymbol{\xi}_1, {}^{k}\boldsymbol{\xi}_2, {}^{k}\boldsymbol{\xi}_3), \tag{3.4a}$$

$$^{k+1}\boldsymbol{\xi}_2 = \arg\min_{\boldsymbol{\xi}_2} g({}^{k+1}\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, {}^{k}\boldsymbol{\xi}_3), \tag{3.4b}$$

$$^{k+1}\boldsymbol{\xi}_3 = \arg\min_{\boldsymbol{\xi}_3} g({}^{k+1}\boldsymbol{\xi}_1, {}^{k+1}\boldsymbol{\xi}_2, \boldsymbol{\xi}_3), \tag{3.4c}$$

where k represents the iteration index. In each iteration, the optimization is done over each variable block in a sequence. While optimizing over a specific block, all other blocks of variables are held fixed at values obtained at either the previous iteration or that obtained at preceding steps of the same iteration.

A Toy Example: Consider $g(x_1, x_2, x_3) = (x_1 x_2 + x_3 - 2)^2$. Three blocks of variables $\boldsymbol{\xi}_1 = x_1$, $\boldsymbol{\xi}_2 = x_2$ and $\boldsymbol{\xi}_3 = x_3$ can be identified with respect to which $g(.)$ is multi-convex. Figure 3 presents the surface plots when one of the variable blocks is fixed and the function is evaluated for the other two. As can be seen in Figure 3(a), $g(.)$ is non-convex when $\boldsymbol{\xi}_3$ is fixed. In contrast, as shown in Figure 3 (b) - (c), $g(.)$ becomes convex when either of $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$, is fixed. Following (3.4a) - (3.4c) will reduce the minimization of $g(.)$ into a sequence of QPs.

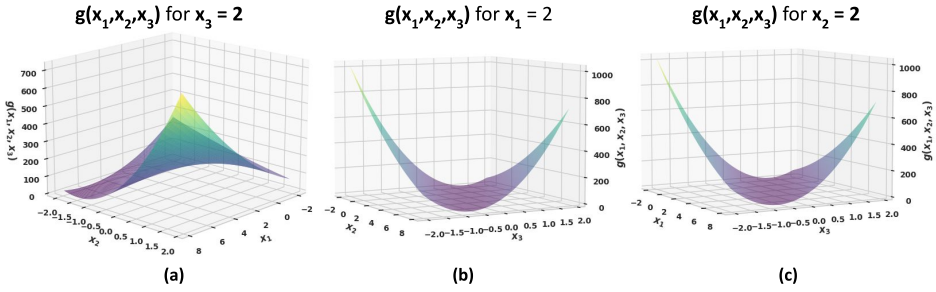


Figure 3. (a)-(c) show surface plots for $g(x_1, x_2, x_3)$, when x_3, x_1, and x_2 are held fixed, respectively and the function is evaluated for the other two variables.

Remark 1. *Identifying the correct variables blocks with respect to which a function is multi-convex is non-trivial. Several layers of reformulation are often needed to make the multi-convex structure more explicit. Moreover, the chosen blocks have a substantial impact on the efficiency of the resulting AM-based optimizer.*

3.3.2. Trajectory Parametrization

The optimizers developed in this work parametrize the position trajectories in the following manner:

$$\begin{bmatrix} x(t_1) \\ x(t_2) \\ \dots \\ x(t_n) \end{bmatrix} = \mathbf{W}\mathbf{c}_x, \begin{bmatrix} \dot{x}(t_1) \\ \dot{x}(t_2) \\ \dots \\ \dot{x}(t_n) \end{bmatrix} = \dot{\mathbf{W}}\mathbf{c}_x, \begin{bmatrix} \ddot{x}(t_1) \\ \ddot{x}(t_2) \\ \dots \\ \ddot{x}(t_n) \end{bmatrix} = \ddot{\mathbf{W}}\mathbf{c}_x. \tag{3.5}$$

$$\begin{bmatrix} y(t_1) \\ y(t_2) \\ \dots \\ y(t_n) \end{bmatrix} = \mathbf{W}\mathbf{c}_y, \begin{bmatrix} \dot{y}(t_1) \\ \dot{y}(t_2) \\ \dots \\ \dot{y}(t_n) \end{bmatrix} = \dot{\mathbf{W}}\mathbf{c}_y, \begin{bmatrix} \ddot{y}(t_1) \\ \ddot{y}(t_2) \\ \dots \\ \ddot{y}(t_n) \end{bmatrix} = \ddot{\mathbf{W}}\mathbf{c}_y. \tag{3.6}$$

$$\begin{bmatrix} z(t_1) \\ z(t_2) \\ \dots \\ z(t_n) \end{bmatrix} = \mathbf{W}\mathbf{c}_z, \begin{bmatrix} \dot{z}(t_1) \\ \dot{z}(t_2) \\ \dots \\ \dot{z}(t_n) \end{bmatrix} = \dot{\mathbf{W}}\mathbf{c}_z, \begin{bmatrix} \ddot{z}(t_1) \\ \ddot{z}(t_2) \\ \dots \\ \ddot{z}(t_n) \end{bmatrix} = \ddot{\mathbf{W}}\mathbf{c}_z. \tag{3.7}$$

where, $\mathbf{W}, \dot{\mathbf{W}}, \ddot{\mathbf{W}}$ are matrices formed with time-dependent basis functions (e.g polynomials) and $\mathbf{c}_x$, $\mathbf{c}_y$, $\mathbf{c}_z$ are the coefficients associated with the basis functions.

4. FORMULATING VISIBILITY AWARE NAVIGATION AS AN OPTIMIZATION PROBLEM

4.1. Problem Formulation

To address the problem of visibility-aware navigation, a trajectory optimization methodology is adopted in this thesis. This choice is motivated by several reasons.

Firstly, optimization-based methods can handle complex objective functions that involve multiple criteria or objectives, such as minimizing tracking errors, maximizing the number of tracked targets, or minimizing the total cost of tracking [49], [45]. Visibility-aware navigation planning often involves such objective functions that can be difficult to solve using search-based or sampling-based methods [2],[69],[4].

Secondly, optimization-based methods can incorporate constraints on the problem, such as maximum field-of-view angles, or physical constraints on sensor placement. This is particularly important for visibility-aware target tracking problems where such constraints should be satisfied for obtaining a feasible solution [69], [66].

Thirdly, optimization-based methods can leverage prior information about the target motion and sensor measurements to improve tracking accuracy and efficiency [2].

Finally, optimization-based methods can be more computationally efficient and scalable, particularly when the problem size is large [49], [38], for example, due to a longer planning horizon.

4.1.1. Assumptions

- It is assumed that the quadrotor (robot) is moving at moderate speeds for which kinematic models are sufficient. Due to the differential flatness property, the motions along x, y, z axes, and the yaw motion are all decoupled. These assumptions are standard in the existing literature on quadrotor motion planning, including those dealing specifically with target tracking [70] and drone-cinematography [71]. Also, like [71], [1], it is assumed that the variation of pitch and roll angles have minimal effects on the visibility of the target. This is reasonable for most of the commercially available quadrotors that have a Field of View (FoV) of approximately 90 degrees.

- Independent Camera Control: Similar to works like [70], [71], it is assumed that the camera motion is decoupled from the quadrotor's body motion. Thus, the quadrotor's camera can always be aligned to the LoS vector. For a quadrotor with a front-facing and a body-fixed

camera, this boils down to aligning the yaw angle of the quadrotor at each instant to the LoS vector.

4.1.2. Symbols and Notations

Normal-faced small case letters are used to represent scalars, while bold font variants represent vectors. The matrices are represented through bold-font upper case letters. The variable t denotes the time stamp of a variable. The upper case variant T indicates the transpose of a matrix. Some symbols are also defined in their first place of use. A special construction is used at some places in the thesis where a variable at different timestamps is stacked to form a vector. For example, $\boldsymbol{\alpha}$ will be formed by stacking different $\alpha(t)$s. The main important symbols are summarized in Table 1.

Table 1. Important Symbols

$x(t), y(t), z(t), \psi(t)$	Position, heading angle of the mobile robot (quadrotor) at time t.
$x_{oi}(t), y_{oi}(t), z_{oi}(t)$	Position of the i^{th}obstacle. The timestamp is relevant in case of dynamic obstacles.
u_j	A variable between 0 and 1 used to identify any point in the LoS trajectory.
$\alpha_{oi}(t, u_j), \beta_{oi}(t, u_j), d_{oi}(t, u_j)$	Variables associated with the collision and occlusion avoidance constraints.
$\alpha_r(t), \beta_r(t), d_r(t)$	Variables associated with our target tracking constraints.
$x_r(t), y_r(t), z_r(t)$	Target position at time t. These indicate the UAV position in chapter 7.
m	Number of planning steps

4.1.3. Trajectory Optimization

Considering the above assumptions, visibility-aware navigation can be mathematically formulated as the following optimization problem:

$$\min \sum_{t=t_0}^{t=t_m} \ddot{x}(t)^2 + \ddot{y}(t)^2 + \ddot{z}(t)^2 + f_{occ}(x(t), y(t), z(t), x_r(t), y_r(t), z_r(t)) \tag{4.1a}$$

$$(x(t_0), y(t_0), z(t_0), \dot{x}(t_0), \dot{y}(t_0), \dot{z}(t_0), \ddot{x}(t_0), \ddot{y}(t_0), \ddot{z}(t_0)) = \mathbf{b}_0 \tag{4.1b}$$

$$(x(t_m), y(t_m), z(t_m), \dot{x}(t_m), \dot{y}(t_m), \dot{z}(t_m), \ddot{x}(t_m), \ddot{y}(t_m), \ddot{z}(t_m)) = \mathbf{b}_f \tag{4.1c}$$

$$s_{min} \leq \left\| \begin{bmatrix} x(t) - x_r(t) \\ y(t) - y_r(t) \\ z(t) - z_r(t) \end{bmatrix} \right\|_2 \leq s_{max}, \tag{4.1d}$$

$$f_{vel}(\dot{x}(t), \dot{y}(t), \dot{z}(t)) \leq v_{max}, \tag{4.1e}$$

$$f_{acc}(\ddot{x}(t), \ddot{y}(t), \ddot{z}(t)) \leq a_{max} \tag{4.1f}$$

$$f_{vel} = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2 + \dot{z}(t)^2} \tag{4.2}$$

$$f_{acc} = \sqrt{\ddot{x}(t)^2 + \ddot{y}(t)^2 + +\ddot{z}(t)^2} \tag{4.3}$$

The first three terms in the cost function (4.1a) minimize the sum of the acceleration magnitude at discrete time instants in the planning horizon, ensuring the smoothness of the trajectory. The forth term ($f_{occ}(.)$) computes the occlusion of LoS between the robot and the target at discrete time instants in the planning horizon. Using the observations from the environment (i.e. obstacles point cloud) as well as the robot and target positions, $f_{occ}(.)$ outputs an occlusion value. The algebraic positive, semi-definite form of $f_{occ}(.)$ is discussed in detail in Section 4.2.1, where it is also explained how collision-avoidance can be rolled out into the occlusion cost. Constraints (4.1b) and (4.1c) define the initial and final boundary conditions on the trajectory, respectively. Thus, the vectors $\mathbf{b}_0, \mathbf{b}_f$ are simply stacking of initial and final positions, velocities, and accelerations. The inequality (4.1d) ensures that the robot would stay within certain bounds (s_{max}, s_{min}) from the target. The lower bound is essentially a collision avoidance requirement between the robot and the target. The upper bound is motivated by applications like aerial cinematography, wherein a large distance between the robot and the target can reduce the viewing quality. The inequality constraint (4.1e) - (4.1f) maintain the norm of the velocities and accelerations within specified bounds. The definitions of velocity and acceleration constraints are presented in (4.2) - (4.3).

4.2. Related Works

The literature contains several works that relate to visibility-aware motion planning. However, these works differ from the approach taken in this thesis in how they model occlusion stemming from the obstacle and the optimization techniques they use for trajectory planning. Thus, this section presents an overview of the existing works related to each of these aspects.

4.2.1. Occlusion Model

There exist several approaches in the literature to compute the cost function $f_{occ}(.)$ in (4.1a). Inspired by the GPU ray casting model [72], authors in [4] propose a cost function to be used within their developed MPC to prevent occlusion from obstacles.

Using the notation presented in Table 1, the occlusion cost of [4] is defined as

$$f_{occ} = \begin{cases} \|d_v\|, & \text{if } d_v > 0 \text{ and } p_{proj} > \mathbf{r}_{cti}^T \mathbf{r}_{cti} - 1. \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

$$d_v = \frac{p_{proj}}{\mathbf{r}_{ch}^T \mathbf{r}_{ch}}, \qquad p_{proj} = \mathbf{r}_{ch}^T \mathbf{r}_{cti}. \tag{4.5}$$

$$\mathbf{r}_{ch} = (x_r(t) - x(t), y_r(t) - y(t), z_r(t) - z(t)). \tag{4.6}$$

$$\mathbf{r}_{cti} = (x_{oi}(t) - x(t), y_{oi}(t) - y(t), z_{oi}(t) - z(t)). \tag{4.7}$$

The variables $\mathbf{r}_{ch}$, $\mathbf{r}_{cti}$ are respectively the vectors to the target and the i^{th} obstacle from the quadrotor's current position. The occlusion model of [4] first checks whether the obstacle blocks the target. That is, whether the target is behind the obstacle and falls within the occlusion cone formed by the tangents drawn to the obstacle ellipsoid from the quadrotor center. If the conditions of occlusion are met, then the visibility cost given by (4.4) is invoked.

Figure 4(a) shows a visualization of the cost surface for a simple scenario with a cylindrical-shaped obstacle and a stationary target point. The following key observations can be made from Figure 4(a). First, the occlusion cost does not depend on the geometry of the obstacle. It just increases radially from the center of the obstacle in a direction away from the target. This is an overly conservative model as a large part of the workspace that is indeed occlusion-free is assigned high-cost values (see Figure 4(b)). As a result, f_{occ} in (4.4) leads to sub-optimal performance when used within a trajectory optimizer. The second important thing to note is that the cost surface itself is non-smooth, indicating the computational difficulty of tractably optimizing over this cost function. Based on the experiments carried out in this thesis, it is also shown that it is difficult to reliably trade-off the occlusion cost of (4.4) with the costs stemming from collision avoidance.

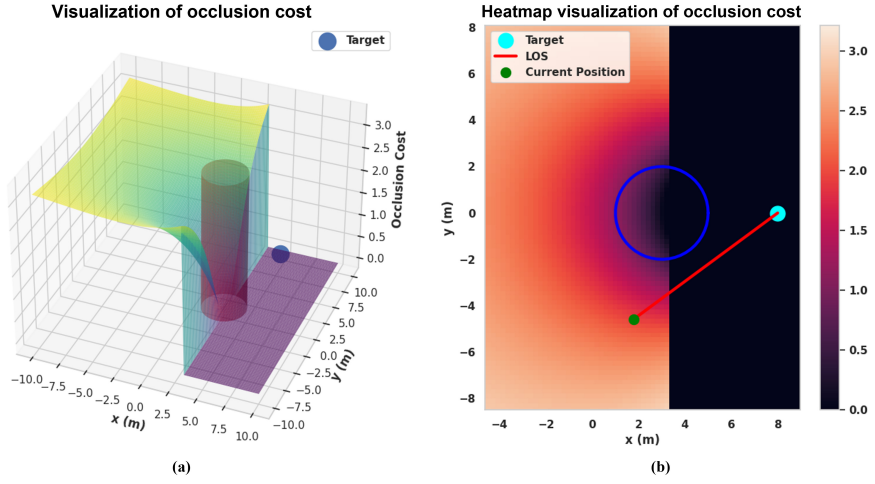


Figure 4. (a) shows the visualization of the occlusion cost in [4] for a simple example with one obstacle. The target is a fixed point. As can be seen, the cost value increases radially from the center of the obstacle and away from the target. The cost surface does not explicitly depend on the geometry of the obstacle. Thus, some points which are occlusion-free are erroneously assigned high-cost values. This is exemplified in (b), wherein the point shown in green is occlusion-free with respect to the target (LoS (red) does not intersect with the obstacle (blue)) but the occlusion cost of this point is not zero.

An alternate occlusion model that directly works in the image space is proposed in [73]. Let $\boldsymbol{\mu}_r$ and $\boldsymbol{\mu}_{oi}$ denote the quadrotor and obstacle projection on the 2D image, then occlusion can be defined in terms of euclidean separation between these image points in the following form:

$$\|\boldsymbol{\mu}_r - \boldsymbol{\mu}_{oi}\|_2^2 \geq r_{oi}^2, \tag{4.8}$$

where, r_{oi} is the length of the semi-minor axis of the ellipse obtained by projecting the sphere-shaped obstacles onto the image.

Finally, many existing works derive occlusion avoidance constraints from the line of sight (LoS), i.e. a line connecting the robot to the target at any given time instant [74], [75], [1], [5]. Intuitively, occlusion happens when the LoS passes through an obstacle (see Figure 5). Thus, occlusion avoidance can be essentially modeled as collision avoidance for the LoS trajectory constructed at each time instant. Authors in [1], [74], [5] use this insight to construct occlusion cost for their trajectory optimizers. Importantly, their cost representation can be derived from the generic signed distance field representation of the environment.

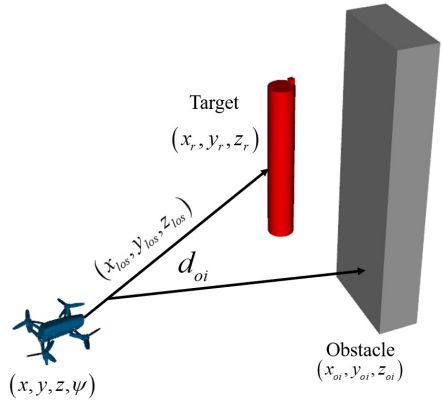


Figure 5. The occlusion model used in the trajectory optimization in this thesis. It is defined using the distance $d_{oi}(t, u_j)$ between the points on the LoS connecting the quadrotor with the target at any given time instant t. A $d_{oi}(t, u_j) < 0$ implies that the LoS at time t intersects with the obstacle, and consequently results in occlusion.

4.2.2. Solution Process and Optimization Method

A standard approach for solving optimization of the form (4.1a) - (4.1f) is to use techniques like SQP that are readily available through software libraries like ACADO [76], NLOPT [77]. For instance, authors in [73], [4] leverage SQP to solve the trajectory optimization problem.

In contrast, works like [74] use a customized form of first-order GD originally proposed for manipulation problems [31]. At the most fundamental level, both SQP and GD are based on first-order Taylor series expansion of

the underlying constraint functions. Thus, the initialization of these optimizers requires us to guess where the Taylor expansion will be computed at the first iteration. This initialization choice has a significant impact on the performance of the GD and SQP optimizers. A good initialization can be ensured by leveraging graph search techniques [62].

Authors in [1], [5] show how graph search techniques can also be used to decompose (4.1a) - (4.1f) into a hierarchy of smaller problems. However, since collision avoidance and smoothness can often conflict with occlusion minimization it makes gradient-based approaches fall in poor local minima most of the times.

Another approach for solving optimization problems of the form (4.1a) - (4.1f) is to use sampling-based trajectory optimization methods such as CEM [13]. CEM is a stochastic optimization technique that iteratively refines a distribution over candidate solutions to maximize a performance criterion. By sampling trajectories from a distribution over control inputs, CEM avoids the need for a good initialization and can find solutions that gradient-based methods may miss. Additionally, by iteratively updating the distribution over control inputs, CEM can adapt to changes in the environment and find solutions that are robust to disturbances.

5. VISIBILITY AWARE NAVIGATION IN POLYGONAL ENVIRONMENTS

This chapter solves the optimization problem (4.1a) - (4.1f) underpinning VAN under the following assumptions:

- The static and dynamic obstacles are modeled as axis-aligned ellipsoids in $2D$ and spheres in $3D$. This is a fairly standard assumption for dynamic obstacles such as humans [4]. For static obstacles in $2D$, the so-called costmap converter in Robot Operating System (ROS) [78] can be leveraged to provide a polygonal decomposition of occupancy maps that can be further decomposed into ellipsoidal representations. An example of this conversion is shown in Figure 6. In $3D$, each occupied voxel can be modeled as spheres. It is important to point out that many existing algorithms work with the same assumption for static obstacles [73], [79].

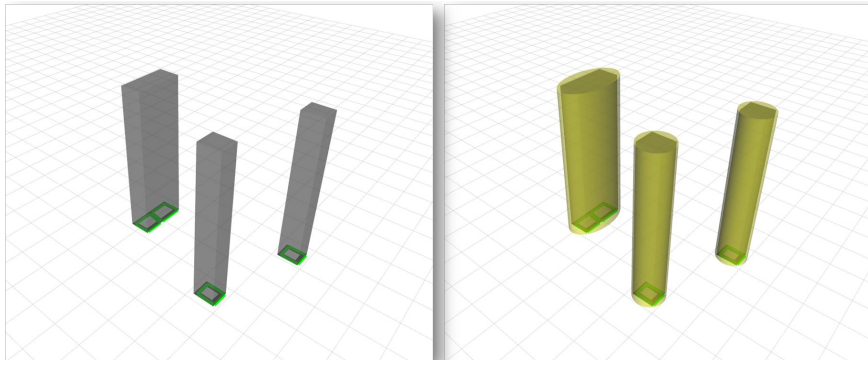


Figure 6. The cost map resulting from an occupancy grid can be converted to a set of polygons as shown in (a). These can be further converted to ellipsoidal representations as shown in (b) that can be handled efficiently by the proposed optimizer.

- The quadratic velocity and acceleration constraints in (4.1e) and (4.1f) are reformulated as decoupled linear inequalities along x, y, z axes.

5.1. Contributions

The core contributions of this chapter are as follows:

- A novel approach is presented for trajectory optimization associated with target tracking and aerial cinematography problems. At the heart of the proposed approach, lies a reformulation of collision/occlusion constraints into a form that resembles a polar representation of Euclidean distance. (refer to equations (5.2) and (5.12)).

This is different from the models used in previous works, such as [4] and [73](refer equations (4.4) and (4.8)). It is demonstrated that the representation leads to a multi-convex structure in the constraints, which can be decomposed into smaller blocks using the Alternating Minimization and Split-Bregman techniques [80].

- An MPC algorithm is developed using the multi-convex optimizer in a receding horizon manner following real-time iteration paradigm [81].
- Several state-of-the-art methods, including CCP [82], as well as those proposed in [1] and [4], are used as benchmarks to establish and validate the superiority of the proposed optimizer.

5.2. Trajectory Optimization

In this section, a trajectory optimization method is proposed to solve the problem in (4.1a) - (4.1f). The tracking constraints are reformulated, a model is proposed to calculate occlusion stemming from static and dynamic obstacles, and finally, the solution process is discussed.

5.2.1. Reformulating Tracking Constraints

This section builds upon the reformulation of quadratic constraints presented in [83] and rephrases the tracking constraint (4.1d) in the following polar form. This reformulation will prove useful for satisfying the constraint without resorting to linearization.

$$\mathbf{f}_{tar} = \mathbf{0}, \qquad s_{min} \leq d_r(t) \leq s_{max} \tag{5.1}$$

$$\mathbf{f}_{tar} = \left\{ \begin{array}{l} x(t) - x_r(t) - d_r(t) \sin\beta_r(t) \cos\alpha_r(t) \\ y(t) - y_r(t) - d_r(t) \sin\beta_r(t) \sin\alpha_r(t) \\ z(t) - z_r(t) - d_r(t) \cos\beta_r(t) \end{array} \right\} \tag{5.2}$$

Using the parametrization presented in (3.5) - (3.7), $\mathbf{f}_{tar} = \mathbf{0}, \forall t$ can be put in the following compact form:

$$\mathbf{A}_{tar}\mathbf{c}_x - \mathbf{b}^x_{tar}(\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) = \mathbf{0} \tag{5.3}$$
$$\mathbf{A}_{tar}\mathbf{c}_y - \mathbf{b}^y_{tar}(\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) = \mathbf{0} \tag{5.4}$$
$$\mathbf{A}_{tar}\mathbf{c}_z - \mathbf{b}^z_{tar}(\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) = \mathbf{0} \tag{5.5}$$

$$\mathbf{b}^x_{tar} = \mathbf{x}_r + \mathbf{d}_r \cos\boldsymbol{\alpha}_r \sin\boldsymbol{\beta}_r \tag{5.6}$$
$$\mathbf{b}^y_{tar} = \mathbf{y}_r + \mathbf{d}_r \sin\boldsymbol{\alpha}_r \sin\boldsymbol{\beta}_r \tag{5.7}$$
$$\mathbf{b}^z_{tar} = \mathbf{z}_r + \mathbf{d}_r \cos\boldsymbol{\beta}_r. \tag{5.8}$$

$\mathbf{A}_{tar} = \mathbf{W}$ and $\mathbf{x}_r, \mathbf{y}_r, \mathbf{z}_r$ are formed by stacking $x_r(t), y_r(t), z_r(t)$ at different time instants. A similar process is followed for constructing $\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r$ from their respective time-stamped scalar values. As depicted in Figure 7, $\alpha_r(t)$ and $\beta_r(t)$ denote the angle that the LoS vector between the robot and the

target (shown with red arrow) makes with the X and Z axes, respectively. Additionally, the variable $d_r(t)$ represents the magnitude of this vector.

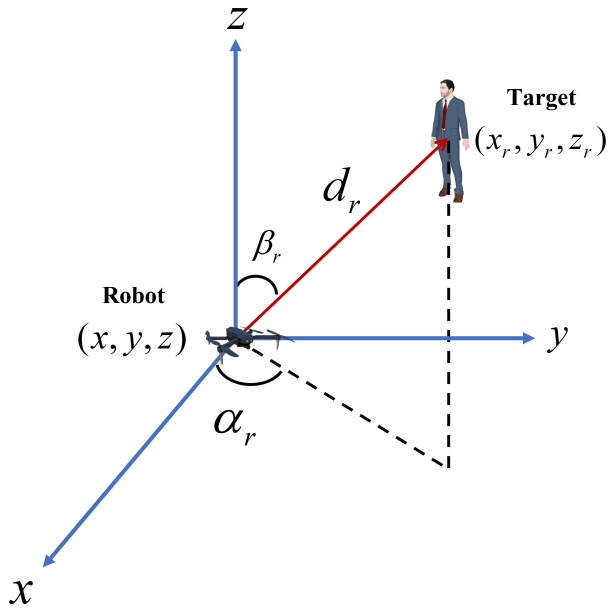


Figure 7. The $3D$ polar representation of a target point with respect to the robot. The red arrow is the LoS vector between the robot and the target. $\alpha_r(t)$ and $\beta_r(t)$ represent the angle this vector makes with the X and Z axes, respectively. $d_r(t)$ denotes the magnitude of this vector.

5.2.2. Combined Occlusion/Collision Avoidance Constraints

The LoS trajectory (shown with magenta in Figure 8) at time t is a straight line connecting the robot's position and the target position. Mathematically, any point on this trajectory can be identified as follows:

$$(x_{los}, y_{los}, z_{los})(t, u_j) = \begin{bmatrix} (1-u_j)x(t) + u_j x_r(t) \\ (1-u_j)y(t) + u_j y_r(t) \\ (1-u_j)z(t) + u_j z_r(t) \end{bmatrix}, \forall u_j \in [0,1] \tag{5.9}$$

The different points on the LoS trajectory are identified by their respective u_j values. Using (5.9), occlusion avoidance can be defined as collision avoidance for each point on the LoS trajectory. This can be derived as follows:

$$\frac{(x_{los}(t,u_j) - x_{oi}(t))^2}{a_i^2} + \frac{(y_{los}(t,u_j) - y_{oi}(t))^2}{b_i^2} + \frac{(z_{los}(t,u_j) - z_{oi}(t))^2}{c_i^2} \geq 1, \tag{5.10}$$

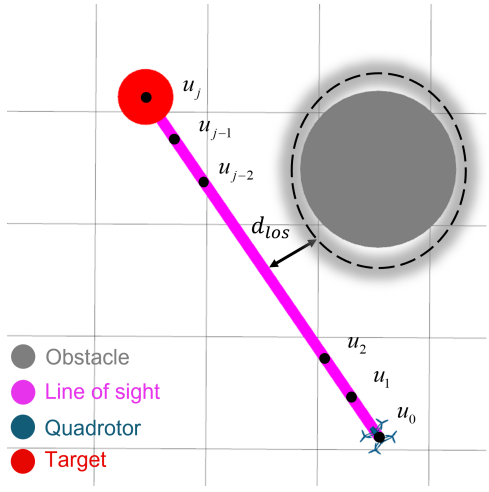


Figure 8. Combined collision/occlusion avoidance model. d_{los} is the distance between a point on the LoS and the obstacle. The dashed ellipse signifies the size of the obstacle inflated by the size of the quadrotor. Avoiding collisions of each point on the LoS (u_j) with the obstacle at each time instant ensures collision/occlusion-free navigation.

where, for ease of exposition, it is assumed that the i^{th} obstacle is an axis-aligned ellipsoid. The size of the ellipsoids (shown with dashed, black lines in Figure 8) is given by the radius along each axis as a_i, b_i, c_i which also incorporates the inflation due to the size of the quadrotor. Extension to rotated ellipsoids is straightforward and does not affect the multi-convex structure of the trajectory optimization in any manner. The inequality (5.10) guarantees that every point on the LoS trajectory is positioned outside of the obstacle ellipsoid.

The occlusion-avoidance constraints (5.10) implicitly include collision avoidance as well. Precisely, by setting u_j to 0 and evaluating (5.10), the standard collision avoidance can be obtained for the robot. The inequalities (5.10) are structured similarly to the tracking constraints shown in (4.1d). Thus, the same reformulation method discussed in the previous subsection is applied, hence (5.10) can be expressed in its polar form as follows:

$$\mathbf{f}_{occ} = \mathbf{0}, d_{oi}(t, u_j) \geq 1, \forall t, u_j. \tag{5.11}$$

$$\mathbf{f}_{occ} = \left\{ \begin{array}{l} (x_{los} - a_i d_{oi} \sin\beta_{oi} \cos\alpha_{oi})(t, u_j) - x_{oi}(t) \\ (y_{los} - b_i d_{oi} \sin\beta_{oi} \sin\alpha_{oi})(t, u_j) - y_{oi}(t) \\ (z_{los} - c_i d_{oi} \cos\beta_{oi})(t, u_j) - z_{oi}(t) \end{array} \right\}. \tag{5.12}$$

Using the trajectory parametrization presented in (3.5) - (3.7), $\mathbf{f}_{occ} = \mathbf{0}, \forall t, u_j$ can be reduced to the following form:

$$\mathbf{A}_{occ}\mathbf{c}_x = \mathbf{b}^x_{occ}(\mathbf{d}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o), \tag{5.13a}$$

$$\mathbf{A}_{occ}\mathbf{c}_y = \mathbf{b}^y_{occ}(\mathbf{d}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o), \tag{5.13b}$$
$$\mathbf{A}_{occ}\mathbf{c}_z = \mathbf{b}^z_{occ}(\mathbf{d}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o), \tag{5.13c}$$

where,

$$\mathbf{A}_{occ} = \begin{bmatrix} \mathbf{A}_u \\ \mathbf{A}_u \\ \vdots \\ \mathbf{A}_u \end{bmatrix}_{(\times n)}, \mathbf{A}_u = \begin{bmatrix} \mathbf{W}(1-u_1) \\ \mathbf{W}(1-u_2) \\ \vdots \\ \mathbf{W}(1-u_m), \end{bmatrix} \tag{5.14}$$

$$\mathbf{b}^x_{occ} = \widetilde{\mathbf{x}}_o + \mathbf{ad}_o \cos\boldsymbol{\alpha}_o \sin\boldsymbol{\beta}_o, \tag{5.15a}$$

$$\widetilde{\mathbf{x}}_o = \begin{bmatrix} \widetilde{\mathbf{x}}_{o1} \\ \widetilde{\mathbf{x}}_{o2} \\ \vdots \\ \widetilde{\mathbf{x}}_{on} \end{bmatrix}, \widetilde{\mathbf{x}}_{oi} = \begin{bmatrix} \mathbf{x}_{oi} + \mathbf{x}_r u_1 \\ \mathbf{x}_{oi} + \mathbf{x}_r u_2 \\ \vdots \\ \mathbf{x}_{oi} + \mathbf{x}_r u_m \end{bmatrix}$$

$$\mathbf{ad}_o \cos\boldsymbol{\alpha}_o \sin\boldsymbol{\beta}_o = \begin{bmatrix} a_1\mathbf{d}_{o1} \cos\boldsymbol{\alpha}_{o1} \sin\boldsymbol{\beta}_{o1} \\ a_2\mathbf{d}_{o2} \cos\boldsymbol{\alpha}_{o2} \sin\boldsymbol{\beta}_{o2} \\ \vdots \\ a_n\mathbf{d}_{on} \cos\boldsymbol{\alpha}_{on} \sin\boldsymbol{\beta}_{on} \end{bmatrix} \tag{5.15b}$$

$$\mathbf{b}^y_{occ} = \widetilde{\mathbf{y}}_o + \mathbf{bd}_o \sin\boldsymbol{\alpha}_o \sin\boldsymbol{\beta}_o, \tag{5.16a}$$

$$\widetilde{\mathbf{y}}_o = \begin{bmatrix} \widetilde{\mathbf{y}}_{o1} \\ \widetilde{\mathbf{y}}_{o2} \\ \vdots \\ \widetilde{\mathbf{y}}_{on} \end{bmatrix}, \widetilde{\mathbf{y}}_{oi} = \begin{bmatrix} \mathbf{y}_{oi} + \mathbf{y}_r u_1 \\ \mathbf{y}_{oi} + \mathbf{y}_r u_2 \\ \vdots \\ \mathbf{y}_{oi} + \mathbf{y}_r u_m \end{bmatrix},$$

$$\mathbf{bd}_o \sin\boldsymbol{\alpha}_o \sin\boldsymbol{\beta}_o = \begin{bmatrix} b_1\mathbf{d}_{o1} \sin\boldsymbol{\alpha}_{o1} \sin\boldsymbol{\beta}_{o1} \\ b_2\mathbf{d}_{o2} \sin\boldsymbol{\alpha}_{o2} \sin\boldsymbol{\beta}_{o2} \\ \vdots \\ b_n\mathbf{d}_{on} \sin\boldsymbol{\alpha}_{on} \sin\boldsymbol{\beta}_{on} \end{bmatrix} \tag{5.16b}$$

$$\mathbf{b}^z_{occ} = \widetilde{\mathbf{z}}_o + \mathbf{cd}_o \cos\boldsymbol{\beta}_o, \tag{5.17a}$$

$$\widetilde{\mathbf{z}}_o = \begin{bmatrix} \widetilde{\mathbf{z}}_{o1} \\ \widetilde{\mathbf{z}}_{o2} \\ \vdots \\ \widetilde{\mathbf{z}}_{on} \end{bmatrix}, \widetilde{\mathbf{z}}_{oi} = \begin{bmatrix} \mathbf{z}_{oi} + \mathbf{z}_r u_1 \\ \mathbf{z}_{oi} + \mathbf{z}_r u_2 \\ \vdots \\ \mathbf{z}_{oi} + \mathbf{z}_r u_m \end{bmatrix},$$

$$\mathbf{cd}_o \cos\boldsymbol{\beta}_o = \begin{bmatrix} c_1\mathbf{d}_{o1} \cos\boldsymbol{\beta}_{o1} \\ c_2\mathbf{d}_{o2} \cos\boldsymbol{\beta}_{o2} \\ \vdots \\ c_n\mathbf{d}_{on} \cos\boldsymbol{\beta}_{on} \end{bmatrix} \tag{5.17b}$$

$(\times n)$ in (5.14) denotes vertically stacking a matrix n times. n is the number of obstacles in the environment while m is the number of discrete $u_j \in [0\ 1]$ chosen to identify points on the LoS trajectory. The vector $\boldsymbol{\alpha}_{oi}$ is formed by stacking $\alpha_{oi}(t, u_j)$ for all t and u_j. A similar construction is followed for $\boldsymbol{\beta}_{oi}, \mathbf{d}_{oi}$. Moreover, different $\boldsymbol{\alpha}_{oi}, \boldsymbol{\beta}_{oi}, \mathbf{d}_{oi}$ are stacked across obstacle index i to construct $\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o, \mathbf{d}_o$.

Remark 2. *$\alpha_{oi}(t)$, $\beta_{oi}(t)$, $d_{oi}(t)$ are extra variables that will be computed by the optimizer along with trajectory parameters $\boldsymbol{c}_x$, $\boldsymbol{c}_y$, $\boldsymbol{c}_z$.*

5.2.3. Proposed Optimization Method

Using the reformulation presented in the previous subsection, the visibility-aware navigation problem in (4.1a) - (4.1f) can be reformulated as the following optimization problem:

$$\min \frac{1}{2}\boldsymbol{\xi}_1^T \mathbf{Q} \boldsymbol{\xi}_1, \tag{5.18a}$$

$$\mathbf{A}\boldsymbol{\xi}_1 = \mathbf{b}(\boldsymbol{\xi}_2, \boldsymbol{\xi}_3), \tag{5.18b}$$

$$\boldsymbol{\xi}_1 \in \mathcal{C}_{\xi_1}, \boldsymbol{\xi}_3 \in \mathcal{C}_{\xi_3}, \tag{5.18c}$$

$$\boldsymbol{\xi}_1 = (\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z) \tag{5.19}$$

$$\boldsymbol{\xi}_2 = (\boldsymbol{\alpha}_r, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_r, \boldsymbol{\beta}_o) \tag{5.20}$$

$$\boldsymbol{\xi}_3 = (\mathbf{d}_r, \mathbf{d}_o) \tag{5.21}$$

$$\mathbf{A} = \begin{bmatrix} \begin{bmatrix} \mathbf{A}_{tar} \\ \mathbf{A}_{occ} \end{bmatrix} & 0 & 0 \\ 0 & \begin{bmatrix} \mathbf{A}_{tar} \\ \mathbf{A}_{occ} \end{bmatrix} & 0 \\ 0 & 0 & \begin{bmatrix} \mathbf{A}_{tar} \\ \mathbf{A}_{occ} \end{bmatrix} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{b}_{tar}^x \\ \mathbf{b}_{occ}^x \\ \mathbf{b}_{tar}^y \\ \mathbf{b}_{occ}^y \\ \mathbf{b}_{tar}^z \\ \mathbf{b}_{occ}^z \end{bmatrix}, \tag{5.22}$$

$$\mathbf{Q} = \begin{bmatrix} \ddot{\mathbf{W}}^T \ddot{\mathbf{W}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddot{\mathbf{W}}^T \ddot{\mathbf{W}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddot{\mathbf{W}}^T \ddot{\mathbf{W}} \end{bmatrix}. \tag{5.23}$$

The cost function (5.18a) is just a matrix representation of the sum of squared acceleration presented in (4.1a), obtained through the trajectory parametrization (3.5) - (3.7). The sets $\mathcal{C}_{\xi_1}, \mathcal{C}_{\xi_3}$ represent the feasible sets for $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_3$ respectively. The set $\mathcal{C}_{\xi_1}$ is formed by the affine equalities and inequalities on $\boldsymbol{\xi}_1$. That is,

$$\mathcal{C}_{\boldsymbol{\xi}_1} = \{\boldsymbol{\xi}_1 \in \mathbb{R}^{3n} \mid \mathbf{C}_{eq}\boldsymbol{\xi}_1 = \mathbf{b}_{eq}, \mathbf{C}_{ineq}\boldsymbol{\xi}_1 \leq \mathbf{b}_{ineq}\}, \tag{5.24}$$

$$\mathbf{b}_{eq} = \begin{bmatrix} \mathbf{b}_{eq}^x \\ \mathbf{b}_{eq}^y \\ \mathbf{b}_{eq}^z \end{bmatrix}, \mathbf{b}_{ineq} = \begin{bmatrix} \mathbf{b}_{ineq}^x \\ \mathbf{b}_{ineq}^y \\ \mathbf{b}_{ineq}^z \end{bmatrix}, \mathbf{C}_{eq} = \begin{bmatrix} \mathbf{W}_0 \\ \dot{\mathbf{W}}_0 \\ \ddot{\mathbf{W}}_0 \\ \dot{\mathbf{W}}_m \\ \ddot{\mathbf{W}}_m \end{bmatrix}, \mathbf{C}_{ineq} = \begin{bmatrix} \dot{\mathbf{W}} \\ -\dot{\mathbf{W}} \\ \ddot{\mathbf{W}} \\ -\ddot{\mathbf{W}} \end{bmatrix} \tag{5.25}$$

$$\mathbf{b}_{eq}^{x} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ \ddot{x}_0 \\ \dot{x}_f \\ \ddot{x}_f \end{bmatrix}, \mathbf{b}_{eq}^{y} = \begin{bmatrix} y_0 \\ \dot{y}_0 \\ \ddot{y}_0 \\ \dot{y}_f \\ \ddot{y}_f \end{bmatrix}, \mathbf{b}_{eq}^{z} = \begin{bmatrix} z_0 \\ \dot{z}_0 \\ \ddot{z}_0 \\ \dot{z}_f \\ \ddot{z}_f \end{bmatrix}, \tag{5.26}$$

$$\mathbf{b}_{ineq}^{x} = \mathbf{b}_{ineq}^{y} = \mathbf{b}_{ineq}^{z} = \begin{bmatrix} v_{max}\mathbf{1} \\ -v_{min}\mathbf{1} \\ a_{max}\mathbf{1} \\ -a_{min}\mathbf{1} \end{bmatrix} \tag{5.27}$$

where the subscripts $0, m$ in $\mathbf{C}_{eq}$ denote the first and last elements of the respective matrices. The affine equalities $\mathbf{b}_{eq}$ model the boundary conditions on position, velocity, and accelerations, while the inequalities $\mathbf{b}_{ineq}$ model the lower and upper bounds on these variables present at all time instants. $\mathbf{1}$ is a vector of ones with the size of the number of planning steps.

The splitting of the variable blocks is done carefully to induce some useful structures in the problem and will be discussed shortly. Moreover, the fact that the cost function only depends on $\boldsymbol{\xi}_1$ will be shown to be useful.

5.2.4. Solution by Split Bregman Iteration

Split Bregman (SB) technique [80] solves optimization (5.18a) - (5.18c) by relaxing the non-convex equality constraints (5.18b) as l_2 penalties.

$$\min_{\boldsymbol{\xi}_1 \in \mathcal{C}_{\xi_1}, \boldsymbol{\xi}_3 \in \mathcal{C}_{\xi_3}} \frac{1}{2}\boldsymbol{\xi}_1^T \mathbf{Q} \boldsymbol{\xi}_1 - \langle \boldsymbol{\lambda}, \boldsymbol{\xi}_1 \rangle + \frac{\rho}{2} \left\| \mathbf{A}\boldsymbol{\xi}_1 - \mathbf{b}(\boldsymbol{\xi}_2, \boldsymbol{\xi}_3) \right\|_2^2 . \tag{5.28}$$

The key point to note in SB technique is the introduction of a Lagrange multiplier $\boldsymbol{\lambda}$ that aims to balance the conflicting objective of minimizing the primary cost function (acceleration norm) and the residual of the equality constraints. Essentially, $\boldsymbol{\lambda}$ reduces the impact of the primary cost function to enable the optimizer to focus on minimizing the constraint residual [84].

Remark 3. *In* (5.28)*, the non-convex constraints are all rolled into the form of an augmented Lagrangian cost. The remaining constraints in the trajectory optimizer are simply convex bounds on the position, velocity, and acceleration. Thus, the reformulated problem is feasible by construction. This is particularly useful when dealing with infeasible initializations.*

Remark 4. *Given $\boldsymbol{\xi}_2, \boldsymbol{\xi}_3$, the equality constraints* (5.18b) *is affine in $\boldsymbol{\xi}_1$ and the optimization* (5.28) *is a convex QP with respect to the same variable. Similarly, for a given $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$, the optimization is convex QP in the $\boldsymbol{\xi}_3$.*

Remark 5. *For a given $\boldsymbol{\xi}_1, \boldsymbol{\xi}_3$, optimization* (5.28) *is solvable in closed form for $\boldsymbol{\xi}_2$.*

a) ***Solution Process.*** SB employs AM approach for minimizing (5.28) that reduces to the following iterates, where left superscript k again specifies the iteration index.

$$^{k+1}\boldsymbol{\xi}_1 = \arg\min_{\boldsymbol{\xi}_1 \in \mathcal{C}_{\xi_1}} \frac{1}{2}\boldsymbol{\xi}_1^T \mathbf{Q}\boldsymbol{\xi}_1 - \langle {}^{k}\boldsymbol{\lambda}, \boldsymbol{\xi}_1 \rangle + \frac{\rho}{2} \left\| \mathbf{A}\boldsymbol{\xi}_1 - \mathbf{b}({}^{k}\boldsymbol{\xi}_2, {}^{k}\boldsymbol{\xi}_3) \right\|_2^2, \tag{5.29}$$

$$^{k+1}\boldsymbol{\xi}_2 = \arg\min_{\boldsymbol{\xi}_2} \frac{\rho}{2} \left\| \mathbf{A}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}(\boldsymbol{\xi}_2, {}^{k}\boldsymbol{\xi}_3) \right\|_2^2, \tag{5.30}$$

$$^{k+1}\boldsymbol{\xi}_3 = \arg\min_{\boldsymbol{\xi}_3 \in \mathcal{C}_{\xi_3}} \frac{\rho}{2} \left\| \mathbf{A}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}({}^{k+1}\boldsymbol{\xi}_2, \boldsymbol{\xi}_3) \right\|_2^2. \tag{5.31}$$

The Lagrange multiplier $\boldsymbol{\lambda}$ is updated based on the following rule [80], [84]:

$$^{k+1}\boldsymbol{\lambda} = {}^{k}\boldsymbol{\lambda} - \rho \nabla_{\boldsymbol{\xi}_1} \left\| \mathbf{A}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}({}^{k+1}\boldsymbol{\xi}_2, {}^{k+1}\boldsymbol{\xi}_3) \right\|_2^2. \tag{5.32}$$

Note that the gradient in the second term of (5.32) is taken with respect to only $\boldsymbol{\xi}_1$. This is because the primary cost function $g(.)$ is independent of $\boldsymbol{\xi}_2, \boldsymbol{\xi}_3$ [80].

b) ***Connections to ADMM***. The SB technique is closely related to the Alternating Direction Method of Multipliers (ADMM) [85], where a Lagrange multiplier is associated with each of the constraints (5.18b). In other words, the dimension of $\boldsymbol{\lambda}$ will be equal to the rows of $\mathbf{A}$. In contrast, in our use of the SB technique, $\boldsymbol{\lambda}$ will have a much lower dimension equal to that of $\boldsymbol{\xi}_1$. However, the main reason for choosing SB over ADMM is because it simplifies the iterates (5.30) - (5.31) to parallel least-squares problems.

c) ***Analysis of step*** (5.29). This optimization is a standard convex-constrained QP. Since the obstacles are axis-aligned ellipsoids (or cylinders), this QP can be split into three decoupled problems for each of $\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z$. For rotated ellipsoids, however, all three coefficients need to be computed simultaneously as the matrix $\mathbf{A}$ would no longer be block-diagonal. The state-of-the-art interior-point solvers for QP show a cubic scaling with respect to the total number of equality and inequality constraints. In (5.29), the number of hard inequality and equality constraints are fixed as these just stem from simple kinematic bounds and boundary conditions, respectively. The occlusion/collision avoidance enters just as a quadratic cost. Thus, the computation complexity of solving QP (5.29) is independent of the number of obstacles. As the number of obstacles increases, only the computation cost associated with constructing the QP, or in other words, obtaining $\mathbf{A}^T\mathbf{b}$ will change. Moreover, the increase in computation complexity can be made approximately linear with the parallelization of matrix-vector products. This is later validated in Section 5.3

d) ***Analysis of step*** (5.30). ${}^{k+1}\boldsymbol{\xi}_1$ $({}^{k+1}\mathbf{c}_x, {}^{k+1}\mathbf{c}_y, {}^{k+1}\mathbf{c}_z)$ are obtained in the previous step. Using this solution, $({}^{k+1}x(t), {}^{k+1}y(t), {}^{k+1}z(t))$ can be computed and stacked up to obtain the position trajectory $({}^{k+1}\mathbf{x}, {}^{k+1}\mathbf{y}, {}^{k+1}\mathbf{z})$. For a given position trajectory and ${}^{k}\boldsymbol{\xi}_3$, the variable blocks $(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r)$ and $(\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o)$ constituting $\boldsymbol{\xi}_2$ are decoupled from each other. Thus, optimization (5.30) decomposes into two parallel problems as follows:

$$\begin{aligned}\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r = \arg\min_{\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r} & \left\| {}^{k+1}\mathbf{x} - \mathbf{x}_r - {}^{k}\mathbf{d}_r \cos\boldsymbol{\alpha}_r \sin\boldsymbol{\beta}_r \right\|_2^2 \\ & + \left\| {}^{k+1}\mathbf{y} - \mathbf{y}_r - {}^{k}\mathbf{d}_r \sin\boldsymbol{\alpha}_r \sin\boldsymbol{\beta}_r \right\|_2^2 + \left\| {}^{k+1}\mathbf{z} - \mathbf{z}_r - {}^{k}\mathbf{d}_r \cos\boldsymbol{\beta}_r \right\|_2^2\end{aligned} \tag{5.33}$$

$$\begin{aligned}\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o = \arg\min_{\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o} & \left\| {}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o - {}^{k}\mathbf{d}_o \cos\boldsymbol{\alpha}_o \sin\boldsymbol{\beta}_o \right\|_2^2 \\ & + \left\| {}^{k+1}\widetilde{\mathbf{y}} - \widetilde{\mathbf{y}}_o - {}^{k}\mathbf{d}_o \sin\boldsymbol{\alpha}_o \sin\boldsymbol{\beta}_o \right\|_2^2 + \left\| {}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o - {}^{k}\mathbf{d}_o \cos\boldsymbol{\beta}_o \right\|_2^2\end{aligned} \tag{5.34}$$

$${}^{k+1}\widetilde{\mathbf{x}} = \mathbf{A}_{occ}\,{}^{k+1}\mathbf{c}_x, {}^{k+1}\widetilde{\mathbf{y}} = \mathbf{A}_{occ}\,{}^{k+1}\mathbf{c}_y, {}^{k+1}\widetilde{\mathbf{z}} = \mathbf{A}_{occ}\,{}^{k+1}\mathbf{c}_z \tag{5.35}$$

Optimizations (5.33) can be further simplified by noting that, for a given position trajectory, $(\alpha_r(t), \beta_r(t))$ at different time instants are decoupled from each other. In other words, all the elements of $\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r$ are independent of each other. With the same reasoning, it can be deduced that each of $(\alpha_{oi}(t, u_j), \beta_{oi}(t, u_j))$ are decoupled from each other and the decoupling here happens across time t, obstacle index i, and u_j. Thus, the elements of $\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o$ also have no inter-dependency.

The decoupled nature of the variables highlights one more computational structure. The l_2 penalties in (5.33) are just several parallel projections of ${}^{k+1}\mathbf{x} - \mathbf{x}_r$, ${}^{k+1}\mathbf{y} - \mathbf{y}_r$ and ${}^{k+1}\mathbf{z} - \mathbf{z}_r$ onto a sphere centered at origin with radius ${}^{k}\mathbf{d}_r(t)$ [86]. Therefore, the solution can be represented as the following closed-form expression.

$$\begin{aligned}{}^{k+1}\boldsymbol{\alpha}_r &= \arctan 2({}^{k+1}\mathbf{y} - {}^{k+1}\mathbf{y}_r, {}^{k+1}\mathbf{x} - {}^{k+1}\mathbf{x}_r), \\ {}^{k+1}\boldsymbol{\beta}_r(t) & \\ &= \arctan 2(\frac{{}^{k+1}\mathbf{x}(t) - {}^{k+1}\mathbf{x}_r}{\cos{}^{k+1}\boldsymbol{\alpha}_r}, {}^{k+1}\mathbf{z} - {}^{k+1}\mathbf{z}_r).\end{aligned} \tag{5.36}$$

Using a similar process, the solution of (5.34) can be derived in the following form.

$$\begin{aligned}{}^{k+1}\boldsymbol{\alpha}_o &= \arctan 2({}^{k+1}\widetilde{\mathbf{y}} - {}^{k+1}\mathbf{y}_o, {}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o), \\ {}^{k+1}\boldsymbol{\beta}_o(t) & \\ &= \arctan 2(\frac{{}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o}{\cos{}^{k+1}\boldsymbol{\alpha}_o}, {}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o).\end{aligned} \tag{5.37}$$

e) ***Analysis of step*** (5.31). For a given position trajectory at the $k+1$ iteration, $\mathbf{d}_r$ and $\mathbf{d}_o$ constituting $\boldsymbol{\xi}_3$ are decoupled from each other and thus (5.31) decomposes into following parallel sub-problems:

$$\begin{aligned} {}^{k+1}\mathbf{d}_r = \arg \min_{s_{min} \leq \mathbf{d}_r \leq s_{max}} & \left\| {}^{k+1}\mathbf{x} - \mathbf{x}_r - \mathbf{d}_r \cos {}^{k+1}\boldsymbol{\alpha}_r \sin {}^{k+1}\boldsymbol{\beta}_r \right\|_2^2 + \\ & \left\| {}^{k+1}\mathbf{y} - \mathbf{y}_r - \mathbf{d}_r \sin {}^{k+1}\boldsymbol{\alpha}_r \sin {}^{k+1}\boldsymbol{\beta}_r \right\|_2^2 + \\ & \left\| {}^{k+1}\mathbf{z} - \mathbf{z}_r - \mathbf{d}_r \cos {}^{k+1}\boldsymbol{\beta}_r \right\|_2^2 \end{aligned} \tag{5.38}$$

$$\begin{aligned} {}^{k+1}\mathbf{d}_o = \arg \min_{\mathbf{d}_o \geq 1} & \left\| {}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o - \mathbf{d}_o \cos {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 + \\ & \left\| {}^{k+1}\widetilde{\mathbf{y}} - \widetilde{\mathbf{y}}_o - \mathbf{d}_o \sin {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 + \\ & \left\| {}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o - \mathbf{d}_o \cos {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 . \end{aligned} \tag{5.39}$$

Both of the optimizations (5.38) and (5.39) are convex QPs with simple box constraints. It is worth noting that different time instant values of $d_r(t)$ are independent of each other when the position trajectory is fixed. In other words, the elements of $\mathbf{d}_r$ do not have any inter-dependency. A similar decoupling exists across the elements of $\mathbf{d}_o$. Thus, optimization (5.38) gets simplified to q parallel single-variable QPs, where q is the number of planning steps. Similarly, (5.39) reduces to $m{*}n{*}q$ parallel single-variable QPs. Each of these single variables QPs is solvable in symbolic form. More precisely, the unconstrained solution can be computed and then ensure the constraints by simply clipping the solution to the min/max values.

Remark 6. *The evaluation of closed-form symbolic solutions of* (5.33) - (5.34) *and* (5.38) - (5.39) *are linearly-complex operations that do not require any matrix-factorization/inverses or matrix-vector products.*

5.2.5. Interpretation of the Occlusion Cost in the Proposed Formulation

Consider the following residual extracted from the norm on the r.h.s of (5.32).

$$\begin{aligned} {}^{k+1}r_{occ} &= \left\| \mathbf{A}_{occ} {}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}_{occ}({}^{k+1}\boldsymbol{\xi}_2, {}^{k+1}\boldsymbol{\xi}_3) \right\|_2^2 \\ &= \left\| {}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o - {}^{k+1}\mathbf{d}_o \cos {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 \\ &\quad + \left\| {}^{k+1}\widetilde{\mathbf{y}} - \widetilde{\mathbf{y}}_o - {}^{k+1}\mathbf{d}_o \sin {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 \\ &\quad + \left\| {}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o - {}^{k+1}\mathbf{d}_o \cos {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 . \end{aligned} \tag{5.40}$$

The variable ${}^{k+1}r_{occ}$ shows how much of the occlusion-constraint is violated after the $k+1$ iteration of the proposed optimizer. It can also be viewed as the analogy of occlusion cost in the proposed formulation. A visualization of this cost surface at any particular time instant as well as its heatmap

representation are shown in Figures 9(a) and (b), respectively. To construct the cost surface, a large number of different $({}^{k+1}\mathbf{x}, {}^{k+1}\mathbf{y})$ are sampled and the corresponding optimal ${}^{k+1}\boldsymbol{\alpha}_o, {}^{k+1}\boldsymbol{\beta}_o, {}^{k+1}\mathbf{d}_o$ are computed from (5.37) and (5.39), respectively.

As it can be seen in Figure 9, the proposed occlusion cost perfectly captures the effect of obstacle geometry. The cost value peaks at the center of the obstacle and then smoothly drops off to zero as we move away from it. The proposed occlusion cost also correctly assigns zero cost to any point from where the LoS to the target is not obstructed by the obstacle. Both the smooth cost surface and accurate modeling of occlusion are in sharp contrast to the cost surface of [4] shown in Figure 4.

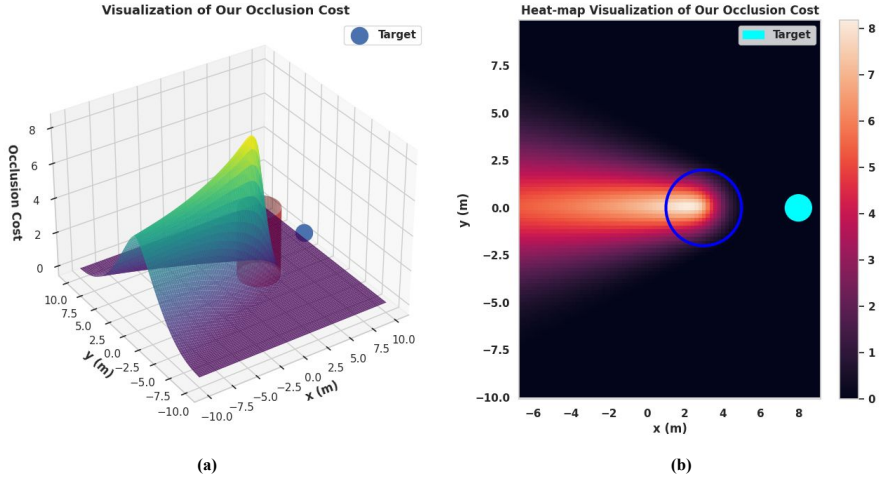


Figure 9. (a) shows the visualization of the proposed occlusion cost given by (5.40) while (b) shows its heatmap representation. As can be seen, the cost-surface peaks at the center of the obstacle and gradually reduces to zero as we move away from it. Importantly, the exact geometry of the obstacle is reflected in the proposed occlusion cost. This is in sharp contrast to the occlusion model of [4] shown in Figure 4.

5.2.6. MPC Through Real-Time Iteration

Here, an MPC approach is proposed that involves solving the trajectory optimization (5.18a) - (5.18c) in a receding horizon manner. That is, at each step, the full trajectory is obtained, but only a small portion of it is executed. Practically, a small portion of the velocity trajectory is averaged to obtain a piece-wise constant approximation of the time-varying profile, which is then commanded to the quadrotor. A similar process has been followed in many works such as [87].

A fundamental challenge in the MPC is to solve the underlying trajectory optimization in real-time. In practice, it is not feasible to solve the optimization till convergence. As a result, the so-called Real-Time Iterations

scheme [81] is adopted, which involves obtaining an approximate solution by running the optimization for a few iterations. For instance, in the non-linear MPC community, it is common to use just one iteration of either the SQP or Gauss-Newton method [81], [88]. The obtained solution is given to the robot and for the next MPC step, the trajectory optimization is warm-started from the previous solution. The obtained solution is then provided to the robot, and for the next step of MPC, the trajectory optimization starts from the previous solution. This process of warm-starting is crucial, as it simulates an online approach to solving an optimization problem.

5.3. Optimizer Convergence

The objective of this section is two-fold. First, the convergence of the proposed optimizer is empirically validated. Second, the proposed optimizer is compared with an alternate approach where the original formulation (4.1a) - (4.3) is solved using the state-of-the-art CCP [82]. The occlusion constraints are modeled through the standard quadratic inequality form presented in (5.10). The comparison with CCP is meant to demonstrate the effectiveness of the multiple layers of proposed reformulations and the resulting AM optimizer.

For empirical validation of the proposed optimizer, a simple problem scenario is considered, where a quadrotor needs to move between a start and a goal position (point-to-point navigation setting) while keeping the LoS to a static target occlusion-free. Thus, the tracking constraints are not present in this setting.

5.3.1. Robustness to Initialization

Figures 10(a) - 10(f) present the convergence results for the proposed optimizer with different initial guesses. In Figures 10(a) and 10(c), the initial-guess trajectory violates both collision and occlusion avoidance constraints. In Figure 10(e), a trivial straight line from the start position to the target is used as the initial guess. As depicted, the proposed optimizer is robust to different initial guesses and as observed, on average, 40-50 iterations are enough to obtain occlusion residual in the order of 10^{-3}.

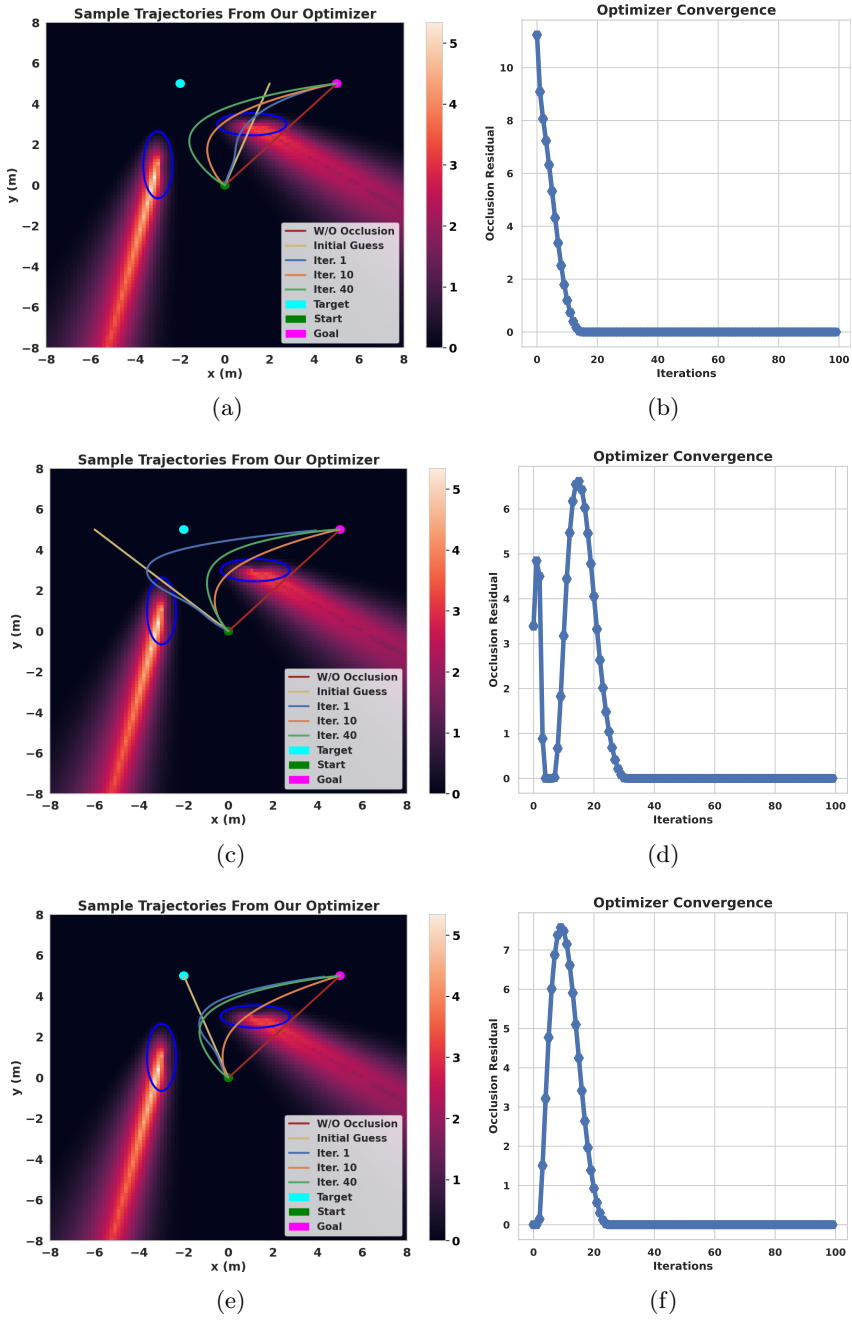


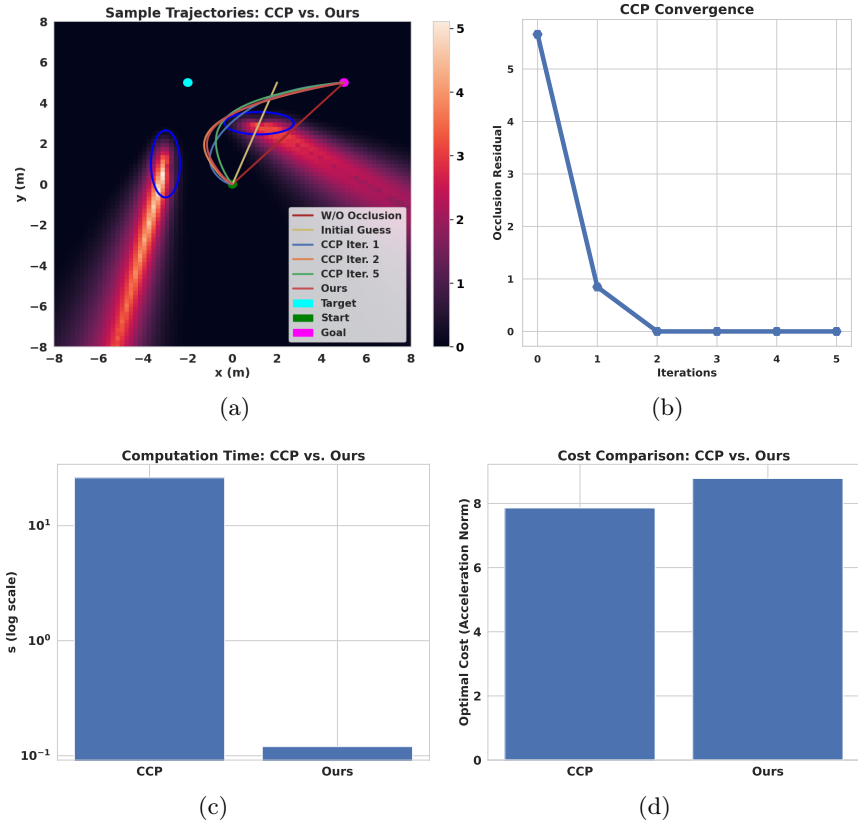
Figure 10. The convergence of the proposed optimizer for different initial guesses for a problem set-up where the quadrotor needs to move between a start and a goal position while keeping the LoS to the target (cyan) occlusion-free at all time instants. The obstacles are shown in blue and have been inflated with the size of the quadrotor. The heatmaps show the occlusion cost in different regions of the workspace. Note that if the occlusion constraint is ignored, the collision avoidance constraints are trivially satisfied by a straight line from the start to the goal point (red line).

5.3.2. Comparisons with CCP

The CCP [82] approach for solving the point-to-point navigation discussed in the previous subsection would boil down to solving the following optimization problem:

$$\text{minimize} \;\; (4.1a), \text{subject to} \;\; \mathbf{F}\boldsymbol{\xi}_1 \leq \mathbf{g} \tag{5.41}$$

where $\mathbf{F}, \mathbf{g}$ are obtained by linearizing the quadratic occlusion constraints (5.10). Figures 11(a) and 11(b) show a qualitative result obtained with CCP. For comparison, the final trajectory obtained with the proposed optimizer is also shown.



(a) (b)

(c) (d)

Figure 11. (a) The sample trajectories obtained from the CCP [82] optimizer for the point-to-point navigation task presented in Figure 10(a). As shown in (b), the CCP optimizer is able to obtain a feasible solution in 3 iterations. However, as shown in (c) each iteration of CCP is computationally very expensive. (d) shows the optimal cost achieved by either of the optimizers.

From a qualitative standpoint, the trajectories resulting from both optimizers seem similar. The CCP optimizer usually obtains a feasible solution within 2 to 3 iterations and a couple of more iterations are required to decrease the cost value. Therefore, the number of iterations required by the

CCP is substantially less than the proposed optimizer (40-50). However, each iteration of CCP is more expensive and thus, as shown in Figure 11(c), the proposed optimizer outperforms CCP's computation time by more than two orders of magnitude. The timings presented in Figure 11(c) are averaged values over 10 different problem instances and correspond to the Python implementations of both of the optimizers on an i7-8600 laptop with 32 GB RAM.

The difference in computation time can be attributed to the structural difference between the proposed optimizer and CCP. The number of constraints stemming from occlusion avoidance in the latter is equal to the number of rows in $\mathbf{F}$ in (5.41), which in turn is equal to $m * n * q$ (planning horizon times number of obstacles times discretization of u_j). The CCP optimizer is run with $m = 20, n = 2, q = 100$ resulting in a total of 2000 inequality constraints. There is also an additional overhead of obtaining the matrix $\mathbf{F}$ and vector $\mathbf{g}$ at each iteration of CCP based on the refined linearization of (5.10) around the current solution. It is worth noting that in some experiments, a denser discretization of u_j could be needed, which would only further increase the computational burden of CCP.

In contrast, the proposed optimizer uses $m = 100, n = 2, q = 100$ and yet has a substantially lower computation time. This is because the proposed optimizer handles occlusion constraints by augmenting them as l_2 norm penalties in steps (5.29) - (5.30) and as a combination of both penalty and inequality constraints in step (5.31). In (5.29), occlusion constraints are reformulated as a quadratic cost (third term), and since there are no velocity/acceleration bounds in the example presented in this subsection, this step is simply an equality-constrained QP with a closed-form solution. The step (5.30) has a symbolic solution (5.37) whose evaluation does not require any computation of matrix-vector/matrix-matrix products or matrix factorization. The step (5.31) is a convex-constrained QP with dimensions equal to $m * n * q$. However, the massive distributive nature of this step allows for obtaining a closed-form symbolic solution with no requirement of any sort of matrix operation (Recall (5.39) and discussions around it).

The CCP optimizer, however, marginally outperforms the proposed optimizer in terms of achieved optimal cost as presented in Figure 11(d). This is because the choice of augmenting occlusion constraints in the cost function leads to a conflict between occlusion and the primary objective of minimizing the acceleration norm.

5.4. Validation and Benchmarking

The objective of this section is to validate the proposed optimizer and benchmark the MPC built on top of the proposed optimizer with existing state-of-the-art approaches. The entire multi-convex MPC and simulation pipeline

is publicly available[1]. The simulation pipeline is shown in Figure 12. The main focus is on designing an optimizer/high-level MPC that provides a feasible feed-forward trajectory to the lower-level controller. The controller in [89] is used as the low-level controller that is guaranteed to track any smooth, differentiable trajectory respecting the kinematic bounds. The trajectories resulting from the proposed optimizer are guaranteed to fulfill all these conditions. The smoothness stems from the polynomial nature of the trajectories, while the kinematic bounds are included as convex constraints in the optimizer, and thus they are guaranteed to be satisfied at each control cycle.



Figure 12. The simulation pipeline. The main focus is on designing the high-level optimizer/MPC that provides a feasible feed-forward trajectory to an off-the-shelf low-level controller [89]. The proposed optimizer is guaranteed to provide a feasible trajectory to the low-level controller. To maintain clarity, only the x component of the state feedback is shown but the remaining y, z components are indeed integral parts of the state feedback.

5.4.1. Implementation Details

In order to perform the benchmarking, the trajectory optimization (5.18a) - (5.18c) and the resulting MPC are implemented in C++ using Eigen [90] as the linear algebra back-end. The MPC is integrated with Gazebo physics engine in ROS [91] to perform real-time high-fidelity simulations. The simulator provides state feedback on the quadrotor and the obstacles at 100 Hz. However, the proposed MPC can potentially run at up to 250Hz, depending on the number of obstacles in the environment as well as the number of iterations of the optimizer. Due to the warm-starting of the MPC from the solution obtained in the previous control cycle, just one iteration of the optimizer proves to be sufficient in almost all the benchmarks. The target trajectory is constructed by first tele-operating it over the workspace and then replaying the recorded trajectory during run-time. The tele-operation setup makes it possible to generate complex trajectories for the target.

The proposed MPC implementation has terminal constraints that force the final velocity and acceleration of the quadrotor to be zero. It was also tried to set the terminal values to the predicted velocity and acceleration of the target. However, this led to poor performance. This is not surprising as the long-term trajectory of the target is not known in advance and the linear predictions used by the proposed MPC are only rough estimates of

[1] https://github.com/Houman-HM/multi-convex-mpc-target-tarcker

the true values. The prediction horizon of the MPC is set to 10s.

5.4.2. State-of-the-art and Comparison Metrics

A core objective of this section is to establish the superiority of the proposed multi-convex MPC over two state-of-the-art approaches: the AutoChaser algorithm proposed in [1] and MPC algorithm of [4]. The former is designed for tracking amongst only static obstacles, while the latter considers dynamic obstacles as well. The publicly available implementation of [1] is used for benchmarking. This implementation has a prediction horizon of 7s. The author tried increasing this value, but it increased the computation time and the resulting control delay led to poor tracking performance.

To the best of the author's knowledge, no open-source implementation of [4] has been released by its authors. Thus, by closely following the mathematical formulation presented in [4], the author implemented their work using the state-of-the-art library ACADO [76] to prototype the trajectory optimization and the resulting MPC of [4]. Extensive tuning was performed between collision avoidance, occlusion avoidance, and acceleration cost to get the best performance for the MPC of [4]. This implementation of [4] is also released for verification[2].

The following metrics are used to benchmark the proposed MPC with state-of-the-art approaches:

Visibility Score: Following [1], the visibility score is defined as the smallest distance between static/dynamic obstacles and the LoS trajectory. As such, a distance below zero would imply that the LoS is blocked by the obstacles and the target is occluded from the quadrotor's camera. The visibility score is a proxy for the satisfaction of occlusion constraints and thus, a value less than zero means the occlusion constraints are not satisfied. It is worth mentioning that the exact value of the visibility score is not important as long as it is above zero.

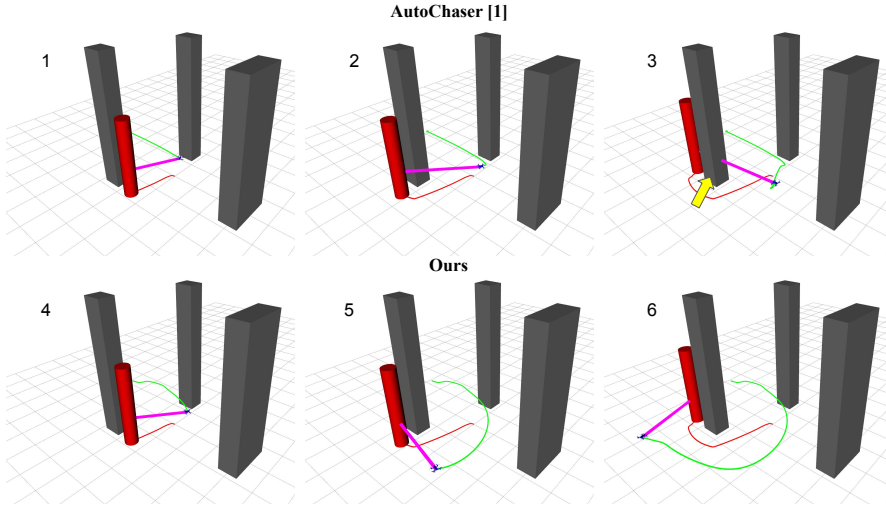
Acceleration Norm: This metric measures how rapidly the quadrotor needs to change its direction and speed to ensure collision/occlusion-free tracking. In other words, the acceleration norm quantifies trajectory smoothness and has been extensively used to benchmark trajectory optimization algorithms. Furthermore, having a smooth trajectory is also critical for applications like drone cinematography to ensure that the images taken of the target are of high-quality [74].

Computation Time: This metric measures the time taken per MPC step and quantifies the real-time applicability of the algorithm.

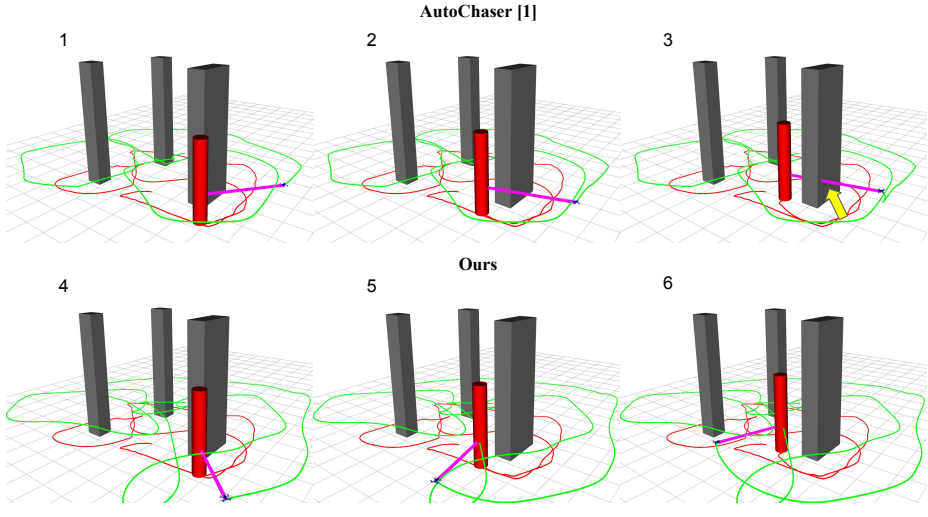
[2] https://github.com/Houman-HM/multi-convex-mpc-target-tarcker

5.4.3. Static Environment Tests

Figure 13 presents the test environment with static obstacles (grey rectangles) and a moving target shown with a red-colored cylinder. The magenta-colored line is the LoS connecting the quadrotor and the target. Different benchmarks are created in this environment by varying the trajectory of the target. The target's trajectories are designed in a way to ensure sharp turns near obstacles to thoroughly test the occlusion avoidance capabilities of the proposed multi-convex MPC and the existing approach [1]. Depicted in Figures 13(a) - 13(b) are the qualitative results obtained across two benchmarks. Note that Figure 13(a) shows only the first few seconds of a long trajectory spread out over several minutes. As can be seen, the proposed multi-convex MPC could leverage fast re-planning to counter the target's motion behind the obstacles and maintain occlusion-free tracking. In contrast, the AutoChaser [1] is not responsive enough, and thus, the target is completely occluded from the FoV of the quadrotor in multiple instances.



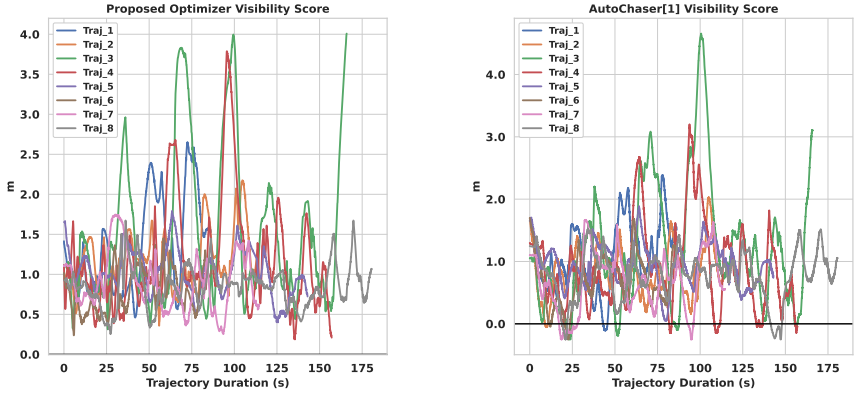
(a)



(b)

Figure 13. The tracking performance of the proposed multi-convex MPC and the state-of-the-art AutoChaser [1] are qualitatively compared in (a) and (b) across two different simulation runs. While AutoChaser gets occluded at multiple instances (shown with yellow arrows), the proposed MPC can react fast to abrupt changes of the target's motion near the obstacles and avoid occlusions. It should be noted that AutoChaser has been provided with information about the global trajectory of the target in the form of intermediate waypoints. In contrast, the proposed MPC only needs information about the instantaneous position and velocity of the target.

Figures 14(a) - 14(b) further quantify the occlusion avoidance in terms of visibility score across eight different trajectories. A value below zero signifies the violation of the occlusion avoidance constraints. As can be seen, the proposed MPC maintains perfect, occlusion-free tracking while in sharp contrast, the visibility score for AutoChaser [1] goes below zero at several time instants.



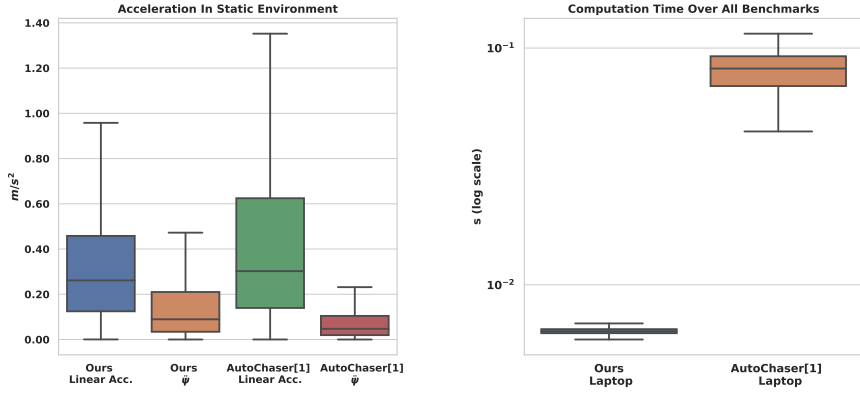
(a) (b)

Figure 14. The visibility score comparison between the proposed multi-convex MPC and the state-of-the-art algorithm AutoChaser [1]. The proposed MPC outperforms the state-of-the-art with visibility scores above zero across all the benchmarks. A visibility score of below zero implies occlusion or alternately non-satisfaction of the occlusion avoidance constraints.

Figure 15(a) compares the acceleration statistics of the proposed MPC with AutoChaser [1] across all benchmarks. The median value of linear acceleration employed by the proposed MPC is $0.26m/s^2$. This is marginally lower than that employed by AutoChaser algorithm [1] standing at $0.30m/s^2$. However, the difference is more pronounced if the maximum acceleration values are compared. The maximum acceleration employed by the proposed MPC across all benchmarks is $0.95m/s^2$. This is 30% lower than that of the AutoChaser's ($1.35m/s^2$). It was observed in the simulation that the higher acceleration of the latter stems from the fact that it has a significant control delay due to its higher computation time; thus it often needs to accelerate sharply to maintain the visibility of the target.

The angular acceleration employed by the proposed MPC is higher than that used by the AutoChaser [1]. The proposed MPC median angular acceleration value is $0.09rad/s^2$, which is two times higher than $0.047rad/s^2$ observed for AutoChaser [1]. The factor of difference between the maximum values is also approximately the same. This pattern in angular acceleration is due to the fact that the proposed MPC aggressively tries to keep the

target at the center of the field of view. AutoChaser, however, adopts a more relaxed approach in maintaining the orientation towards the target.



(a) (b)

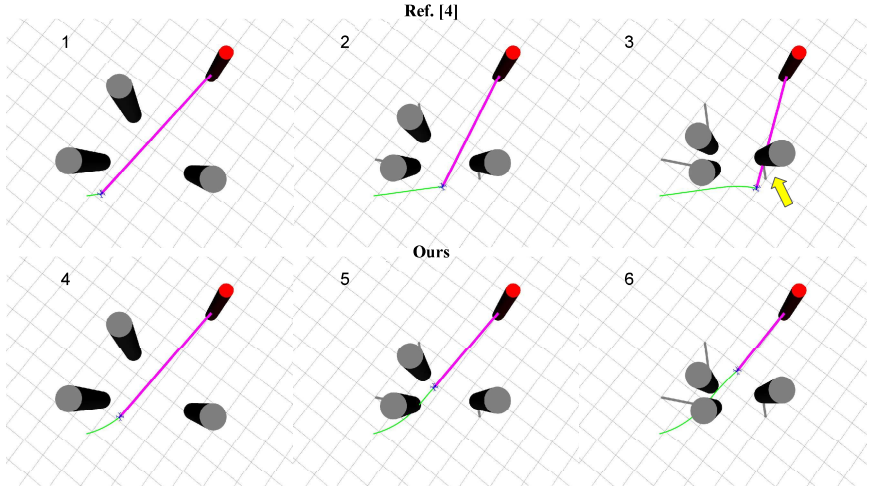
Figure 15. Acceleration profiles and computation time of the proposed MPC and the state-of-the-art AutoChaser[1] are sown in (a) and (b), respectively. The proposed MPC achieves smoother acceleration profiles as well as lower computation times.

Figure 15(b) compares the computation time statistics observed across all benchmarks for the proposed MPC and AutoChaser algorithm [1]. The median computation time for the proposed MPC was $0.006s$ which was more than one order of magnitude smaller than $0.08s$ observed for the AutoChaser. The worst-case timing of AutoChaser was around $0.11s$ while the proposed MPC showed minimal variation across all MPC iterations and always hovered around the median value.

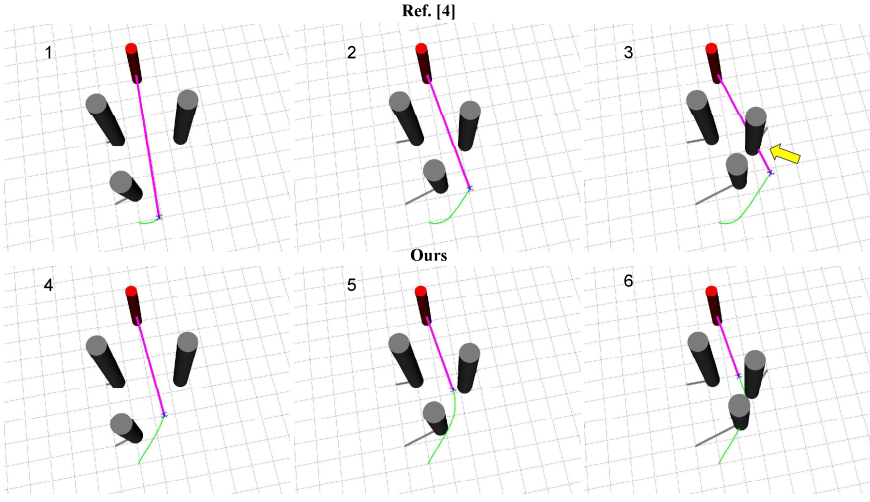
5.4.4. Dynamic Environment Tests

Figure 16 shows the test environments with dynamic obstacles (grey cylinders). For these test cases, the target is considered to be stationary and thus the quadrotor needs to change its position only to avoid occlusion stemming from the dynamic obstacles. However, as it moves, collision avoidance also comes into play. It is worth mentioning that if the occlusion is not factored in, then collision avoidance in these test cases becomes trivial as the quadrotor can just hover at its initial position. The target is chosen to be stationary because it is easier to benchmark the proposed MPC with [4] in this setting. To elaborate further, the target trajectory also needs to avoid collisions with the dynamic obstacles for a meaningful comparison. Thus, it is extremely difficult to recreate the exact same target trajectory within a physics engine for a fair evaluation of the proposed MPC and [4]. Different benchmarks are created for this test case by varying the trajectories of the

dynamic obstacles. Figures 16(a) - 16(b) show the qualitative results obtained across two benchmarks. In both of these benchmarks, the quadrotor needs to find the narrow path among the obstacles and execute it before the obstacles enter its field of view. These represent a very challenging scenario, and as shown, the proposed MPC outperforms that of the [4] in avoiding occlusions.



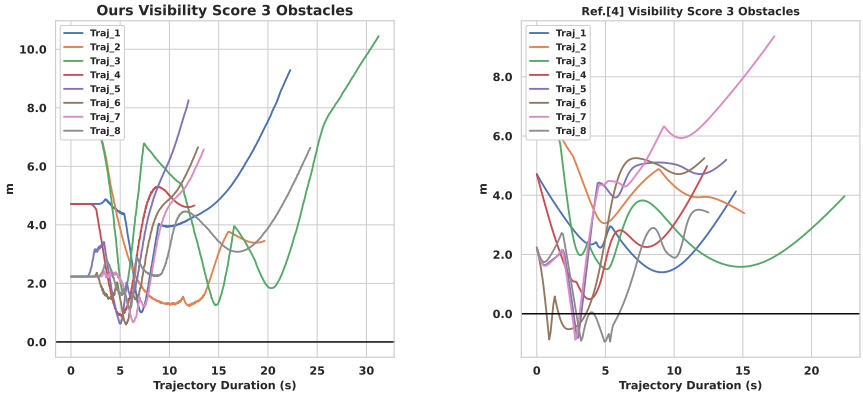
(a)



(b)

Figure 16. The qualitative comparison between the proposed multi-convex MPC and [4]. The environment consists of a static target amongst three dynamic obstacles. The main task for the quadrotor is to continuously adapt its position to prevent occlusion from the dynamic obstacles. The proposed MPC maintains perfect visibility while the trajectories resulting from [4] get occluded at multiple instances.

Figures 17(a) - 17(b) further quantify the performance of the proposed optimizer and that of the [4] in terms of visibility score observed across all benchmarks. The visibility scores along trajectories obtained with the MPC in [4] go below zero at several time instants across different benchmarks, while the proposed optimizer maintains perfect visibility across all the runs.



(a) (b)

Figure 17. The visibility scores achieved by (a) the proposed multi-convex MPC and (b) [4]. The proposed MPC outperforms [4] by maintaining a visibility score of above zero in all the benchmarks. The visibility score is a proxy for the satisfaction of occlusion avoidance constraints. In this particular experiment, the target is static and the simulation runs till the quadrotor reaches an occlusion-free region. Thus, the simulation times of the proposed MPC and that of the [4] are different.

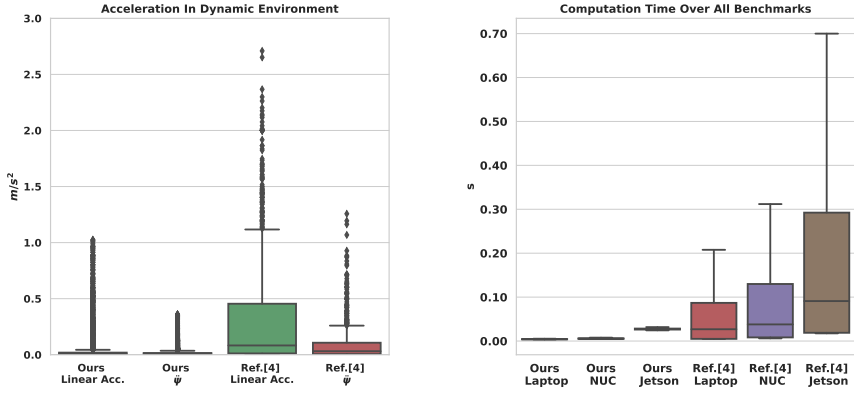
Figure 18(a) presents a comparison of the acceleration profiles observed across all the benchmarks in the considered dynamic environment tests. Due to the very nature of this particular test set-up, it is observed that accelerations generated by both the proposed MPC and [4] follow a bang-bang-like structure. That is, it increases sharply for a brief moment to provide the necessary velocity to the quadrotor followed by a large portion of zero accelerations. Negative accelerations are then applied to bring the quadrotor to rest at the end. Due to this behavior, the median (or even mean) acceleration values would naturally be small and will not provide any meaningful information. Consequently, the comparison is made between the maximum acceleration values of the proposed MPC and [4]. The proposed MPC uses a maximum acceleration of around $1.1m/s$ which is 2.8 times lower than that generated by the [4]. The difference is similar for the angular acceleration with the proposed MPC's maximum value ($0.4rad/s^2$) being 2.5 times lower than that of the [4].

Figure 18(b) presents a comparison of the computation time statistics. In this particular test setup, comparisons are carried out on both laptops

as well as embedded hardware devices such as Intel i5 NUC and NVIDIA Jetson TX2. The laptop is equipped with an i7-8750 processor and 32 GB of RAM.

On the laptop, the proposed MPC's median computation time of $0.004s$ is 5 times less than that required by [4]. However, a more meaningful difference can be obtained by looking at the maximum computation time required as this reflects the worst-case performance. The computation time of the proposed MPC has minimal variation and thus it hovers around the median value of $(0.006s)$. This turns out to be 15 times less than the maximum computation time observed for [4].

The proposed MPC maintains almost identical performance on both laptops and Intel NUC. But the worst-case computation time of [4] increases by 54%. The proposed MPC is also able to maintain a real-time performance on Jetson TX2 with the worst-case computation time of $0.06s$. However, the performance of [4] degrades significantly with the worst-case computation time reaching up to $0.70s$.



(a) (b)

Figure 18. The figure shows the comparison of acceleration profiles and computation time of the proposed MPC with [4]. The proposed MPC outperforms that of [4] in terms f trajectory smoothness by employing smaller acceleration profiles. The proposed MPC can run in real-time on embedded hardware such as Intel NUC and NVIDIA Jetson TX2. The latter is particularly crucial for quadrotors, given their lightweight structure.

5.4.5. Additional Results

In this section, the proposed optimizer is further evaluated for target tracking in an environment with both dynamic obstacles and a moving target. In addition, the proposed optimizer is evaluated for tracking constraint violations for the simulations performed in 5.4.3.

a) ***Target Tracking in Dynamic Environments.*** Figure 19 presents snapshots of a quadrotor tracking a moving target (red cylinder) amongst dynamic obstacles (grey cylinders) using the proposed MPC. The quadrotor is provided with information about only the instantaneous positions and velocities of the target and the obstacles. Thus, it can construct only a rough linear approximation of their future trajectory. Although the actual trajectories for the target and the obstacles are highly non-linear, the fast re-planning ensured by the proposed MPC allows the quadrotor to cope with the prediction errors and ensure occlusion/collision-free tracking.

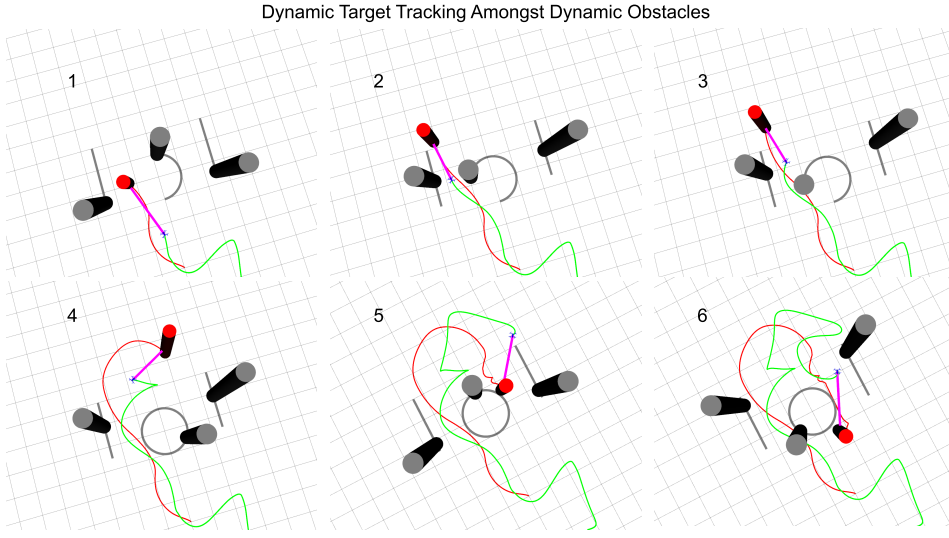


Figure 19. Demonstration of the performance of the proposed MPC while tracking a dynamic target (red) amongst dynamic obstacles (grey).

b) ***Range-Constraints in Target-Tracking.*** For the benchmark considered in Section 5.4.3, the proposed optimizer works under the assumption that the target trajectory is not known. As a result, it is not possible to ensure the satisfaction of the range constraints (4.1d) or (5.1) at all times in a real-time MPC setting. Nevertheless, in Figures 20(a) - 20(h), it is verified how well the proposed optimizer is able to maintain the desired minimum and maximum distance from the target. The min and max values are chosen as $2m$ and $2.5m$, respectively. The results show that despite not having long-term information about the target trajectory, the proposed MPC still adheres to the range thresholds for most of each simulation run. The violation of the minimum and maximum distance thresholds is less than 10cm for a significant portion of the simulation.

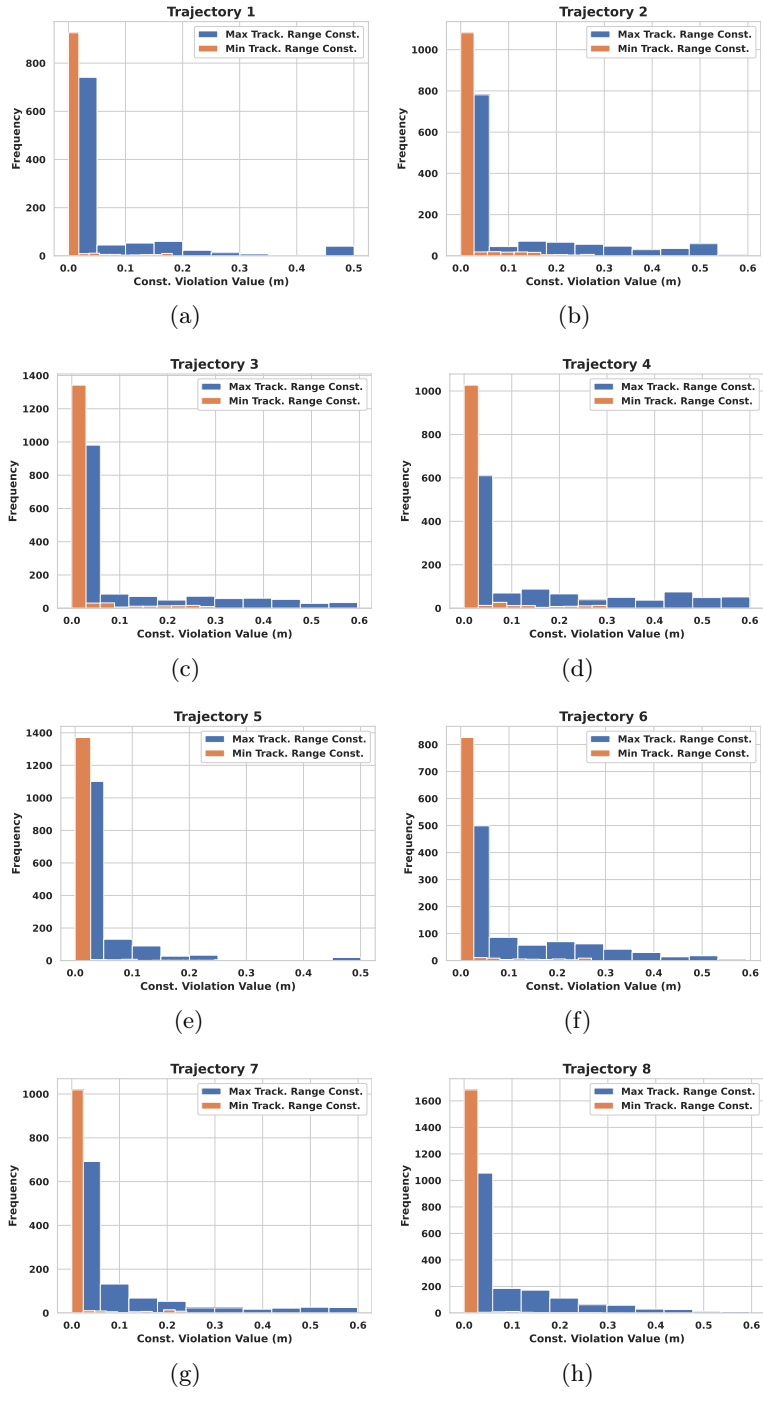
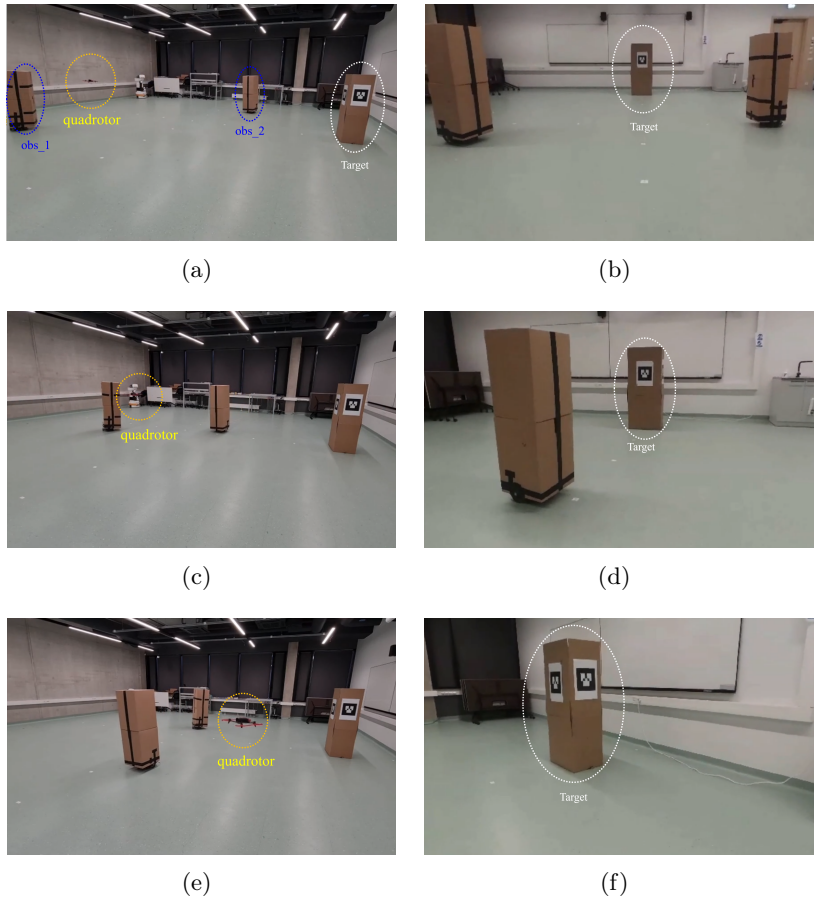


Figure 20. The figure verifies how well the proposed MPC maintains the range specified for tracking constraints (5.2) in the benchmarks considered in Section 5.4.3. As the target trajectory is not known in advance, ensuring the satisfaction of range thresholds at all times is intractable. Thus, the evaluation is done for the extent to which the proposed MPC can minimize the violation of minimum and maximum specified distance for target tracking.

5.4.6. Real World Demonstrator

Figure 21 shows the snapshots from the hardware implementation of the proposed optimizer. The setup is similar to that described in Section 5.4.4 with a static target amongst dynamic obstacles. If the occlusion constraint is ignored, the quadrotor could just hover in place and satisfy the collision-avoidance constraint. In contrast, to avoid occlusions of the quadrotor-target LoS from the dynamic obstacles, the quadrotor needs to constantly change its positions based on the current prediction of the obstacle trajectory. The full video of real-world experiments is also publicly available[3].



(a) (b)

(c) (d)

(e) (f)

Figure 21. Snapshots from the hardware experiment of the proposed optimizer. The setup is similar to the one discussed in Section 5.4.4 with a static target amongst dynamic obstacles. The quadrotor needs to constantly change its position to avoid both occlusion and collision from the dynamic obstacles. The first column shows the third-person view while the second column depicts the view from the quadrotor's onboard camera.

[3]https://doi.org/10.1109/ACCESS.2022.3157977

6. VISIBILITY-AWARE NAVIGATION IN ARBITRARY ENVIRONMENTS

This chapter extends the work of chapter 5 to handle visibility-aware navigation in an environment with obstacles of any geometry. More specifically, the obstacles are detected through LiDAR and resulting point clouds are taken as point obstacles.

6.1. Contributions

The key contributions of this chapter are threefold:

- Using the formulation developed in chapter 5, a novel occlusion model is learned to calculate occlusion based on the laser point cloud, which relaxes the limitations of obstacles to have defined geometries (i.e. ellipsoids).
- A new variant of the sampling-based optimization approach called Cross Entropy Method (CEM) is proposed to optimize over the learned occlusion cost subject to other constraints. In particular, a projection operator is proposed that projects sampled trajectories onto the constraint sets before evaluating the cost on them. It is shown that such augmentation dramatically improves the constraint satisfaction performance within the CEM.
- On top of the proposed CEM method, an MPC is developed that can handle occlusion-free tracking of a target at speeds of up to 1 m/s while outperforming the state-of-the-art in terms of different metrics such as occlusion-avoidance and trajectory quality.
- The learned occlusion model does not require any explicit map building. Thus, it allows the optimizers to adapt on the fly to the changes in the environment. Moreover, it can be queried up to $3\times$ faster than the standard computation of occlusion through distance queries on occupancy or voxel maps [1], [62].
- The proposed optimizers outperform strong baselines [4], [1] in maintaining the target's visibility and success rate while being comparable or better in acceleration effort and computation time.

6.2. Trajectory Optimization

To facilitate the development of the new optimizer, the optimization problem presented in equation (4.1a) is considered in a $2.5D$ scenario, where the UAV operates in $3D$ but can avoid collisions or occlusions only through motions in the $X - Y$ plane.

Consequently, the optimization problem outlined in equation (4.1a) is simplified as follows:

$$\min \sum_{t=t_0}^{t=t_m} \ddot{x}(t)^2 + \ddot{y}(t)^2 + f_{occ}(x(t), y(t), x_r(t), y_r(t)) \tag{6.1a}$$

$$(x(t_0), y(t_0), \dot{x}(t_0), \dot{y}(t_0), \ddot{x}(t_0), \ddot{y}(t_0)) = \mathbf{b}_0 \tag{6.1b}$$

$$(\dot{x}(t_m), \dot{y}(t_m), \ddot{x}(t_m), \ddot{y}(t_m)) = \mathbf{b}_f \tag{6.1c}$$

$$s_{min} \leq \left\| \begin{bmatrix} x(t) - x_r(t) \\ y(t) - y_r(t) \end{bmatrix} \right\|_2 \leq s_{max}, \tag{6.1d}$$

$$f_{vel}(\dot{x}(t), \dot{y}(t)) \leq v_{max}, \tag{6.1e}$$

$$f_{acc}(\ddot{x}(t), \ddot{y}(t)) \leq a_{max} \tag{6.1f}$$

$$f_{vel} = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \tag{6.2}$$

$$f_{acc} = \sqrt{\ddot{x}(t)^2 + \ddot{y}(t)^2} \tag{6.3}$$

using the parametrization presented in 3.3.2 following compact form can be derived for (6.1a) - (6.1f):

$$\min \frac{1}{2} \boldsymbol{\xi}^T \overbrace{\begin{bmatrix} \ddot{\mathbf{W}} & \mathbf{0} \\ \mathbf{0} & \ddot{\mathbf{W}} \end{bmatrix}}^{\mathbf{Q}} \boldsymbol{\xi} + \left\| \mathbf{f}_{occ}(\boldsymbol{\xi}, \mathbf{x}_r, \mathbf{y}_r) \right\|_1 \tag{6.4}$$

$$\mathbf{A}_{eq} \boldsymbol{\xi} = \mathbf{b}_{eq}, \qquad \mathbf{g}(\boldsymbol{\xi}) \leq \mathbf{0}, \tag{6.5}$$

$$\mathbf{A}_{eq} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}, \mathbf{A} = \left[\mathbf{W}_0 | \dot{\mathbf{W}}_0 | \ddot{\mathbf{W}}_0 | \dot{\mathbf{W}}_m | \ddot{\mathbf{W}}_m | \right]^T, \mathbf{b}_{eq} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_f \end{bmatrix} \tag{6.6}$$

The affine constraints (6.5) are the matrix representation of (4.1b) and (4.1c), wherein $\mathbf{W}_j$ represents the j^{th} row of the matrix $\mathbf{W}$. The algebraic form of $\mathbf{g}$ can be easily obtained by substituting (3.5) - (3.6) into (4.1d) - (6.1f). The vector $\mathbf{f}_{occ}$ is formed by stacking the f_{occ} at different time instants. Similar construction is followed to obtain $(\mathbf{x}_r, \mathbf{y}_r)$ from $(x_r(t), y_r(t))$.

The polynomial parametrization guarantees that any randomly generated set of $\boldsymbol{\xi}$ will result in a trajectory that is both continuous and differentiable.

6.2.1. Proposed CEM-based Trajectory Optimizers

a) ***CEM with Soft Inequality Constraints***. One way of handling inequality constraints is to augment them into the cost function in the following manner.

$$\begin{aligned} &\min \frac{1}{2} \boldsymbol{\xi}^T w_1 \mathbf{Q} \boldsymbol{\xi} + w_2 \left\| \mathbf{f}_{occ}(\boldsymbol{\xi}, \mathbf{x}_r, \mathbf{y}_r) \right\|_1 + \sum w_3 \max(\mathbf{0}, -\mathbf{g}(\boldsymbol{\xi})) \\ &\mathbf{A}_{eq} \boldsymbol{\xi} = \mathbf{b}_{eq}, \end{aligned} \tag{6.7}$$

where w_i's are user-defined constants used to trade-off the relative importance of each cost term.

Alg. 1 presents the CEM-based solution of (6.7), wherein the left superscript l is used to track variables across iteration. For example, ${}^{l}\boldsymbol{\mu}$ represents the mean of the sampling distribution at iteration l. On line 4, n samples of $\boldsymbol{\xi}$ are drawn from a Gaussian distribution. The j^{th} drawn sample is denoted as $\boldsymbol{\xi}_j$. On line 5, an empty list is initialized for storing the cost function corresponding to the drawn samples. On lines 7-8, $\boldsymbol{\xi}_j$ is projected to $\overline{\boldsymbol{\xi}}_j$ to satisfy the boundary conditions. The cost is then evaluated for each of the projected samples on lines 9-10 and subsequently the q best-performing samples of $\overline{\boldsymbol{\xi}}_j$ are selected on line 11. These are typically known as the *EliteSet*. On line 12, a Gaussian distribution is fitted to this set, and the resulting mean and covariance would be used for sampling in the next iteration.

Algorithm 1: CEM-I

1 N_{CEM} = Number of CEM iterations
2 Initiate mean ${}^{l}\boldsymbol{\mu}$ and co-variance ${}^{l}\boldsymbol{\Sigma}$, at $l = 0$
3 **for** $l = 1, l \leq N_{CEM}, l++$ **do**
4 Draw n Samples $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_j, ..., \boldsymbol{\xi}_n)$ from $\mathcal{N}({}^{l}\boldsymbol{\mu}, {}^{l}\boldsymbol{\Sigma})$
5 Initialize $CostList = []$
6 Project $\boldsymbol{\xi}_j$ to $\overline{\boldsymbol{\xi}}_j$ through the following optimization $\forall j$:
7 • $\overline{\boldsymbol{\xi}}_j = \arg\min_{\overline{\boldsymbol{\xi}}_j} \left\| \overline{\boldsymbol{\xi}}_j - \boldsymbol{\xi}_j \right\|_2^2$,
8 • $\boldsymbol{A}_{eq}\overline{\boldsymbol{\xi}}_j = \boldsymbol{b}_{eq}$
9 $cost \leftarrow \frac{1}{2}\overline{\boldsymbol{\xi}}_j^T w_1 \mathbf{Q} \overline{\boldsymbol{\xi}}_j + w_2 \|\mathbf{f}_{occ}(\overline{\boldsymbol{\xi}}_j)\|_1 + w_3 \sum \max(\mathbf{0}, -\mathbf{g}(\overline{\boldsymbol{\xi}}_j))$
10 append $cost$ to $CostList$
11 $EliteSet \leftarrow$ Select top q $\overline{\boldsymbol{\xi}}_j$ samples corresponding to lowest cost
12 $({}^{l+1}\boldsymbol{\mu}, {}^{l+1}\boldsymbol{\Sigma}) \leftarrow$ Fit Gaussian distribution $EliteSet$
13 **end**
14 **return** *Sample* $\overline{\boldsymbol{\xi}}_j$ *corresponding to the lowest cost in the EliteSet*

b) ***CEM with Hard Inequality Constraints.*** Here, instead of incorporating inequalities into the cost function, they are maintained in the projection step (lines 7-8). Consequently, some essential modifications are required in the CEM steps compared to Alg. 1. Initially, on line 4 of Alg. 2, $\overline{n}$ samples of $\boldsymbol{\xi}$ are generated and designated as $\overline{\boldsymbol{\xi}}_j$. Typically $\overline{n}$ is much larger than the batch size n over which the cost evaluation would be performed. Second, on line 9, the notion of $ProjEliteSet$ is introduced, which is the set containing n samples of $\overline{\boldsymbol{\xi}}_j$ that performed the best in terms of the projection residual (constraint satisfaction). On line 10, the cost evaluation is performed on the $ProjEliteSet$. Lines 11-13 are the same as Alg. 1.

Algorithm 2: CEM-II

1 N_{CEM} = Number of CEM iterations
2 Initiate mean ${}^{l}\boldsymbol{\mu}, {}^{l}\boldsymbol{\Sigma}$, at $k = 0$
3 **for** $l = 1, l \leq N_{CEM}, l++$ **do**
4 Draw $\overline{n}$ Samples $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_j, ..., \boldsymbol{\xi}_{\overline{n}})$ from $\mathcal{N}({}^{l}\boldsymbol{\mu}, {}^{l}\boldsymbol{\Sigma})$
5 Initialize $CostList = []$
6 Project $\boldsymbol{\xi}_j$ to $\overline{\boldsymbol{\xi}}_j$ through the following optimization $\forall j$:
7 • $\overline{\boldsymbol{\xi}}_j = argmin_{\overline{\boldsymbol{\xi}}_j} \left\| \overline{\boldsymbol{\xi}}_j - \boldsymbol{\xi}_j \right\|_2^2$
8 • $\boldsymbol{A}_{eq}\overline{\boldsymbol{\xi}}_j = \boldsymbol{b}_{eq}, \qquad \boldsymbol{g}(\overline{\boldsymbol{\xi}}_j) \leq 0$
9 $ProjEliteSet \leftarrow$ Select top n samples with lowest projection residuals
10 $cost \leftarrow \frac{1}{2}\overline{\boldsymbol{\xi}}_j^T w_1 \mathbf{Q} \overline{\boldsymbol{\xi}}_j + w_2 \|\mathbf{f}_{occ}(\overline{\boldsymbol{\xi}}_j)\|_1$, over $ProjEliteSet$
11 append $cost$ to $CostList$
12 $EliteSet \leftarrow$ Select top q lowest cost samples
13 $({}^{l+1}\boldsymbol{\mu}, {}^{l+1}\boldsymbol{\Sigma}) \leftarrow$ Fit Gaussian distribution $EliteSet$
14 **end**
15 **return** *Sample corresponding to the lowest* $\overline{\boldsymbol{\xi}}_j$ *in the EliteSet*

6.2.2. Efficient Batch Projection

Alg. 2 contains projection optimizations on lines 7-8 that are independent of each other, creating the potential for parallel processing. Nevertheless, solving numerous non-convex optimizations simultaneously is difficult. Therefore, this section introduces alternative formulations of inequality constraints that simplify the projection optimization and make it amenable to trivially parallelizable tasks such as matrix-vector multiplication. These reformulations are derived from the author's work in Publication IV.

In light of the aforementioned objective, the quadratic tracking constraints (4.1d) are rephrased into the following polar representation.

$$\mathbf{f}_r = \left\{ \begin{array}{c} x(t) - x_r(t) - d_r(t)\cos\alpha_r(t) \\ y(t) - y_r(t) - d_r(t)\sin\alpha_r(t) \end{array} \right\}, s_{min} \leq d_r(t) \leq s_{max} \tag{6.8}$$

where $\alpha_r(t)$ represents the angle that the LoS vector between the robot and the target makes with the X axis. Additionally, the variable $d_r(t)$ represents the magnitude of this vector. Following a similar approach, the quadratic motion bounds can be rephrased in the following form.

$$\mathbf{f}_v = \left\{ \begin{array}{c} \dot{x}(t) - d_v(t)\cos\alpha_v(t) \\ \dot{y}(t) - d_v(t)\sin\alpha_v(t) \end{array} \right\}, d_v(t) \leq v_{max} \tag{6.9}$$

$$\mathbf{f}_a = \left\{ \begin{array}{c} \ddot{x}(t) - d_a(t)\cos\alpha_a(t) \\ \ddot{y}(t) - d_a(t)\sin\alpha_a(t) \end{array} \right\}, d_a(t) \leq a_{max} \tag{6.10}$$

Remark 7. *The variables $\alpha_r(t)$, $\alpha_v(t)$, $\alpha_a(t)$, $d_r(t)$, $d_v(t)$, and $d_a(t)$ are additional variables that will be obtained by the batch projection optimizer along with $\overline{\boldsymbol{\xi}}_j$.*

Matrix Representation: Using the trajectory parametrization in (3.5) - (3.6), constraints (6.8) - (6.10) could be rephrased in the following form.

$$\mathbf{W}\mathbf{c}_x = \mathbf{x}_r + \mathbf{d}_r \cos\boldsymbol{\alpha}_r, \mathbf{W}\mathbf{c}_y = \mathbf{y}_r + \mathbf{d}_r \sin\boldsymbol{\alpha}_r \tag{6.11a}$$

$$\dot{\mathbf{W}}\mathbf{c}_x = \mathbf{d}_v \cos\boldsymbol{\alpha}_v, \dot{\mathbf{W}}\mathbf{c}_y = \mathbf{d}_v \sin\boldsymbol{\alpha}_v, \tag{6.11b}$$

$$\ddot{\mathbf{W}}\mathbf{c}_x = \mathbf{d}_a \cos\boldsymbol{\alpha}_a, \ddot{\mathbf{W}}\mathbf{c}_y = \mathbf{d}_a \sin\boldsymbol{\alpha}_a, \tag{6.11c}$$

where $\boldsymbol{\alpha}_r$ is formed by stacking $\alpha_r(t)$ at different time instants. Similar procedure is followed for constructing $\boldsymbol{\alpha}_v$, $\boldsymbol{\alpha}_a$, $\mathbf{d}_r$, $\mathbf{d}_v$, $\mathbf{d}_a$, $\mathbf{x}_r$, and $\mathbf{y}_r$.

Reformulation of Projection: Using (6.11a) - (6.11c), the projection optimization on lines 7-8 of Alg. 2 could be replaced with the following.

$$\min_{\overline{\boldsymbol{\xi}}_j} \frac{1}{2} \left\| \overline{\boldsymbol{\xi}}_j - \boldsymbol{\xi}_j \right\|_2^2 \tag{6.12a}$$

$$\mathbf{A}_{eq}\overline{\boldsymbol{\xi}}_j = \mathbf{b}_{eq} \tag{6.12b}$$

$$\mathbf{F}\overline{\boldsymbol{\xi}}_j = \mathbf{h}_j(\boldsymbol{\alpha}_{r,j}, \boldsymbol{\alpha}_{a,j}, \boldsymbol{\alpha}_{v,j}\mathbf{d}_{r,j}, \mathbf{d}_{v,j}, \mathbf{d}_{a,j}) \tag{6.12c}$$

$$s_{min} \leq \mathbf{d}_{r,j} \leq s_{max},\ \mathbf{d}_{v,j} \leq v_{max},\ \mathbf{d}_{a,j} \leq a_{max} \tag{6.12d}$$

The subscript j denotes that (6.12a) - (6.12d) are defined for projection of the j^{th} sample of $\boldsymbol{\xi}_j$. In total, there would be $\overline{n}$ such optimization problems. The constraints (6.12c) - (6.12d) together serve as the replacement for $\mathbf{g}(\overline{\boldsymbol{\xi}}_j) \leq 0$. It is worth noting that the matrices $\mathbf{F}$ and $\mathbf{A}_{eq}$ are the same for all the projection problems.

a) ***Solution Process.***

The key idea here is to simplify the non-convex equality constraints (6.12c) by transforming them into l_2 penalties and including them in the projection cost (6.12a).

$$\mathcal{L}(\overline{\boldsymbol{\xi}}_j, \boldsymbol{\lambda}_j) = \frac{1}{2} \left\| \overline{\boldsymbol{\xi}}_j - \boldsymbol{\xi}_j \right\|_2^2 - \langle \boldsymbol{\lambda}_j, \overline{\boldsymbol{\xi}}_j \rangle + \frac{\rho}{2} \left\| \mathbf{F}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j \right\|_2^2 \tag{6.13}$$

The so-called Lagrange multipliers $\boldsymbol{\lambda}_j$ play a crucial role in driving the residuals to zero for any arbitrary ρ [84].

The right-hand side of equation (6.13) is non-convex but has some underlying partially convex structures. These structures become more apparent when using the AM technique to compute the minimum of (6.7) under constraints (6.12b) and (6.12d). This derivation is shown in Alg. 3, where the left superscript is utilized to track variables throughout the iterations. At each step of AM, optimization is performed over only one particular variable, while the remaining variables are kept fixed at values obtained in the previous iteration or the previous step of the current iteration. A more comprehensive analysis is provided in the following subsection.

*b) **Analysis**.*

Step (6.16): This optimization is a convex-equality-constrained QP. It can be simplified into the following system of linear equations, where $\boldsymbol{\nu}_j$ represents the dual variable linked to the equality constraints.

$$\overbrace{\begin{bmatrix}\mathbf{I}+\rho\mathbf{F}^T\mathbf{F} & \mathbf{A}_{eq}^T\\ \mathbf{A}_{eq} & \mathbf{0}\end{bmatrix}}^{\mathbf{Q}_{project}}\begin{bmatrix}\overline{\boldsymbol{\xi}}_j\\ \boldsymbol{\nu}_j\end{bmatrix}=\overbrace{\begin{bmatrix}\rho\mathbf{F}^T{}^k\mathbf{h}_j+{}^k\boldsymbol{\lambda}_j+\boldsymbol{\xi}_j\\ \mathbf{b}_{eq}\end{bmatrix}}^{\mathbf{q}_j} \tag{6.14}$$

The set of equations (6.14) computes the projection of the j^{th} sample. However, as the left-hand side of (6.14) would be identical for all the projection optimizations, the solution for the entire batch could be obtained simultaneously through the following equation.

$$\begin{bmatrix}\overline{\boldsymbol{\xi}}_1,\boldsymbol{\nu}_1\\ \vdots\\ \overline{\boldsymbol{\xi}}_{\overline{n}},\boldsymbol{\nu}_{\overline{n}}\end{bmatrix}=\left(\mathbf{Q}_{project}^{-1}\left[\mathbf{q}_1|\mathbf{q}_2|\mathbf{q}_3\dots\mathbf{q}_{\overline{n}}\right]\right)^T \tag{6.15}$$

Remark 8. *The inverse of $\boldsymbol{Q}_{project}$ needs to be performed only once. Therefore, during each iteration of Alg. 3, only computing large matrix-vector products is required in equation* (6.15)*, which can be easily parallelized on GPUs.*

Step (6.17): The ${}^{k+1}\overline{\boldsymbol{\xi}}_j$ obtained at step (6.16) can be mapped to the j^{th} sampled trajectory $({}^{k+1}\mathbf{x}_j, {}^{k+1}\mathbf{y}_j)$ using (3.5) - (3.6). This allows us to consider each element of $\boldsymbol{\alpha}_{r,j}$ as independent of the others. As a result, the optimization problem in (6.17) could be divided into m separate problems, each with a closed-form solution.

$${}^{k+1}\boldsymbol{\alpha}_{r,j}=\arctan 2(({}^{k+1}\mathbf{y}_j-\mathbf{y}_r),({}^{k+1}\mathbf{x}_j-\mathbf{x}_r)) \tag{6.22}$$

Step (6.18): Similar to the previous step, each element of $\mathbf{d}_{r,j}$ can be considered independent Consequently, optimizing over $\mathbf{d}r, j$ reduces to solving n parallel single-variable QP problems subject to simple bound constraints. A closed-form solution could be obtained by first solving the unconstrained problem and then simply clipping the value to lie between $[s_{min}\ s_{max}]$.

Step (6.20): The two optimizations in this step have the same structure as (6.17) - (6.18) and thus have a closed-form solution.

Remark 9. *The solution of optimizations (6.17) - (6.20) involves only element-wise operations. Such numerical operations could be easily computed for the entire batch in one shot.*

6.2.3. Learned Occlusion Model

In this section, the final component of the proposed CEM pipeline is introduced: an occlusion cost that can be queried efficiently in a batch on GPUs. This is accomplished by using the same formula as presented in

Algorithm 3: Efficient Batch Projection

1: Initialize ${}^{k}\boldsymbol{\alpha}_{r,j}, {}^{k}\boldsymbol{\alpha}_{v,j}, {}^{k}\boldsymbol{\alpha}_{a,j}, {}^{k}\mathbf{d}_{r,j}, {}^{k}\mathbf{d}_{v,j}, {}^{k}\mathbf{d}_{a,j}$ for iteration $k = 0$

2: **while** $k \leq$ *max projection iter., or till norm of residuals are below some threshold* **do**

$$
\begin{aligned}
{}^{k+1}\overline{\boldsymbol{\xi}}_j = \arg\min_{\overline{\boldsymbol{\xi}}_j} \frac{1}{2} \left\| \overline{\boldsymbol{\xi}}_j - \boldsymbol{\xi}_j \right\|_2^2 - \langle {}^{k}\boldsymbol{\lambda}_j, \overline{\boldsymbol{\xi}}_j \rangle + \\
\frac{\rho}{2} \left\| \mathbf{F}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j({}^{k}\boldsymbol{\alpha}_{r,j}, {}^{k}\boldsymbol{\alpha}_{v,j}, {}^{k}\boldsymbol{\alpha}_{a,j}, {}^{k}\mathbf{d}_{r,j}, {}^{k}\mathbf{d}_{v,j}, {}^{k}\mathbf{d}_{a,j},) \right\|_2^2, \\
\mathbf{A}_{eq}\overline{\boldsymbol{\xi}}_j = \mathbf{b}_{eq}
\end{aligned} \tag{6.16}
$$

$$
{}^{k+1}\boldsymbol{\alpha}_{r,j} = \arg\min_{\boldsymbol{\alpha}_{r,j}} \left\| \mathbf{F}^{k+1}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j \right\|_2^2 = \arg\min_{\boldsymbol{\alpha}_{r,j}} \left\| \begin{matrix} {}^{k+1}\mathbf{x}_j - \mathbf{x}_r - {}^{k}\mathbf{d}_{r,j}\cos\boldsymbol{\alpha}_{r,j} \\ {}^{k+1}\mathbf{y}_j - \mathbf{y}_r - {}^{k}\mathbf{d}_{r,j}\sin\boldsymbol{\alpha}_{r,j} \end{matrix} \right\|_2^2 \tag{6.17}
$$

$$
{}^{k+1}\mathbf{d}_{r,j} = \arg \min_{s_{min} \leq \mathbf{d}_{r,j} \leq s_{max}} \left\| \mathbf{F}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j({}^{k+1}\boldsymbol{\alpha}_{r,j}) \right\|_2^2 \tag{6.18}
$$

$$
\begin{aligned}
{}^{k+1}\boldsymbol{\alpha}_{v,j} = \arg\min_{\boldsymbol{\alpha}_{v,j}} \left\| \mathbf{F}^{k+1}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j({}^{k}\mathbf{d}_{v,j}, \boldsymbol{\alpha}_{v,j}) \right\|_2^2, \\
{}^{k+1}\mathbf{d}_{v,j} = \arg \min_{\mathbf{d}_{v,j} \leq v_{max}} \left\| \mathbf{F}^{k+1}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j({}^{k+1}\boldsymbol{\alpha}_{v,j}, \mathbf{d}_{v,j}) \right\|_2^2
\end{aligned} \tag{6.19}
$$

$$
\begin{aligned}
{}^{k+1}\boldsymbol{\alpha}_{a,j} = \arg\min_{\boldsymbol{\alpha}_{a,j}} \left\| \mathbf{F}^{k+1}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j({}^{k}\mathbf{d}_{a,j}, \boldsymbol{\alpha}_{a,j}) \right\|_2^2, \\
{}^{k+1}\mathbf{d}_{a,j} = \arg \min_{\mathbf{d}_{a,j} \leq a_{max}} \left\| \mathbf{F}^{k+1}\overline{\boldsymbol{\xi}}_j - \mathbf{h}_j({}^{k+1}\boldsymbol{\alpha}_{a,j}, \mathbf{d}_{a,j}) \right\|_2^2
\end{aligned} \tag{6.20}
$$

$$
({}^{k+1}\boldsymbol{\lambda}_j) = ({}^{k}\boldsymbol{\lambda}_j) - \rho\mathbf{F}^T(\mathbf{F}\overline{\boldsymbol{\xi}}_j - {}^{k+1}\mathbf{h}_j) \tag{6.21}
$$

3: Return ${}^{k+1}\overline{\boldsymbol{\xi}}_j$

(5.9) to define the LoS at time t as a straight line connecting the robot and the target position. Mathematically, any point on the 2D LoS can be identified by selecting an appropriate value of $u \in [0, 1]$ in (6.23).

$$
(x_{los}, y_{los})(t, u) = \begin{bmatrix} (1-u)x(t) + ux_r(t) \\ (1-u)y(t) + uy_r(t) \end{bmatrix} \tag{6.23}
$$

The points on the LoS are a function of $(x(t), y(t))$ and consequently $\boldsymbol{\xi}$. Suppose a stationary circular obstacle located at (x_o, y_o) with a radius of a, which takes into account the robot's size and inflation. The proposed occlusion cost is defined as follows:

$$
f_{occ}(x(t), y(t), x_r(t), y_r(t)) = \sum_u \{\max(0, -d_{los}(t, u))\}^2 \tag{6.24}
$$

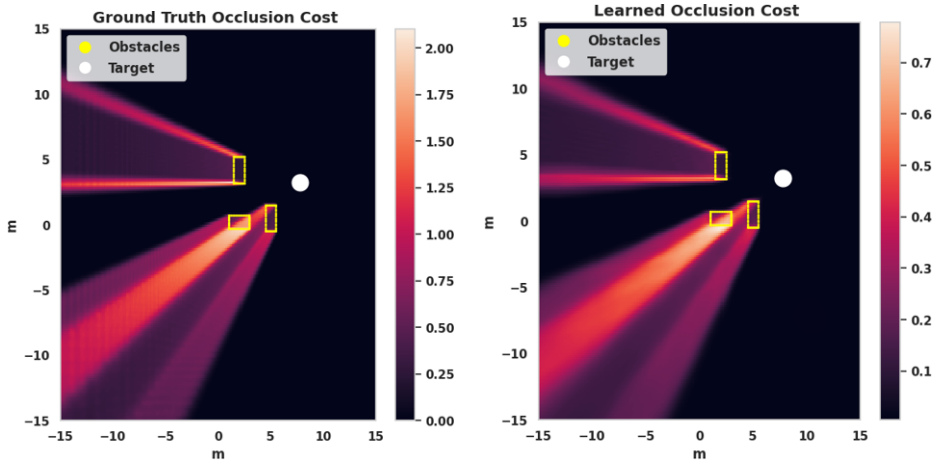
$$
d_{los}(t, u) = (x_{los} - x_o)^2 + (y_{los} - y_o)^2 - a^2 \tag{6.25}
$$

It can be inferred from equation (6.24) that f_{occ} is influenced by the distance between the points on the LoS vector and a specific obstacle (recall Figure 8). The collision-avoidance of the robot with the environment is ensured when f_{occ} is evaluated for $u = 0$.

a) ***Reducing distance queries through learning.*** To calculate (6.24), the variable u is discretized within the range of 0 to 1 and $d_{los}(t, u)$ is evaluated at each of these discrete points. There are two common methods to compute these distances. The first involves obtaining them from the occupancy or voxel map of the environment, but this requires constructing an explicit map and must be repeated if the environment changes. The second approach computes distance queries from point obstacles generated from LiDAR scans, but this can be prohibitively expensive in terms of both memory and computation. To illustrate, suppose that the resolution of u is n_u, and the number of point obstacles is n_o. For a batch of n trajectories and m planning steps, the number of distance queries required would be $(n_u * n_o * n * m)$. Using the values $n_u = 100$, $n_o = 200$, $n = 200$, and $m = 100$, this amounts to $4 * 10^8$ distance queries. In addition, there is an overhead associated with generating the same number of (x_{los}, y_{los}) prior to performing the distance queries. Dealing with such a large number of samples can become computationally infeasible.

In order to address the issues mentioned above, a learning approach is proposed for occlusion computation that utilizes LiDAR-generated point obstacles and reduces the distance queries by a factor of n_u (approximately 100). The key idea is to avoid the process of querying $d_{los}(t, u)$ for a specific set of discrete u values and instead approximate its effects via a neural network. To achieve this, a Multi-Layer Perceptron (MLP) neural network is trained to learn the right-hand side of (6.24). Specifically, the MLP takes as input the positions of the robot, target, and a single obstacle and outputs the occlusion value. The inferencing with the MLP could be done in batches to query the occlusion cost for the entire CEM batch and all obstacles simultaneously.

Figure 22 shows the comparison between the ground truth and learned occlusion model for an arbitrary environment. The obstacle boundaries are represented through an overlap of point obstacles. Although the exact analytical values are different, the learned model is able to correctly identify the areas from where the target is not visible.



(a) (b)

Figure 22. Occlusion heatmap. (a) is the ground truth and (b) is predicted by the proposed learned occlusion cost.

Figure 23 compares the time required to query $d_{los}(t, u)$ from an occupancy map and computing the r.h.s of (6.24) with the proposed learned occlusion cost. The Signed Distance Field (SDF) library is utilized to query the distances from the occupancy map, and multi-threaded CPU parallelization is used to execute the queries in parallel. Despite these customizations, the learned occlusion cost is shown to be three times faster than the one based on the occupancy map.

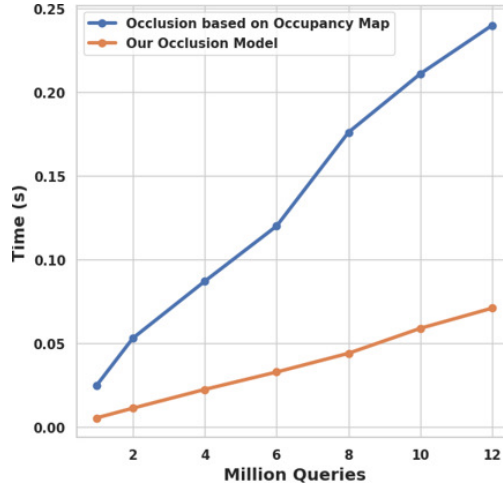


Figure 23. The Figure shows the time taken to perform occlusion cost queries from an occupancy map using the SDF library compared with the proposed learned occlusion model.

6.3. Validation and Benchmarking

In this section, the optimizers in Alg. 1 and 2 are validated in various settings, and the results are benchmarked against different state-of-the-art approaches in static and dynamic environments.

6.3.1. Implementation Details

The trajectory optimizers are implemented in Python using the JAX [92] library as the GPU-accelerated linear algebra backend. The simulation pipeline is built on top of ROS [91] and the physics simulator Gazebo. AORUS Laptop Intel Core $i7-11800H$ with Nvidia RTX 3080 is used to execute all the benchmarks.

For the MLP-based occlusion cost, the ground-truth data is derived from the analytical model (6.24) and trained in PyTorch with AdamW optimizer. 400000 (x, y) points paired with 100 random obstacle positions are used to create a dataset of 40 million instances. The architecture details of the learned occlusion model are presented in Table 2.

Table 2. Occlusion Model Architecture

Layers	Dimension	Activation
1. Input	6 x 512	Relu
2. Hidden	512 x 256	Relu
3. Hidden	256 x 128	Relu
4. Output	128 x 1	Linear

The MPC pipeline developed on top of the optimizers takes the robot and target's current positions and velocities along with LiDAR point clouds to generate occlusion-free trajectories in real-time.

a) Hyper parameters. The batch size of CEM in Alg. 1 and Alg. 2 are chosen as 500 and 200, respectively. However, in Alg. 2, $\overline{n} = 1000$ trajectories are sampled initially and then the top 200 among them are chosen based on projection residuals.

b) Baselines. The same baselines used in 5.4.3 - 5.4.4, namely **AutoChaser** [1] and **Nägeli et al** [4] are considered for benchmarking in static and dynamic environments, respectively.

c) Metrics. Following metrics are considered for benchmarking.

Occlusion Time: It is defined as the total time during which the robot and the target LoS are in collision with an obstacle.

Acceleration: A measure to determine how fast the robot changes its speed and direction while following a target to avoid collisions or occlusion stemming from the obstacles.

Computation Time: Time taken to compute a solution trajectory.

Success Rate: Total number of occlusion-free runs divided by the total number of runs in each benchmark.

6.3.2. Point-to-Point Navigation

This benchmark considers point-to-point navigation with target-visibility constraints in a known environment. The obstacles are given through a point cloud (shown in yellow in Figure 24). An additional constraint (lower bound constraint in (6.1d)) is added to guarantee that the optimal trajectory does not collide with the target.

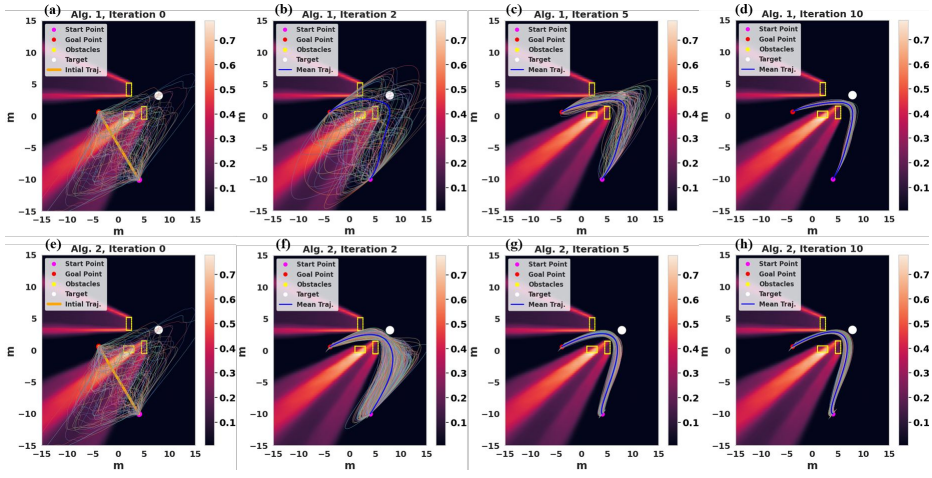


Figure 24. (a) - (d) depict the visibility-aware trajectories generated using Alg. 1 while Figures (e) - (h) show the trajectories computed by Alg. 2. Although both algorithms work, from iteration 2 onwards, Alg. 2 ensures that all the sampled trajectories satisfy collision avoidance with the target.

Figure 24 illustrates that both Alg. 1 and Alg. 2 gradually converge to a trajectory that is free from occlusions and collisions after being initialized with a simple Gaussian distribution. However, Alg. 1 takes ten iterations to obtain an occlusion-free trajectory that also avoids collisions with the target. On the other hand, Alg. 2 can obtain a large number of samples that avoid collisions with the target from the second iteration itself, thanks to the projection step (lines 7-8). Due to this behavior, Alg. 2 obtains a solution trajectory in roughly half the time required by that of Alg. 1. This feature of constraint satisfaction is even more crucial in the real-time MPC setting, where only a few iterations of the CEM can be performed before executing the best available action.

6.3.3. Static Environments

This section benchmarks the developed MPC pipeline for target-tracking application in a static environment. 10 different target trajectories are generated with speeds of 0.5 to 1 m/s in two different environment configurations. The qualitative results are shown in Figure 25.

As shown in Table 3, AutoChaser [1] performs very poorly in terms of

occlusion time and success rate when it does not have prior access to target intermediate waypoints. In fact, in none of the 10 trajectories, this variant of [1] managed to remain completely occlusion-free. The performance improves dramatically when AutoChaser is provided with a dense set of target waypoints. However, even in this scenario, AutoChaser is outperformed by both of the proposed optimizers regarding success rate and occlusion time. The optimizer presented in Alg. 2 achieves near-perfect visibility and success rate. All the compared approaches required similar acceleration efforts. Alg. 1 and AutoChaser[1] have similar computation times while Alg. 2 is approximately 40% faster than the baseline. The timing difference between Alg. 1 and 2 is interesting since, on the surface, Alg. 2 appears to have more computations due to the presence of the batch projection operation. However, the batch projection innovates at the matrix algebra level to leverage GPUs for fast distributed computations.

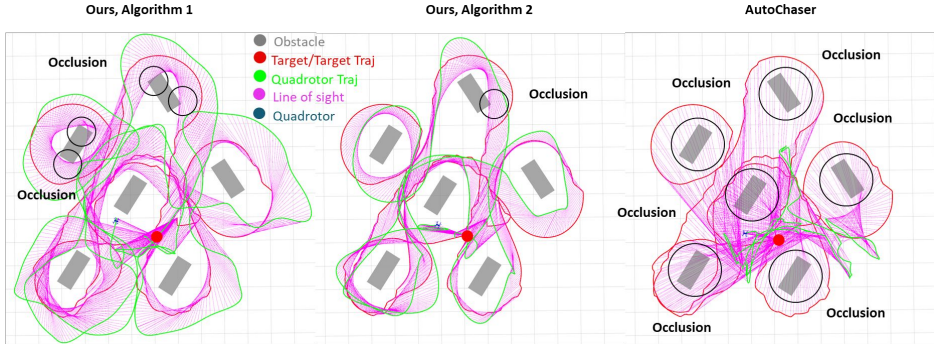


Figure 25. A qualitative comparison of target tracking is conducted in an environment with 6 static obstacles. As observed, the baseline AutoChaser [1] is outperformed by both Alg. 1 and 2. Furthermore, Alg. 2 exhibits the best performance with near-perfect occlusion-free tracking.

Table 3. Total Target Trajectories: 10 / Sim. Time: 1290s

Method	Occlusion Time (s) / Succes-rate	Linear Acceleration Mean / Min / Max	Computation Time (s)
Alg. 1	6.7 / 0.70	0.65 / 0 / 1.75	0.11
Alg. 2	**0.7 / 0.95**	**0.432 / 0.00 / 1.26**	**0.06**
AutoChaser [1], no prior target waypoints	72.68 /0.0	0.312 / 0.00 / 1.14	0.085
AutoChaser [1], 100 prior target waypoints	45.42 /0.10	0.478 / 0.00 / 1.35	0.085

6.3.4. Dynamic Environment

In this section, the performance of the proposed optimizers is compared to [4]. Eight different configurations with dynamic obstacles, as in Section 5.4.4, are considered for this benchmark. The obstacles are considered cylindrical to make it compatible with the assumption in [4], However, for the proposed optimizers, the obstacles are given through a point cloud without any notion of their shapes. The target is considered to be stationary in this benchmark for a meaningful comparison. To elaborate, since the target trajectory also needs to avoid collision with the dynamic obstacles, it is extremely difficult to replicate the same trajectory for a fair comparison of different algorithms within a physics simulation engine. Hence, different configurations are using obstacles with varying velocities.

Figure 26 shows the qualitative results in one benchmark. Here, the quadrotor needs to keep its view of the target occlusion-free while avoiding dynamic obstacles. As can be seen, the baseline gets occluded in different instances, while the proposed Alg. 2 maintains perfect visibility of the target. The quantitative results for all the configurations are summarized in Table 4. The baseline [4] is outperformed by both of the proposed optimizers in occlusion time and success rate metrics. The performance gap between the proposed approaches and that of [4] could be attributed to its simplified occlusion model that does not explicitly account for obstacle geometry.

Table 4. Total Obstacle Configurations: 8 / Sim. Time: 240s

Method	Occlusion Time(s) / Success rate	Linear Acc. Mean / Max	Computation Time (s)
Alg. 1	**0 / 1.0**	1.06 / 4.58	0.12
Alg. 2	**0 / 1.0**	0.98 / 1.43	0.06
Nägeli et al [4]	6.3 /0.375	0.4 / 1.7	0.022 (warm-start 0.37)

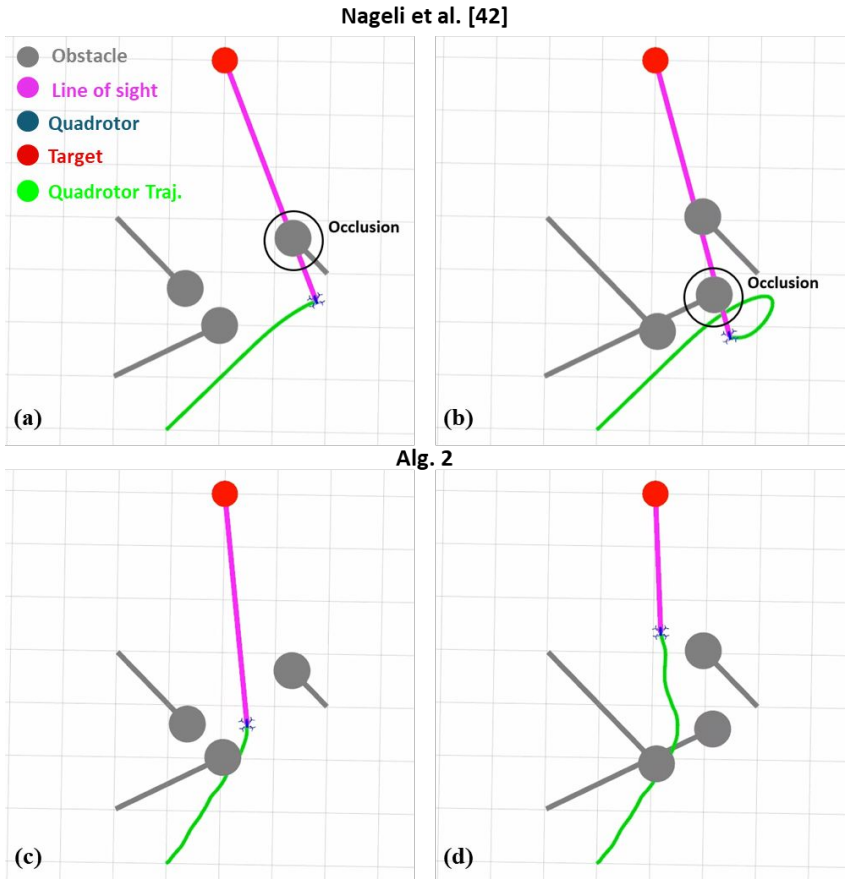


Figure 26. A qualitative comparison between Alg. 2 and [4]. The environment consists of a static target amongst three dynamic obstacles. The task is for the quadrotor to continuously adapt its position to prevent occlusion from the dynamic obstacle. As can be seen, [4] gets occluded at several instances (shown in (a) - (b)) while the proposed Alg. 2 maintains perfect visibility (depicted in (c) - (d)).

The behavior observed in [4] is similar to a bang-bang controller. The robot used sparse bursts of sharp accelerations to reach specific velocities. As a result, the average acceleration of [4] is lower than those of the proposed optimizers. However, its max acceleration is 20% higher than that employed by the proposed optimizer in Alg. 2. Although the MPC iterations of [4] are faster than that achieved by the proposed optimizers, when the warm-up time is taken into account, the proposed approaches are more tractable for real-time performance. In fact, [4] needs a separate warm start for each obstacle configuration and fails when the obstacle trajectories change drastically in the middle of the runs. For the same reason, the author could not make [4] work in scenarios with a dynamic target. Note that the proposed optimizers do not need/use any of such warm starts.

6.3.5. Online Obstacle Adaptation

Here, the proposed optimizer in Alg. 2 is validated in terms of adapting to obstacles on the fly. Figure 27(a) shows a scenario where the quadrotor is initially hovering at its place since its view of the target is free of occlusions. However, suddenly two obstacles are added to the environment (shown with yellow arrows in Figure 27(b)) and occlude its LoS. Thus, the quadrotor moves to free its LoS from occlusion. Note, how the quadrotor moves to the bottom and also avoids collision with the newly added obstacle at the bottom (Figures 27 (c-d)). This behavior is achievable thanks to the use of online point cloud by the proposed optimizers.

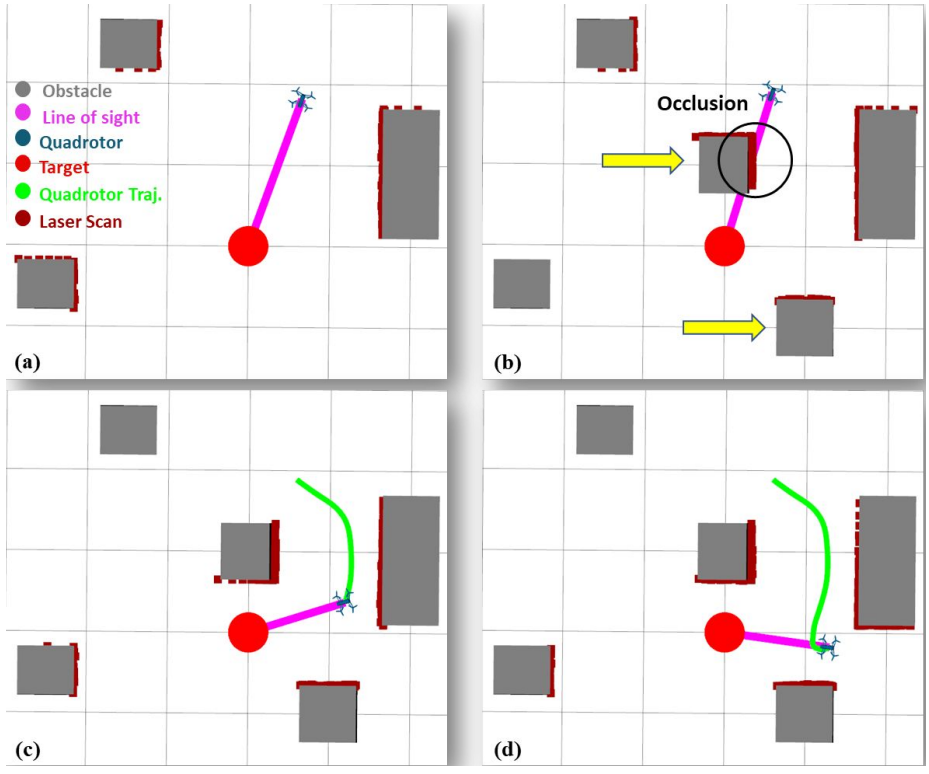


Figure 27. Online obstacle adaptation of the proposed optimizer in Alg. 2. (a) shows the initial configuration with three obstacles. Two new obstacles are suddenly added to the environment (shown with yellows arrows in (b)) and occlude the quadrotor's LoS. The quadrotor moves to the bottom to clear its LoS from occlusions (shown in (c)). Notice how the quadrotor also avoids the new obstacle on the bottom (shown in (d)).

6.3.6. Real World Demonstrator

This section presents the qualitative results obtained from the real-world experiment of the proposed optimizers. The quadrotor used is Parrot Bebop 2 and Optitrack Motion Capture System is utilized for quadrotor and obstacles localization. Figure 28 illustrates an environment with three obstacles and a moving human target. The obstacle point cloud is generated around obstacles given the center of each obstacle (shown with brown in the RViz views). While moving, the human target displaces the obstacles, leading to new obstacle configurations (indicated with yellow arrows in Figures (e) and (h)). The quadrotor is able to counter these changes and continue tracking the target without losing the LoS thanks to the utilization of an online point cloud that is updated as the obstacles move. The full video of the runs is also released publically[1].

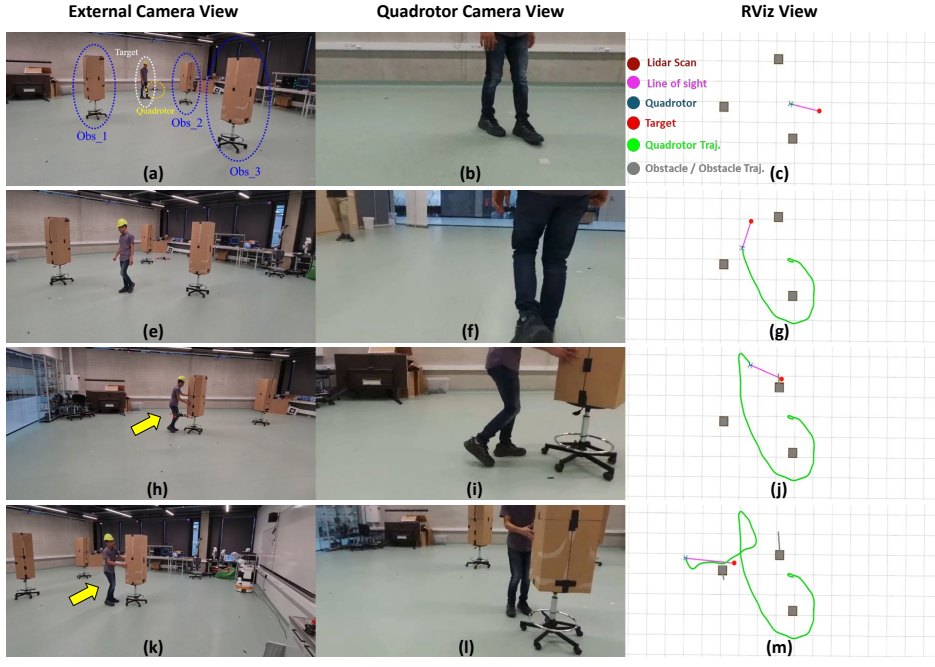


Figure 28. The qualitative comparison of a hardware experiment of the proposed optimizers. The environment consists of three obstacles and a moving human target. The human target rearranges the obstacles throughout the run and creates new configurations. (shown with yellow arrows)

[1] https://www.youtube.com/watch?v=jNCdLur_NaY

7. COOPERATIVE VISIBILITY-AWARE NAVIGATION IN ARBITRARY ENVIRONMENTS

This chapter extends the proposed optimization-based algorithms for VAN to a cooperative multi-robot setting, with applications to warehouse inventory management. More specifically, a trajectory planning algorithm is developed for a team of a UAV and a UGV to autonomously scan warehouse racks for inventory management. The main motivation is to make small-sized UAVs with limited computing and sensing hardware capable of reliably performing the scanning task in cluttered environments.

7.1. Contributions

The key contributions of this chapter are summarized as follows:

- Visibility-Aware Cooperative Navigation (VACNA) is introduced for enabling UGVs to guide the navigation of small-sized UAVs. Specifically, VACNA is characterized by a set of cost and constraint functions that allow a UAV-UGV pair to navigate smoothly in cluttered environments while maintaining each other in their field of view and within a specified distance threshold. Moreover, these functions also model the primary task: UAV maintaining visibility of the rack during navigation.
- A CEM-based optimizer is developed to solve the VACNA problem in completely unknown environments using only LiDAR point cloud.
- A deep generative model is learned in form of a CVAE from expert demonstrations to warm-start the proposed CEM-based optimizer.
- Using the CVAE initialization, the optimizer achieves real-time performance for solving the VANCA problem on embedded devices such as NVIDIA Jetson Xavier.
- The proposed CEM optimizer outperforms the baseline variants in all benchmark metrics such as occlusion time, traversal distance, control effort, and computation time.

7.2. VACNA with Application in Warehouse Inventory

One of the prominent applications for UAVs is warehouse automation [93],[94], where a UAV needs to scan the racks of the warehouses through its monocular camera while navigating through narrow, cluttered corridors. From the perspective of autonomous navigation, there are two core challenges to this task. First, UAVs have limited computing power onboard, and second, mounting sensors like LiDAR and RGBD cameras on the UAV

can increase its form factor and reduce its flight time, making it unsuitable for navigation in tight indoor spaces.

This chapter aims to develop an approach to make small-sized UAVs with only monocular cameras capable of reliably performing rack scanning tasks in warehouses. A cooperative system is proposed and validated, wherein a UAV is guided by a UGV with a more significant form factor that can carry more extensive computing and sensing hardware to plan motions for itself and the UAV.

At the core of the approach lies a novel motion planning concept referred to as VACNA, characterized by cost and constraint functions that allow a UAV-UGV pair to navigate smoothly in cluttered environments while maintaining each other in their FoV and within a specified distance threshold. Moreover, these functions also model the primary task of maintaining the visibility of racks during navigation.

A CEM-based approach is adopted to solve the trajectory optimization underpinning VACNA. Several modifications over the baseline variant are proposed, significantly improving the quality of cooperation between the UAV-UGV and the efficiency of the scanning task.

7.3. Trajectory Optimization for UAV-UGV Warehouse Inventory

Assuming that both UAV-UGV are holonomic and the obstacles can be avoided in 2D through motions in $X-Y$ plane, the optimization problem for VACNA for a UAV-UGV pair can be formalized as in (7.1a) - (7.1h). Note that the UAV still operates in 3D, but its height is independently controlled.

$$\min \sum_{t=t_0}^{t=t_m} w_1 \ddot{\mathbf{p}}_r(t)^2 + w_1 \ddot{\mathbf{p}}_g(t)^2 + w_2 f_{occ}(\mathbf{p}_r(t), \mathbf{p}_g(t)) + w_3 f_{occ}(\mathbf{p}_r(t), \mathbf{p}_{sc}(t)) \quad (7.1a)$$

$$(\mathbf{p}_r(t_0), \dot{\mathbf{p}}_r(t_0), \ddot{\mathbf{p}}_r(t_0)) = \mathbf{b}_{0,r}, (\dot{\mathbf{p}}_r(t_m), \ddot{\mathbf{p}}_r(t_m)) = \mathbf{b}_{f,r} \quad (7.1b)$$

$$(\mathbf{p}_g(t_0), \dot{\mathbf{p}}_g(t_0), \ddot{\mathbf{p}}_g(t_0)) = \mathbf{b}_{0,g}, (\dot{\mathbf{p}}_g(t_m), \ddot{\mathbf{p}}_g(t_m)) = \mathbf{b}_{f,g} \quad (7.1c)$$

$$f_{vel}(\dot{\mathbf{p}}_r(t)) \leq v_r, f_{acc}(\ddot{\mathbf{p}}_r(t)) \leq a_r \quad (7.1d)$$

$$f_{vel}(\dot{\mathbf{p}}_g(t)) \leq v_g, f_{acc}(\ddot{\mathbf{p}}_g(t)) \leq a_g \quad (7.1e)$$

$$f_{o,i}(\mathbf{p}_r, \mathbf{p}_{o,i}) \leq 0, f_{o,i}(\mathbf{p}_g, \mathbf{p}_{o,i}) \leq 0, \forall i \quad (7.1f)$$

$$\underline{s}_{los} \leq \left\| \left[\mathbf{p}_r(t) - \mathbf{p}_g(t) \right] \right\|_2 \leq \overline{s}_{los} \quad (7.1g)$$

$$\underline{s}_{ref} \leq \left\| \left[\mathbf{p}_r(t) - \mathbf{p}_{ref}(t) \right] \right\|_2 \leq \overline{s}_{ref} \quad (7.1h)$$

$$f_{vel} = \sqrt{\dot{\mathbf{p}}_{(.)}(t)^2}, f_{acc} = \sqrt{\ddot{\mathbf{p}}_{(.)}(t)^2} \quad (7.2)$$

$$f_{o,i} = -\frac{(x_{(.)} - x_{o,i})^2}{a^2} - \frac{(y_{(.)} - y_{o,i})^2}{a^2} + 1 \leq 0 \quad (7.3)$$

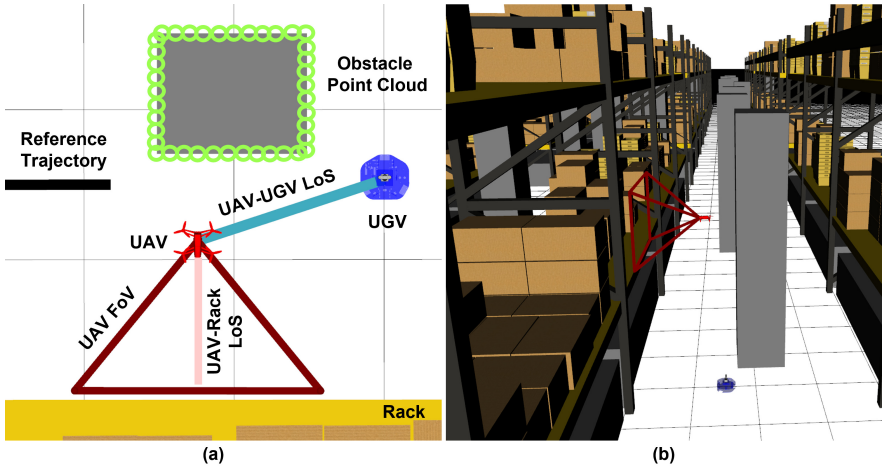


Figure 29. (a) - (b) show top-down and third-person views of a typical warehouse inventory setting.

The vector $\mathbf{p}(.) = [x(.), y_{(.)}]^T, (.) \in r, g$ denotes the 2D position of either the UAV or UGV vehicles. The cost function (7.1a) consists of four terms, with the first two minimizing the sum of the acceleration magnitude for the UAV and UGV, and the last two minimizing the occlusion between the UAV and UGV, and the UAV and the rack, respectively. The occlusion model $f_{occ}(.)$ is the same learned model developed in Publication II and explained in Section 6.2.3. $w_1 - w_3$ are user-defined weights that control the importance of each cost term. Constraints (7.1b) - (7.1c) specify the initial and final boundary conditions for the UAV and UGV trajectories. The inequality constraints (7.1d) and (7.1e) enforce velocity and acceleration limits for the UAV and UGV, respectively, with the exact definitions provided in (7.2). Constraints (7.1f) ensure that the UAV and UGV do not collide with each other. The definition of $f_{o,i}(.)$ is brought in (7.3), where $x_{(.)}, y_{(.)}$ are the positions of UAV/UGV and $x_{o,i}, y_{o,i}$ are the position of the i^{th} obstacle point obtained from the LiDAR. In equation (7.3), the constant a represents the footprint radius of the UAV/UGV. To ensure that the UAV and UGV remain within certain bounds ($\underline{s}_{los}$, $\overline{s}_{los}$) of each other, inequality (7.1g) is utilized. Similarly, inequality (7.1h) ensures that the UAV stays within certain bounds ($\overline{s}_{ref}$, $\underline{s}_{ref}$) of the reference trajectory, which is depicted in black in Figure 29(a). The reference trajectory provides the desired path and velocity that the UAV must follow to scan the warehouse racks, and the tracking bounds on the reference trajectory encode the minimum and maximum distance from which the UAV should view the racks. By projecting the reference trajectory onto the surface of the racks, the scanning trajectory ($x_{sc}(t)$, $y_{sc}(t)$) is obtained, which is used to calculate occlusion between the UAV and the rack at any given time t.

7.3.1. Trajectory Parametrization

Using the parametrization presented in Section 3.3.2, the following compact representations can be derived for (7.1a) - (7.1f).

$$\min_{\boldsymbol{\xi}_r, \boldsymbol{\xi}_g} \frac{1}{2}\boldsymbol{\xi}_r^T w_1 \mathbf{Q} \boldsymbol{\xi}_r + w_2 \left\| \mathbf{f}_{occ}(\boldsymbol{\xi}_r, \boldsymbol{\xi}_g) \right\|_1 + w_3 \left\| \mathbf{f}_{occ}(\boldsymbol{\xi}_r, \mathbf{x}_{sc}, \mathbf{y}_{sc}) \right\|_1 \tag{7.4}$$

$$\begin{aligned} \mathbf{A}_{eq}\boldsymbol{\xi}_r = \mathbf{b}_{eq,r}, \qquad \mathbf{A}_{eq}\boldsymbol{\xi}_g = \mathbf{b}_{eq,g}, \\ \mathbf{h}_1(\boldsymbol{\xi}_r) \leq \mathbf{0}, \qquad \mathbf{h}_1(\boldsymbol{\xi}_g) \leq \mathbf{0}, \qquad \mathbf{h}_2(\boldsymbol{\xi}_r, \boldsymbol{\xi}_g) \leq \mathbf{0}, \end{aligned} \tag{7.5}$$

$$\mathbf{A}_{eq} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}, \mathbf{A} = \left[\mathbf{P}_0 | \dot{\mathbf{P}}_0 | \ddot{\mathbf{P}}_0 | \dot{\mathbf{P}}_m | \ddot{\mathbf{P}}_m | \right]^T, \mathbf{b}_{eq} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_f \end{bmatrix} \tag{7.6}$$

Constraints (7.5) are the matrix representations of (7.1b) - (7.1h). The algebraic form of $\mathbf{h}_1$ and $\mathbf{h}_2$ can be easily obtained by substituting (3.5) - (3.6) into (7.1f) - (7.1h). The vector $\mathbf{f}_{occ}$ is constructed by stacking the f_{occ} at different time instants.

7.3.2. Proposed Optimizer

*a) **Sequential Minimization with Warm-Start**.*

Solving (7.4) - (7.5) jointly for $(\boldsymbol{\xi}_r, \boldsymbol{\xi}_g)$ can be computationally prohibitive. Therefore, a sequential minimization approach is adopted where $\boldsymbol{\xi}_r, \boldsymbol{\xi}_g$ are optimized one at a time. This is summarized in Alg. 1, wherein superscript l tracks the values of variables throughout planning steps. For instance, ${}^{l}\boldsymbol{\xi}_g$ denotes the value of the variable obtained at planning step l. While optimizing for $\boldsymbol{\xi}_g$, only the occlusion cost of UAV-UGV LoS is considered since there is no requirement for the UGV to maintain the visibility of the warehouse racks.

As presented in lines 3-4 of Alg. 1, $\boldsymbol{\xi}_g$ is fixed at the values obtained in the previous step while optimizing for $\boldsymbol{\xi}_r$. The resulting $\boldsymbol{\xi}_r$ is then fixed in the cost and constraint functions to optimize over $\boldsymbol{\xi}_g$ in lines 5. At each optimization step, only constraints that explicitly depend on the variables currently being optimized are considered.

Algorithm 1: Sequential Minimization with Warm-Start

1 Input ${}^{l}\boldsymbol{\xi}_g$, at l^{th} planning step.

2 ${}^{l+1}\boldsymbol{\xi}_r = \arg\min \frac{1}{2}\boldsymbol{\xi}_r^T w_1 \mathbf{Q} \boldsymbol{\xi}_r + w_2 \mathbf{f}_{oc}(\boldsymbol{\xi}_r, {}^{l}\boldsymbol{\xi}_g) + w_3 \mathbf{f}_{oc}(\boldsymbol{\xi}_r, \mathbf{x}_{sc}, \mathbf{y}_{sc})$

3 • $\mathbf{A}_{eq}\boldsymbol{\xi}_r = \mathbf{b}_{eq,r}, \qquad \mathbf{h}_1(\boldsymbol{\xi}_r) \leq \mathbf{0}, \qquad \mathbf{h}_2(\boldsymbol{\xi}_r, {}^{l}\boldsymbol{\xi}_g) \leq \mathbf{0}$

4 ${}^{l+1}\boldsymbol{\xi}_g = \arg\min \frac{1}{2}\boldsymbol{\xi}_g^T w_1 \mathbf{Q} \boldsymbol{\xi}_g + w_2 \mathbf{f}_{oc}(\boldsymbol{\xi}_g, {}^{l+1}\boldsymbol{\xi}_r)$

5 • $\mathbf{A}_{eq}\boldsymbol{\xi}_g = \mathbf{b}_{eq,g}, \qquad \mathbf{h}_1(\boldsymbol{\xi}_g) \leq \mathbf{0}, \qquad \mathbf{h}_2(\boldsymbol{\xi}_g, {}^{l+1}\boldsymbol{\xi}_r) \leq \mathbf{0}$

b) ***CEM-based Optimization***. A CEM-based approach is utilized to solve the optimizations over $\boldsymbol{\xi}_r$ and $\boldsymbol{\xi}_g$ in lines 2-3 and 4-5 of Alg. 1. The process for $\boldsymbol{\xi}_r$ is outlined in Alg. 2, where the left superscript i is employed to monitor the variables across the iterations. At iteration i, n samples of $\boldsymbol{\xi}_r$ are drawn from a Gaussian distribution in line 4, denoted as $\boldsymbol{\xi}_{r,j}$. In line 5, an empty list is initialized to store the cost function corresponding to the drawn samples. The core distinction between the proposed CEM and the baseline variants is represented in lines 7-8. Herein, $\boldsymbol{\xi}_{r,j}$ is projected to $\overline{\boldsymbol{\xi}}_{r,j}$ to satisfy the boundary conditions and other inequality constraints. The cost is evaluated for each of the projected samples in lines 9-10, where $w_1 - w_3$ are user-defined weights to balance the relative importance of the cost terms. The *EliteSet* refers to the q samples of $\overline{\boldsymbol{\xi}}_{r,j}$ with the lowest cost, which are chosen in line 11. In line 12, a Gaussian distribution is fitted to this set, and the resulting mean and covariance are employed for sampling in the next iteration.

The same procedures apply to the UGV with one exception: the UGV's cost will not include the third term in line 9 of Alg. 2, as there is no occlusion requirement between the UGV and the warehouse racks.

Algorithm 2: CEM-Based Optimization

1 N_{CEM} = Number of CEM iterations
2 Initiate mean ${}^{i}\boldsymbol{\mu}, {}^{i}\boldsymbol{\Sigma}$, at $i = 0$
3 **for** $i = 1, i \leq N_{CEM}, i++$ **do**
4 Draw n Samples $(\boldsymbol{\xi}_{r,1}, \boldsymbol{\xi}_{r,2}, \boldsymbol{\xi}_{r,j}, .., \boldsymbol{\xi}_{r,n})$ from $\mathcal{N}({}^{i}\boldsymbol{\mu}, {}^{i}\boldsymbol{\Sigma})$
5 Initialize $CostList$ = []
6 Project $\boldsymbol{\xi}_{r,j}$ to $\overline{\boldsymbol{\xi}}_{r,j}$ through the following optimization $\forall j$:
7 • $\overline{\boldsymbol{\xi}}_{r,j} = argmin_{\overline{\boldsymbol{\xi}}_{r,j}} \left\| \overline{\boldsymbol{\xi}}_{r,j} - \boldsymbol{\xi}_{r,j} \right\|_2^2$
8 • $\boldsymbol{A}_{eq}\overline{\boldsymbol{\xi}}_{r,j} = \boldsymbol{b}_{eq,r}, \quad \mathbf{h}_1(\overline{\boldsymbol{\xi}}_{r,j}) \leq 0, \quad \mathbf{h}_2(\overline{\boldsymbol{\xi}}_{r,j}, {}^{l}\boldsymbol{\xi}_g) \leq 0$
9 $cost \leftarrow \frac{1}{2}\overline{\boldsymbol{\xi}}_{r,j}^{T} w_1 \mathbf{Q} \overline{\boldsymbol{\xi}}_{r,j} + w_2 \|\mathbf{f}_{oc}(\overline{\boldsymbol{\xi}}_{r,j}, \boldsymbol{\xi}_g)\|_1 + w_3 \|\mathbf{f}_{oc}(\overline{\boldsymbol{\xi}}_{r,j}, \boldsymbol{x}_{sc}, \boldsymbol{y}_{sc})\|_1$, over n samples
10 append $cost$ to $CostList$
11 $EliteSet \leftarrow$ Select top q lowest cost samples
12 $({}^{i+1}\boldsymbol{\mu}, {}^{i+1}\boldsymbol{\Sigma}) \leftarrow$ Fit Gaussian distribution $EliteSet$
13 **end**
14 **return** *Sample corresponding to the lowest* $\overline{\boldsymbol{\xi}}_{r,j}$ *in the EliteSet*

c) ***Efficient Batch Projection***. Alg. 2 includes projection optimizations in lines 7-8, which can be performed in parallel for each sample $\boldsymbol{\xi}_{r,j}$ to enhance computational efficiency. To achieve this, the constraints (7.1d) - (7.1h) are modified to create a suitable structure in the projection optimization. The primary computation in the projection optimization is equivalent

to solving a batch of QPs in form of (7.7).

$$\frac{1}{2}\overline{\boldsymbol{\xi}}_{r,j}^{T}\overline{\mathbf{Q}}\,\overline{\boldsymbol{\xi}}_{r,j}+\overline{\mathbf{q}}_{r,j}^{T}\boldsymbol{\xi}_{r,j},\mathbf{A}_{eq}\overline{\boldsymbol{\xi}}_{r,j}=\mathbf{b}_{eq,r} \tag{7.7}$$

The vector $\overline{\mathbf{q}}_{r,j}$ in the above formulation should be recomputed for each batch. The other matrices $\overline{\mathbf{Q}}, \mathbf{A}_{eq}$ remain constant across all the batches. Thus, the solution to (7.7) can be computed in one shot across all the batches through (7.8).

$$\begin{bmatrix} \boldsymbol{\xi}_{r,j} & \cdot\cdot & \boldsymbol{\xi}_{r,n} \\ \boldsymbol{\nu}_{r,j} & \cdot\cdot & \boldsymbol{\nu}_{r,n} \end{bmatrix} = \overbrace{\left(\begin{bmatrix} \overline{\mathbf{Q}} & \mathbf{A}_{eq}^{T} \\ \mathbf{A}_{eq} & \mathbf{0} \end{bmatrix}^{-1}\right)}^{constant} \begin{bmatrix} \overline{\mathbf{q}}_{r,j} & \cdot\cdot & \overline{\mathbf{q}}_{r,n} \\ \mathbf{b}_{eq,r} & \cdot\cdot & \mathbf{b}_{eq,r} \end{bmatrix} \tag{7.8}$$

wherein $\boldsymbol{\nu}_{r,j}$ is the dual-variable associated with the equality constraints. Irrespective of the batch size, the inverse on the right-hand side of (7.8) is computed only once. Thus the batch solution of (7.7) reduces to just a large matrix-vector product that can be trivially parallelized over GPUs. The solution process of (7.7) is presented next.

Here, a more detailed analysis of the projection optimization in lines 7-8 of Alg. 2 is presented. The derivations in Publication II presented in Section 6.2.2 are extended to include the cooperative navigation setting. Specifically, the optimizer introduced in Publication II is updated with new constraints for collision avoidance and UAV-UGV separation.

To begin, the inequality constraints (7.1d) - (7.1h) are reformulated by noting that in lines 7-8, the optimal UGV trajectory coefficients ${}^{l}\boldsymbol{\xi}_{g}$ are known from the previous planning step. In other words, the waypoints ${}^{l}\mathbf{p}_{g} = ({}^{l}x_{g}(t), {}^{l}y_{g}(t))$ for the UGV are known. For notational simplicity, the superscript l is dropped in the following.

The quadratic collision constraints (7.1f) are reformulated into the following polar form:

$$\mathbf{f}_{o,i}(.) = 0, d_{o,i} \geq 1, \forall i \tag{7.9}$$

$$\mathbf{f}_{o,i} = \left\{ \begin{array}{l} x_r(t) - x_i(t) - a d_{o,i}(t)\cos\alpha_{o,i}(t) \\ y_r(t) - y_i(t) - a d_{o,i}(t)\sin\alpha_{o,i}(t) \end{array} \right\} \tag{7.10}$$

The angle between the vector connecting the center of the UAV and the i^{th} obstacle, and the X axis is represented by $\alpha_{o,i}(t)$. The variable $d_{o,i}(t)$ denotes the normalized magnitude of this vector. It can be easily verified that satisfaction of (7.9) ensures the satisfaction of original constraints (7.1f) based on the Euclidean distance.

Following a similar approach, constraints (7.1d) - (7.1e) and (7.1g) - (7.1h) can be reformulated to the forms (7.11) - (7.14).

$$\mathbf{f}_v(.) = 0, d_v(t) \leq v_r, \tag{7.11}$$
$$\mathbf{f}_a(.) = 0, d_a(t) \leq a_r, \tag{7.12}$$
$$\mathbf{f}_{los}(.) = 0, \underline{s}_{los} \leq d_{los}(t) \leq \overline{s}_{los}, \tag{7.13}$$
$$\mathbf{f}_{ref}(.) = 0, \underline{s}_{ref} \leq d_{ref}(t) \leq \overline{s}_{ref}, \tag{7.14}$$

where

$$\mathbf{f}_v = \left\{ \begin{array}{c} \dot{x}_r(t) - d_v(t)\cos\alpha_v(t) \\ \dot{y}_r(t) - d_v(t)\sin\alpha_v(t) \end{array} \right\}, \tag{7.15}$$

$$\mathbf{f}_a = \left\{ \begin{array}{c} \ddot{x}_r(t) - d_a(t)\cos\alpha_a(t) \\ \ddot{y}_r(t) - d_a(t)\sin\alpha_a(t) \end{array} \right\}, \tag{7.16}$$

$$\mathbf{f}_{los} = \left\{ \begin{array}{c} x_r(t) - x_g(t) - d_{los}(t)\cos\alpha_{los}(t) \\ y_r(t) - y_g(t) - d_{los}(t)\sin\alpha_{los}(t) \end{array} \right\}, \tag{7.17}$$

$$\mathbf{f}_{ref} = \left\{ \begin{array}{c} x_r(t) - x_{ref}(t) - d_{ref}(t)\cos\alpha_{ref}(t) \\ y_r(t) - y_{ref}(t) - d_{ref}(t)\sin\alpha_{ref}(t) \end{array} \right\}. \tag{7.18}$$

Remark 10. *$\alpha_{o,i}$, $\alpha_{los}(t)$, $\alpha_{ref}(t)$, $\alpha_v(t)$, $\alpha_a(t)$, $d_{o,i}$, $d_{los}(t)$, $d_{ref}(t)$, $d_v(t)$, and $d_a(t)$ are extra variables that will be obtained by the batch projection optimizer along with $\overline{\boldsymbol{\xi}}_{r,j}$.*

d) ***Reformulation of Projection.*** Using the formulation developed in the previous section as well as the parametrizations in (3.5) - (3.6), the projection optimization in lines 7-8 of Alg. 2 can be replaced with the following:

$$\min_{\overline{\boldsymbol{\xi}}_{r,j}} \frac{1}{2} \left\| \overline{\boldsymbol{\xi}}_{r,j} - \boldsymbol{\xi}_{r,j} \right\|_2^2 \tag{7.19a}$$
$$\mathbf{A}_{eq}\overline{\boldsymbol{\xi}}_{r,j} = \mathbf{b}_{eq,r} \tag{7.19b}$$
$$\mathbf{F}\overline{\boldsymbol{\xi}}_{r,j} = \mathbf{e}_j(\boldsymbol{\alpha}_j, \mathbf{d}_j), \qquad \underline{\mathbf{d}} \leq \mathbf{d}_j \leq \overline{\mathbf{d}} \tag{7.19c}$$

$$\mathbf{F} = \begin{bmatrix} \begin{bmatrix} \mathbf{F}_o \\ \dot{\mathbf{W}} \\ \ddot{\mathbf{W}} \\ \mathbf{W} \\ \mathbf{W} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} \mathbf{F}_o \\ \dot{\mathbf{W}} \\ \ddot{\mathbf{W}} \\ \mathbf{W} \\ \mathbf{W} \end{bmatrix} \end{bmatrix}, \mathbf{e} = \begin{bmatrix} \mathbf{x}_o + a\mathbf{d}_{o,j}\cos\boldsymbol{\alpha}_{o,j} \\ \mathbf{d}_{v,j}\cos\boldsymbol{\alpha}_{v,j} \\ \mathbf{d}_{a,j}\cos\boldsymbol{\alpha}_{a,j} \\ \mathbf{x}_g + \mathbf{d}_{los,j}\cos\boldsymbol{\alpha}_{los,j} \\ \mathbf{x}_{ref} + \mathbf{d}_{ref,j}\cos\boldsymbol{\alpha}_{ref,j} \\ \mathbf{y}_o + a\mathbf{d}_{o,j}\sin\boldsymbol{\alpha}_{o,j} \\ \mathbf{d}_{v,j}\sin\boldsymbol{\alpha}_{v,j} \\ \mathbf{d}_{a,j}\sin\boldsymbol{\alpha}_{a,j} \\ \mathbf{y}_g + \mathbf{d}_{los,j}\sin\boldsymbol{\alpha}_{los,j} \\ \mathbf{y}_{ref} + \mathbf{d}_{ref,j}\sin\boldsymbol{\alpha}_{ref,j} \end{bmatrix} \tag{7.20}$$

$$\boldsymbol{\alpha}_j = (\boldsymbol{\alpha}_{o,j}, \boldsymbol{\alpha}_{los,j}, \boldsymbol{\alpha}_{ref,j}, \boldsymbol{\alpha}_{v,j}, \boldsymbol{\alpha}_{a,j}) \tag{7.21}$$

$$\mathbf{d}_j = (\mathbf{d}_{o,j}, \mathbf{d}_{los,j}, \mathbf{d}_{ref,j}, \mathbf{d}_{v,j}, \mathbf{d}_{a,j}) \tag{7.22}$$

The matrix $\mathbf{F}_o$ is obtained by stacking the matrix $\mathbf{W}$ from (3.5) - (3.6) as many times as the number of obstacles in the point cloud, for collision avoidance at a given planning cycle. The vectors $\mathbf{x}_o, \mathbf{y}_o$ are formed by appropriately stacking $x_{o,i}(t), y_{o,i}(t)$ at different time instants. Similar construction is followed to obtain $\boldsymbol{\alpha}_o$, $\boldsymbol{\alpha}_{los}$, $\boldsymbol{\alpha}_{ref}$, $\boldsymbol{\alpha}_v$, $\boldsymbol{\alpha}_a$, $\mathbf{d}_o$, $\mathbf{d}_{los}$, $\mathbf{d}_{ref}$, $\boldsymbol{d}_v$, $\boldsymbol{d}_a$. The vectors $\underline{\mathbf{d}}$, $\overline{\mathbf{d}}$ are constructed by stacking the lower and upper bounds of individual d's from (7.9) - (7.18).
The subscript j signifies that (7.19a)-(7.19c) are defined for projection of the j^{th} sample of $\boldsymbol{\xi}_{r,j}$. In total, there would be n such optimization problems. Second, the constraints (7.19c) serve as the replacement for $\mathbf{h}_1(\overline{\boldsymbol{\xi}}_{r,j}) \leq 0$ and $\mathbf{h}_2(\overline{\boldsymbol{\xi}}_{r,j}, {}^{l}\boldsymbol{\xi}_g) \leq 0$ in line 8 of the Alg. 2

e) ***Solution Process****.* The non-convex equality constraints (7.19c) are relaxed as l_2 penalties and augmented into the projection cost (7.19a).

$$\mathcal{L}(\boldsymbol{\xi}) = \frac{1}{2}\left\|\overline{\boldsymbol{\xi}}_{r,j} - \boldsymbol{\xi}_{r,j}\right\|_2^2 - \langle \boldsymbol{\lambda}_j, \overline{\boldsymbol{\xi}}_{r,j} \rangle + \frac{\rho}{2}\left\|\mathbf{F}\overline{\boldsymbol{\xi}}_{r,j} - \mathbf{e}(\boldsymbol{\alpha}_j, \mathbf{d}_j)\right\|_2^2 \tag{7.23}$$

The use of so-called Lagrange multipliers $\boldsymbol{\lambda}_j$ plays a crucial role in driving the residuals to zero for any arbitrary ρ [84].

AM is used to minimize (7.23) subject to (7.19b). This reduces to the following recursive steps, wherein the left superscript k is used to track the values across iterations.

$${}^{k+1}\overline{\boldsymbol{\xi}}_r = \arg\min_{\boldsymbol{\xi}_{r,j}} \mathcal{L}(\overline{\boldsymbol{\xi}}_r, {}^{k}\boldsymbol{\alpha}_j, {}^{k}\mathbf{d}_j), \text{s.t } \mathbf{A}_{eq}\overline{\boldsymbol{\xi}}_{r,j} = \mathbf{b}_{eq} \tag{7.24a}$$

$${}^{k+1}\boldsymbol{\alpha}_j = \arg\min_{\boldsymbol{\alpha}_j} \mathcal{L}({}^{k+1}\overline{\boldsymbol{\xi}}_r, \boldsymbol{\alpha}_j, {}^{k}\mathbf{d}_j) \tag{7.24b}$$

$${}^{k+1}\mathbf{d}_j = \arg\min_{\mathbf{d}_j} \mathcal{L}({}^{k+1}\overline{\boldsymbol{\xi}}_r, {}^{k+1}\boldsymbol{\alpha}_j, \mathbf{d}_j) \tag{7.24c}$$

More specifically, only one variable amongst $\overline{\boldsymbol{\xi}}_{r,j}$, $\boldsymbol{\alpha}_j$, and $\mathbf{d}_j$ is optimized, while the other two are held fixed at the previous update.

The minimization (7.24a) is a convex QP with the same structure as (7.7), with $\overline{\mathbf{Q}} = \mathbf{I} + \rho\mathbf{F}^T\mathbf{F}$, $\overline{\mathbf{q}}_{r,j} = -\rho\mathbf{F}^T\mathbf{e} - \boldsymbol{\lambda}_j$. Therefore, the solution across the entire batch can be calculated in parallel using (7.8). The steps (7.24b) and (7.24c) have a closed-form solution that can be trivially batched using the approach presented in Section 6.2.2 in the previous chapter. The projection operator can be trivially extended for the UGV case. It will be used in line 4 of Alg. 1 for optimizing UGV trajectory with the CEM of Alg. 2.

7.3.3. CVAE Initialization

With naive Gaussian initialization, the CEM in Alg. 2 may require a large batch size and many iterations to converge to an optimal solution. Thus, in this section, a data-driven approach is adopted to derive a more informed initialization for the CEM. More specifically, a Behavior Cloning (BC) framework is derived to learn a policy $\boldsymbol{\pi_\theta}$ that maps state observations $\mathbf{o}$ and laser (LiDAR) point cloud $\mathbf{c}$ to a distribution over optimal trajectory coefficients for UAV/UGV, $\boldsymbol{\xi}^* = (\boldsymbol{\xi}_r^*, \boldsymbol{\xi}_g^*)$. The observations $\mathbf{o}$ have information about the positions and velocities of the UAV/UGV.

$\pi_{\boldsymbol{\theta}}$ is modeled as a probabilistic generative model in the form of a CVAE [95], [96]. To train, a dataset with different samples of $(\mathbf{o}, \mathbf{c})$ and their corresponding expert demonstration trajectories $\boldsymbol{\tau}_e$ is collected.

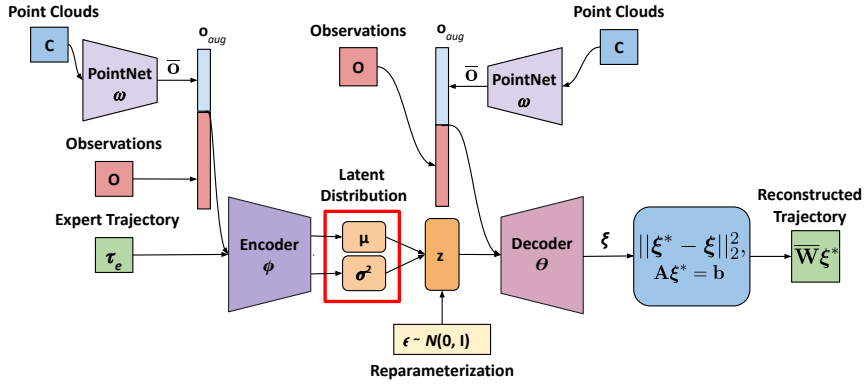


Figure 30. The pipeline for the learned CVAE.

The proposed CVAE model, illustrated in Figure 30, consists of an encoder-decoder architecture augmented with PointNet [97] to extract features $\overline{\mathbf{o}}$ from unstructured and unordered LiDAR point clouds $\mathbf{c}$. The learnable parameters of PointNet are represented as $\boldsymbol{\omega}$. The features from PointNet and state observations are stacked to create an augmented vector $\mathbf{o}_{aug} = (\mathbf{o}, \overline{\mathbf{o}})$, which is then fed to an encoder MLP along with the expert trajectory $\boldsymbol{\tau}_e$. The encoder compresses this information to a latent variable $\mathbf{z}$ with distribution $\mathbf{q}_\phi(\mathbf{z}|\mathbf{o}_{aug}, \boldsymbol{\tau}_e) = \mathcal{N}(\boldsymbol{\mu}_\phi, diag(\boldsymbol{\sigma}^2{}_\phi))$

The decoder model consists of an MLP with parameters $\boldsymbol{\theta}$ and a differentiable optimization layer. It takes in $\mathbf{o}_{aug}$ and maps $\mathbf{z} \sim \mathbf{q}_\phi(\mathbf{z}|\mathbf{o}_{aug}, \boldsymbol{\tau}_e)$ to the output distribution $\boldsymbol{\pi}_\theta(\boldsymbol{\xi}^*|\mathbf{z}, \mathbf{o}_{aug})$ over the optimal trajectory coefficients. The role of the optimization layer is to project the UAV/UGV coefficient distribution resulting from the MLP to the vehicles' current positions and velocities. Thus, the optimization layer is simply a differentiable equality-constrained QP. Both the encoder and decoder along with PointNet are trained in an end-to-end fashion using the loss function presented in (7.25), where $\overline{\mathbf{W}} = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix}$ (recall (3.5) - (3.6)). The first term is

called the reconstruction loss and aims to make the trajectory predicted by the CVAE as close as possible to the expert trajectory. The second term aims to make the latent distribution very similar to an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix.

$$\min \sum \left\| \overline{\mathbf{W}} \boldsymbol{\xi}^* - \boldsymbol{\tau}_e \right\|_2^2 + \beta \, D_{\mathbf{KL}}[\mathbf{q}_\phi(\mathbf{z} \,|\, \mathbf{o}_{aug}, \, \boldsymbol{\tau}_e) \,|\, \mathcal{N}(\mathbf{0}, \mathbf{I})] \tag{7.25}$$

The back-propagation, required for updating the parameters $(\boldsymbol{\omega}, \boldsymbol{\phi}, \boldsymbol{\theta})$, traces the gradient with respect to the loss function through the QP layer. In the inference phase, the samples are drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (instead of $\mathbf{q}_\phi(\mathbf{z}|\mathbf{o}_{aug}, \boldsymbol{\tau}_e)$) and then passed through the decoder network to generate a distribution over optimal trajectory coefficients $\boldsymbol{\xi}^*$, conditioned on state observations $\mathbf{o}$ and point clouds $\mathbf{c}$. The β hyper-parameter acts as a trade-off between the two cost terms. The KL-vanishing issue is tackled by applying a monotonic annealing schedule of β hyper-parameter [98], starting with 0 and gradually annealing the β at each optimizer step.

7.4. Validation and Benchmarking

This section validates the proposed approach over different ablations and presents the comparisons with the baselines.

7.4.1. Implementation Details

Alg. 1 and 2 are implemented in Python using JAX [92] library as the GPU-accelerated linear algebra backend. The polynomial matrix $\mathbf{W}$ is constructed from a 10^{th} order polynomial. The simulation pipeline is built on top of ROS [91] and the physics simulator Gazebo. The benchmarks are run on AORUS Laptop Intel core $i7-11800H$ with NVIDIA RTX 3080 and 32 GB RAM. For the real-world experiments, Parrot Bebop 2 and Robotont UGV [99] are used. For occlusion handling, the learned occlusion model in Publication II (Section 6.2.3) is used that takes in a robot, a target, and obstacle positions as inputs and outputs an occlusion value. Thus, this model is evaluated over the UAV-UGV pair and between the UAV and the moving point of interest on the specified warehouse rack. The expert demonstrations to learn the CVAE are collected through teleoperation. The parameters used in the CVAE and its architecture are brought in Tables 5, 6, 7.

An MPC pipeline is developed on top of the proposed optimizer. It takes the UGV and UAV's current positions and velocities as feedback along with LiDAR (laser) point cloud and generates occlusion-free trajectories for both vehicles in real-time. No warm-starting is used for the proposed CEM optimizer in Alg. 2.

Table 5. PointNet Architecture

PointNet Network ω					
Block	***Layers***	***Output Size***	***Kernel Size***	***Bias***	***Activation***
Conv 1-3	1D Convolution, Batchnorm	64	1	False	ReLU
Conv 4	1D Convolution, Batchnorm	128	1	False	ReLU
Conv 5	1D Convolution, Batchnorm	1024	1	False	ReLU
MLP 1	Adaptive Maxpool, Linear, Batchnorm, Dropout	256	None	False	ReLU
$\overline{\mathbf{o}}$	Linear	40	None	True	None

Table 6. CVAE Encoder Architecture

Encoder Network ϕ			
Block	***Layers***	***Output Size***	***Activation***
MLP 1-4	Linear, Batchnorm	1024	ReLU
MLP 5	Linear, Batchnorm	256	ReLU
Mean	Linear	2 / 6	None
Variance	Linear	2 / 6	Softplus

Table 7. CVAE Decoder Architecture

Decoder Network θ			
Block	***Layers***	***Output Size***	***Activation***
MLP 1-4	Linear, Batchnorm	1024	ReLU
MLP 5	Linear, Batchnorm	256	ReLU
$\boldsymbol{\xi}$	Linear	44	None
$\boldsymbol{\xi}^*$	Optimization Layer	44	None

a) Baselines. The proposed optimizer is compared with the following baselines:

Conventional CEM: This is the most popular optimizer for planning with learned neural-network models such as the learned occlusion cost presented in this work (e.g. see [100]). Recall that Alg. 2 improves CEM by augmenting a projection optimization and thus, this comparison essentially presents the benefit of the said augmentation.

CMA-ES: This sampling-based optimizer differs from CEM in the way it adapts the covariance matrix of the sampling distribution. In this work, the customization of CMA-ES proposed in [101] is used that is re-implemented with the JAX GPU accelerated library [102].

b) Hyper-parameter Selection. The batch size n of CEM in Alg. 2 is 100 and the top 20 of them are chosen as the *EliteSet* in each iteration. The smoothness and occlusion weights are scaled to obtain a trade-off between fluent motion and occlusion avoidance. It is worth mentioning that this scaling is kept the same throughout all the experiments.

c) Metrics. The following metrics are utilized for benchmarking:

UAV-UGV Occlusion Time: The total time during which the UAV and UGV's line of sight is occluded by the obstacles.

UAV-Rack Occlusion Time: The total time during which UAV cannot scan the racks due to the occlusions stemming from the obstacles.

Acceleration: This metric measures how fast the UAV and UGV change their speed during experiments.

Computation Time: The time taken to compute the optimal trajectories for the UAV and UGV.

Normalized Distance Traversed: The distance traveled before the occurrence of any collisions divided by the total distance of that particular experimental run.

7.4.2. Gaussian and CVAE Initialization

This subsection presents a qualitative demonstration of Alg. 1 and Alg. 2 for both Gaussian and learned CVAE initializations of the latter. Figure 31 presents the top-down view of a typical warehouse scenario where the UAV needs to scan the racks while following the reference trajectory (black) and avoiding the obstacles (gray). Thus, both UAV and UGV need to avoid the obstacle from the bottom so that the UAV can keep seeing the rack and to ensure that the UAV-UGV LoS is not occluded. When Alg. 2 is initialized with a Gaussian distribution, we obtain a large number of occlusion/collision-free samples from the second iteration. From the fifth iteration onwards, all the samples converge and become occlusion/collision-free. However, when Alg. 2 is initialized using the learned CVAE model, almost all the trajectories are already collision/occlusion free in the first iteration itself. And starting from the second iteration, all the samples

converge to the optimal solution. In general, it is observed that CVAE-based initialization significantly accelerates the convergence of Alg. 2. Moreover, it also allows us to use a smaller batch size. This is discussed in more detail in the next subsection.

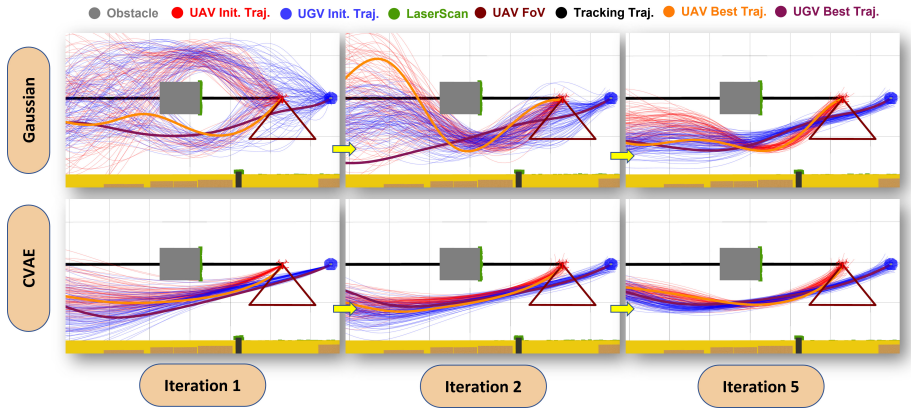


Figure 31. The proposed CEM optimizer Alg. 2 with naive Gaussian and learned CVAE initialization. The latter produces distributions more focused around regions that are collision-free and maintain UAV-UGV and UAV-rack visibility.

7.4.3. Effect of CVAE

Here, we quantify the effect of initializing the proposed CEM-based optimizer with a learned CVAE distribution vis-a-vis Gaussian. The results of different metrics are shown in Figure 32. The CVAE initialization can match the performance of the Gaussian variant with $1/5^{th}$ of the batch (sample) size and fewer iterations. This, in turn, allows the CVAE variant to run on embedded devices such as NVIDIA Jetson Xavier with a computation time of 0.09s. It is important to point out two additional experimental details. First, during the experiments with Jetson Xavier, the entire Gazebo simulation pipeline and the proposed optimizer ran on the embedded board. Second, only 50% of the available GPU memory was allocated to the proposed optimizer using JAX's internal parameters. Both these experimental settings give a reliable indication that the proposed Alg. 2 for both UAV and UGV can run in real-time alongside another software stack for perception, state estimation, etc.

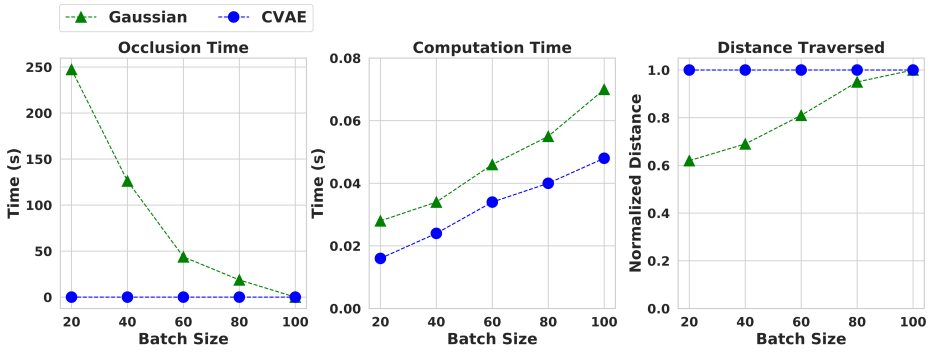


Figure 32. The comparison of different metrics of the proposed optimizer when it is warm-started with Gaussian sampling and CVAE. Occlusion time refers to combined UAV-UGV and UAV-rack occlusion time.

7.4.4. Comparison with SOTA Sampling Based Optimizers

In this subsection, the proposed Alg. 2 with Gaussian and CVAE initializations is compared with CEM and CMA-ES baselines.

The results are summarized in Table 8. The UAV-rack and UAV-UGV occlusion times achieved for the baseline CEM with 100 batch size were 15.4 and 290.5 seconds, respectively. These figures decreased considerably to 10.2 and 25.7 seconds when the batch size increased to 400. CMA-ES, expectedly performed better than CEM on the occlusion time metric. In fact CMA-ES with a batch size of 250 outperformed CEM operating with a batch size of 400. The projection-augmented CEM of Alg. 2 with Gaussian initialization and the batch size of 100 outperformed both CEM and CMA-ES on the occlusion time metric. When warm-started with the learned CVAE initialization, Alg. 2 could achieve the same performance with a batch size of only 20.

Table 8. Overall Distance Traveled / Sim. Time: 1045.2m / 1628s

Ablations	UAV-UGV/ UAV-Rack Occ. Time(s)	Normalized Distance Mean / STD	Linear Acc. Mean / Max	Comp. Time (s)
CEM Batch 100	15.4 / 290.5	0.62 / 0.15	0.17 / 0.49	0.15
CEM Batch 400	10.2 / 25.7	0.94 / 0.1	0.14 / 0.52	0.36
CMA-ES Batch 100	18.4/ 153.7	0.66 / 0.14	0.19 / 0.42	0.14
CMA-ES Batch 250	7.8 / 12.4	0.95 / 0.12	0.12 / 0.68	0.19
Alg. 2 - Gauss. Batch 100	**0 / 0**	**1 / 0**	**0.32 / 0.83**	**0.07**
Alg. 2 - CVAE Batch 20	**0 / 0**	**1 / 0**	**0.34 / 0.52**	**0.016**

The normalized distance metric exhibits the same trends for all the baselines and the proposed approach. Both CEM and CMA-ES require a large batch size to reach the mean normalized distance of around 0.95. In contrast, the proposed Alg. 2 with both Gaussian and CVAE initializations achieves the perfect value of 1 in the runs across all the benchmark environments.

The proposed optimizer, Alg. 2, however, used more acceleration effort than both CEM and CMA-ES to adapt the motions quickly to unforeseen obstacles. Both CEM and CMA-ES required more iterations to obtain a feasible solution. This coupled with a larger batch size make their computation times two to five times that of the proposed Alg. 2.

7.4.5. Comparison with A^* planners

Here, the proposed approach is compared with A^*-based planning. More specifically, the sequential optimization framework of Alg. 1 is retained, but the projection augmented CEM of Alg. 2 is replaced with an A^*-based approach. The endpoint of the reference trajectory is given as the goal point for both UAV and UGV. The qualitative results are summarized in Figure 33. As shown, the A^* planner can find the correct occlusion-free homotopy for collision avoidance. However, the main difference from the proposed optimizer lies in the computation time. The improvement provided by the proposed approach is particularly stark while planning over longer horizons, which is indeed critical while navigating in unknown environments. Furthermore, the A^*-computed trajectories need additional smoothing. In contrast, the proposed approach directly produces the smooth velocity commands that can be executed on the UAV and UGV.

The difference in the computation time can be explained in the following manner. A^* performs an incremental search over a discretized grid representation of the workspace and thus requires several sequential queries of occlusion and tracking costs. The latter is particularly computationally expensive. In contrast, the proposed optimizer performs a more focused search along smooth trajectories. Moreover, cost computation along all the sampled trajectories is parallelized over GPUs.

Table 9. Computation Time Comparison of A* Planners

Method	Horizon 2m	Horizon 4m	Horizon 6m	Horizon 8m	Horizon 10m
A*	0.002s	0.004s	0.006s	0.018s	0.028s
A* with Occ. Cost	0.24s	0.5s	0.8s	0.1.42s	1.86s
Alg. 2 (Ours)	0.028s	0.036s	0.048s	0.056s	0.07s

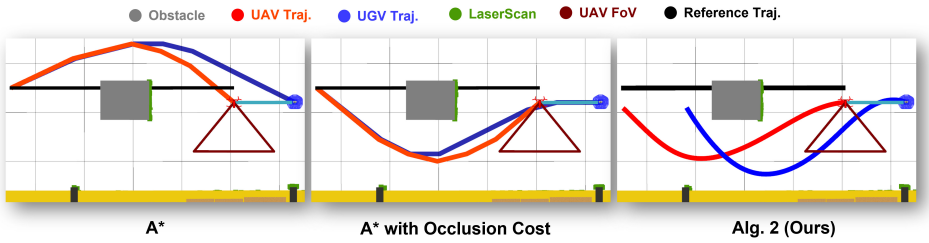


Figure 33. The comparison of trajectories generated for UAV and UGV using A* and the optimizer presented in Alg. 2

7.4.6. Ablation Studies

a) ***Importance of Occlusion Cost with Respect to Racks.*** For scanning warehouse racks, a UAV is often expected to follow a reference trajectory which may not be collision-free in unknown environments. Thus, the UAV has to avoid obstacles while performing the scanning task. Figure 34(a) shows an instance where the UAV can avoid the obstacle from the top or the bottom. Although the UAV and UGV could easily avoid the obstacle from the top, while doing so, the UAV's view of the rack will be occluded and the scanning task would have to be repeated. To alleviate this, in addition to the UAV-UGV occlusion cost, a UAV-rack occlusion cost ($f_{oc}(\mathbf{p}_r(t), \mathbf{p}_{sc}(t))$ from (7.1a)) must be added to the optimizer.

Figure 35(b) and Figure 35(c) show the results in the absence and presence of UAV-rack occlusion cost, respectively. As it can be seen, in the absence of UAV-rack occlusion cost, the obstacle is avoided from the top, and thus the UAV loses visibility of the rack. However, when the UAV-rack occlusion cost is added, the UAV overtakes the obstacle from the bottom and keeps seeing the rack.

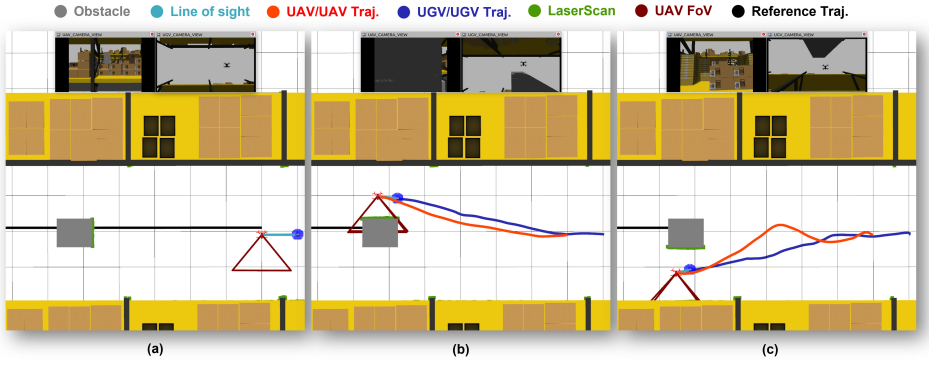


Figure 34. An experiment demonstrating the importance of having an occlusion cost between the UAV and the rack. Figure (b) shows the result when the UAV's FoV is occluded due to absence of the occlusion cost. Figure (c) shows how adding the occlusion cost makes the vehicles overtake the obstacle from the bottom to maintain the rack's visibility (recall last term of (7.1a)).

Table 10. Overall Distance Traveled: 1508.4m / Sim. Time: 1890s

Ablations	UAV-UGV/ UAV-Rack Occlusion Time(s)	Normalized Distance Mean / STD	Linear Acceleration Mean / Max	Computation Time (s)
With rack Occlusion	0 / 0	1 / 0	0.32 / 0.83	0.07
Without rack Occlusion	0 / 140.4	1 / 0	0.15 / 0.36	0.052

The simulation benchmarks are run for both variants, and the results regarding different metrics are provided in Table 11. The simulation benchmarks spanned approximately 1890 seconds, covering a total distance of 1508.4 meters with 10 different obstacle configurations. As shown, having no rack occlusion cost results in 140.4 seconds where obstacles occlude the UAV's view of the racks. In contrast, when UAV-rack occlusion cost is introduced to the optimizer, the vehicles finish the scanning task perfectly. In both cases, the vehicles finish the runs with zero collisions. Naturally, the UAV-UGV pair used more aggressive accelerations in the presence of UAV-rack occlusion cost. The optimizer with UAV-rack occlusion cost is slightly more computationally expensive but real-time nonetheless.

*b) **Importance of Incorporating Occlusion Costs for Both UAV and UGV**.* In a cooperative navigation setting, authors in [66] recommend putting the complete responsibility of keeping the inter-vehicle LoS occlusion-free to either of the vehicles. In this subsection, we test this hypothesis for our problem setting. We ran all the ablations across the same benchmark scenarios as in Section a).

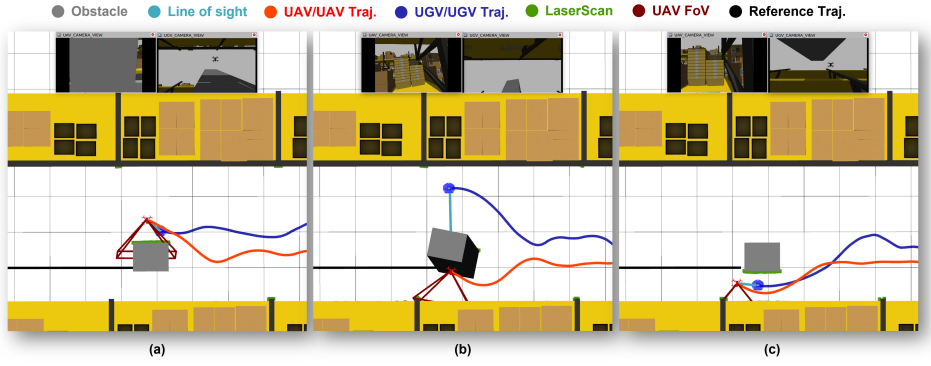


Figure 35. (a) - (c) present the results of three ablations, where the responsibility of keeping the UAV-UGV LoS occlusion-free is handled by UAV, UGV or both. In Figure (a), the occlusion is handled by the UGV. Figure (b) depicts the results when the responsibility of avoiding occlusion is on the UAV. And finally, Figure (c) shows the result when occlusion is cooperatively handled by both.

Occlusion handled by UAV: Here, the UAV takes the occlusion-avoidance responsibility. The UGV does not help the UAV keep their LoS occlusion free. Mathematically, it means the UGV, while solving its op-

Table 11. Overall Distance Traveled: 1508.4m / Sim. Time: 890s

Ablations	UAV-UGV/ UAV-Rack Occlusion Time(s)	Normalized Distance Mean / STD	Linear Acceleration Mean / Max	Computation Time (s)
Occlusion handled by UAV	6.2 / 120.6	0.73 / 0.31	0.27 / 0.64	0.034
Occlusion handled by UGV	8.2 / 15.3	0.76 / 0.32	0.24 / 0.63	0.034
Occlusion handled by UAV-UGV	0 / 0	1 / 0	0.32 / 0.83	0.07

timization problem, does not consider the cost term $f_{oc}(\mathbf{p}_r(t), \mathbf{p}_g(t))$ from (7.1a). Figure 35(a) shows a scenario where the UAV would like to avoid the obstacle from the bottom to keep the racks visible. However, no such requirement exists for the UGV, for whom avoiding the obstacle from the top is easier in the current configuration. Thus, the UAV ends up sacrificing its primary task to keep the UAV-UGV LoS occlusion-free. However, this behavior is not fixed and depends on many factors. Based on the relative positions, the UAV may choose to maintain the visibility of the racks and not the UGV. Table 11 (first row) presents the quantitative metrics for this ablation.

Occlusion handled by UGV: In this scenario, the UGV is responsible for keeping UAV-UGV LoS free of occlusion. Thus, the UAV does not consider $f_{oc}(\mathbf{p}_r(t), \mathbf{p}_g(t))$ (recall (7.1a)) during its optimization. Figure 35(b) shows that the UAV takes the path from the bottom to keep scanning the rack. However, due to conflicting costs, the UGV decides to take the path to the top. Therefore, the UAV and UGV lose visibility of each other because of the obstacle in between. As a result, the UGV can no longer guide the UAV, resulting in the latter colliding with the obstacle. The second row in Table 11 shows the quantitative metrics of these ablations.

Occlusion handled by both UAV and UGV: Here, the responsibility of occlusion avoidance is taken by both vehicles. In other words, both would help each other to prevent obstacles from coming in the way of their LoS. Figure 35(c) shows the same configuration in which the previous two variants failed. As can be seen, having $f_{oc}(\mathbf{p}_r(t), \mathbf{p}_g(t))$ during both vehicles' optimization made the UGV follow the UAV and avoid the obstacle from the bottom. Thus, UGV could keep guiding the UAV to maintain collision-free motions, and the UAV also kept seeing the rack. We achieved zero occlusions across all the benchmarks in this setting. The vehicles, too, finished all the runs without any collisions. As shown in Table 11, this setting has the best performance across occlusion and collision metrics at the expense of a slight increase in computation time and acceleration.

7.4.7. CAVE Generalization to Unseen Enviornments

This section presents the results for the generalization of the learned CVAE to unseen environments. Figure 36(a and (c) demonstrate two warehouse inventory settings where new obstacles are added on the fly (shown with black circles). Figures 36 (b) and (d) correspondingly show how the vehicles adapt to these changes and avoid collisions with the new obstacles while maintaining their LoS free of occlusions. Thanks to the learned CVAE generating trajectories for the initialization that accounts for the new configurations, the proposed MPC can adapt to these abrupt changes and perform the task successfully.

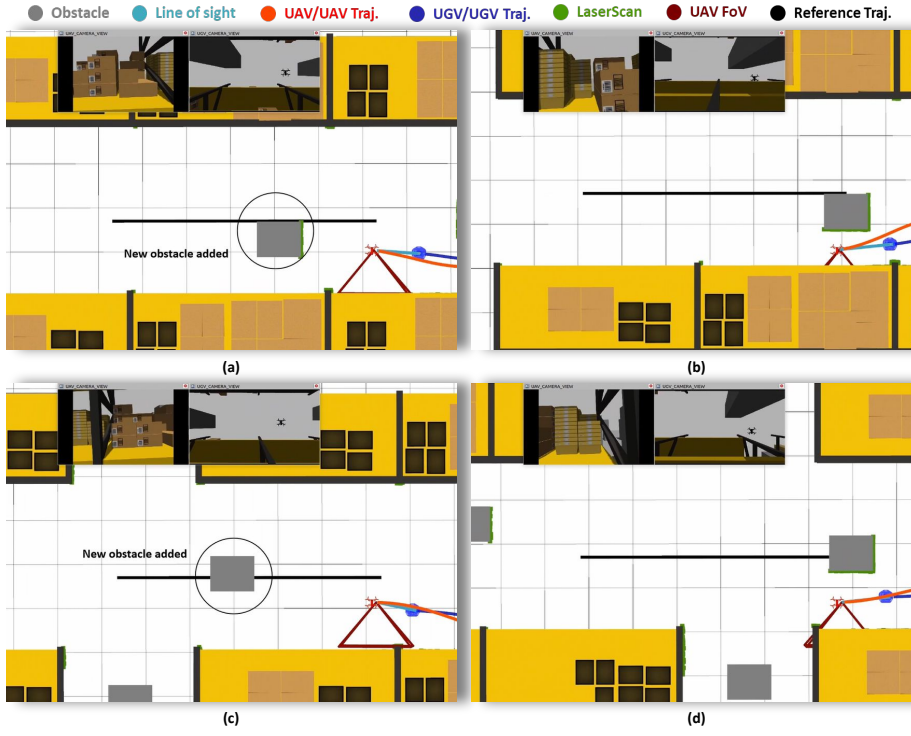
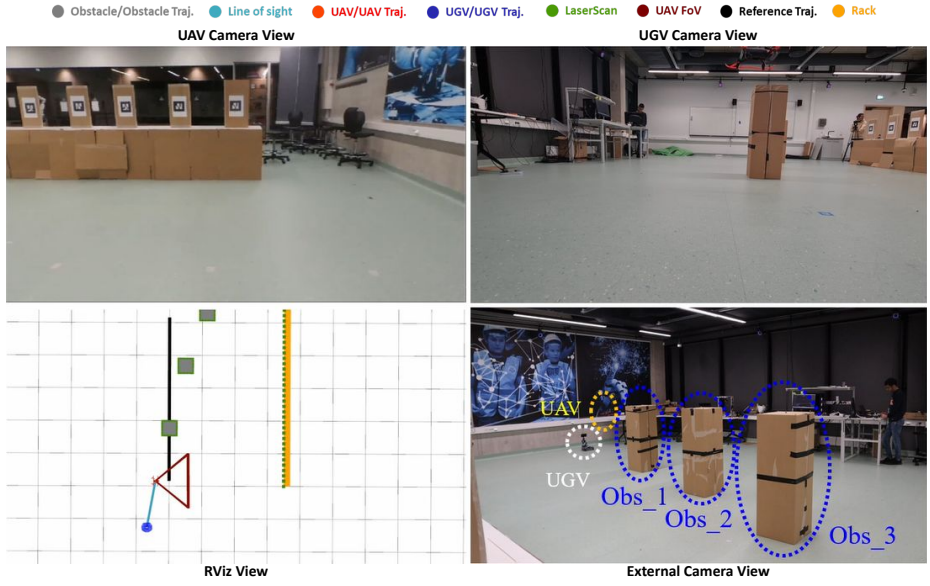


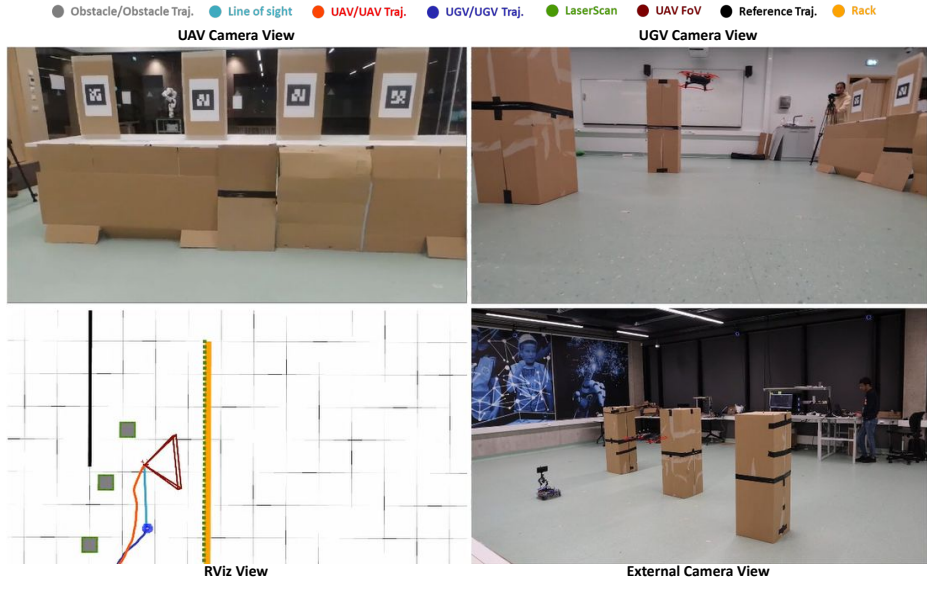
Figure 36. The experiment depicted here illustrates the generalization of the proposed CVAE model to unseen scenarios. (a) and (c) present different parts of an inventory run where obstacles are introduced dynamically (shown as black circles) into the environment. (b) and (d) correspondingly indicate how the vehicles adapt to the new changes. The CVAE model adjusts the initialization trajectories to accommodate these changes, leading to trajectories that avoid collisions/occlusions stemming from the newly added obstacles.

7.4.8. Real World Demonstrator

Presented here is the result of the real-world experiment for the proposed VACNA in a warehouse inventory setting. The environment consists of three obstacles and the items on the racks are represented using fiducial markers(visible in the UAV view). Figure 37 illustrates a static environment, where the UGV guides the UAV to scan the fiducial markers while avoiding the obstacles and tracking the reference trajectory (shown with black in the RViz view) without getting their LoS occluded. Figure 38, however; shows another setting where the obstacles are displaced during the run and UAV/UGV would adapt to new obstacle configurations and avoid collision/occlusions on the fly.

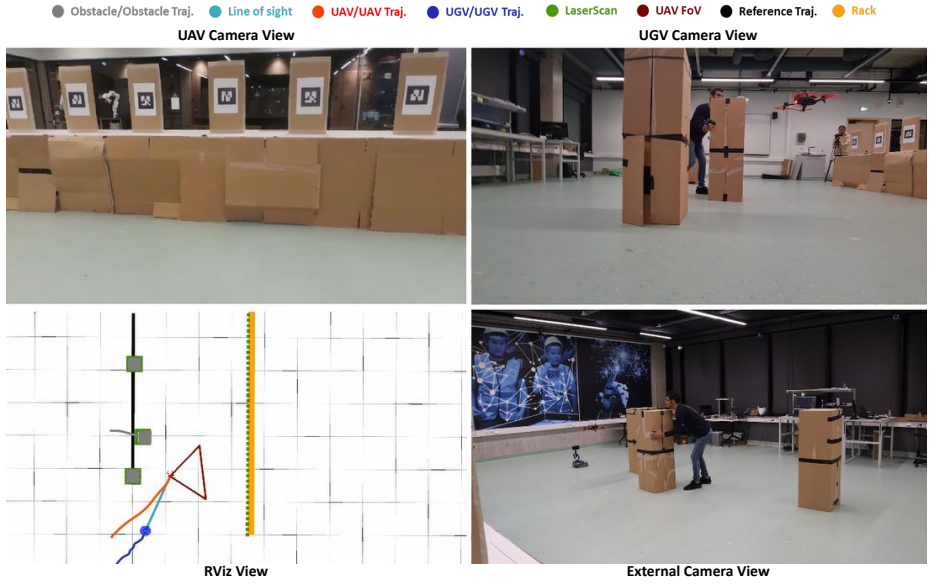


(a)

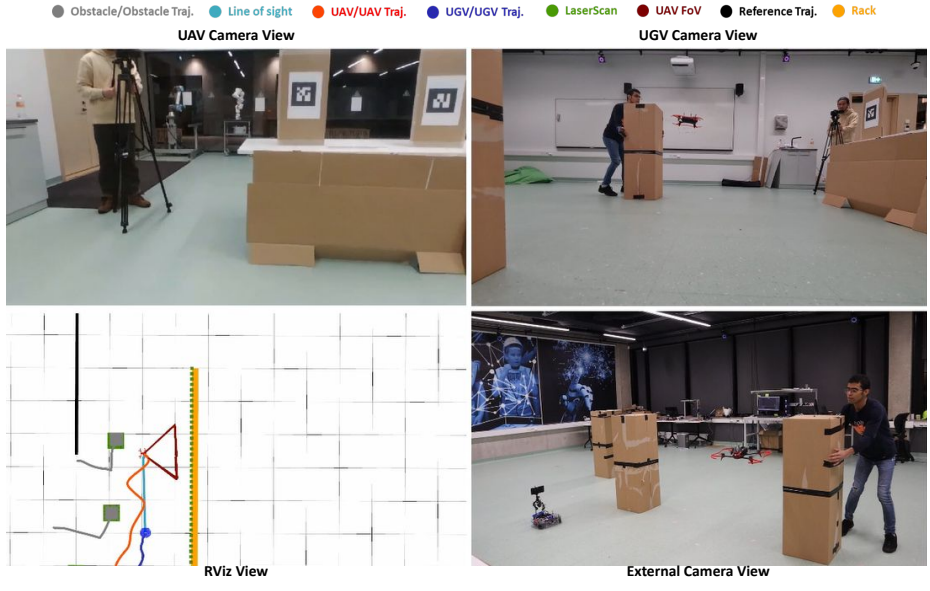


(b)

Figure 37. The real-world experiment of VACNA for scanning warehouse racks in a static environment. The fiducial markers visible in the UAV camera view resemble the items on the racks.



(a)



(b)

Figure 38. The figures show the real-world experiment of VACNA for scanning warehouse racks in a dynamic environment. The obstacles are displaced throughout the run (as can be seen in th external camera view) and UAV/UGV has to avoid collisions while maintaining the visibility of the rack and keeping UAV-UGV LoS free of occlusions.

8. CONCLUSION

The thesis addressed the challenge of Visibility Aware Navigation (VAN) in environments that are highly cluttered and dynamic. The contributions of this work were demonstrated across various scenarios such as target tracking, aerial cinematography, and cooperative navigation for warehouse inventory applications.

The first contribution developed a real-time Model Predictive Control (MPC) that can track a target without any occlusion in the presence of both static and dynamic obstacles. To solve non-convex problems, the optimizer employs multiple layers of reformulation to introduce multi-convexity into the problem structure. The proposed MPC outperforms the state-of-the-art in various metrics, such as visibility score, control effort, and computation time. Moreover, it can handle a rough linear prediction of obstacle and target trajectories. The MPC achieves real-time performance on embedded devices such as NVIDIA Jetson TX2, thanks to the proposed reformulations.

The second contribution brought the advantages of gradient-free trajectory optimization into VAN. A new batch projection operator is proposed and integrated into an MPC pipeline based on the Cross-Entropy Method, which can efficiently handle inequality constraints. This projection results in a significant decrease in constraint violations and outperforms the state-of-the-art in various metrics of target-tracking benchmarks. The proposed approach does not require explicit map building of the environment and can adapt to changes in the environment based on point cloud data from LiDAR.

Finally, the third contribution extended VAN to Visibility-Aware Cooperative Navigation (VACNA) for warehouse inventory applications. The proposed approach allows small-sized UAVs with only a monocular camera to scan warehouse racks while being guided by another UGV that handles all the perception and planning for both vehicles. Additionally, a CVAE model learned from expert demonstrations is used to initialize the optimizer, further increasing the efficiency of the proposed optimizer. The whole planning pipeline can run in real-time on embedded devices such as NVIDIA Jetson Xavier, thanks to the CVAE initialization.

8.1. Limitations

The occlusion model learned in Chapter 6 only considers 2D environments when computing occlusions. This implies that obstacles are included in the calculation with infinite height, rather than their actual height. As a result, trajectories must avoid obstacles by passing them from the side, as passing over them is not considered occlusion-free. While this approach is effective in some cases, it can also be inefficient since trajectories that could pass

over obstacles without obstruction are automatically excluded. However, with the utilization of the PointNet architecture outlined in Section 7.3.3, it is possible to adapt and retrain the occlusion model to account for 3D environments. This modification would allow for more accurate and efficient trajectory planning, as the model would be capable of recognizing obstacles with varying heights and plan trajectories accordingly.

8.2. Future Work

Even though the VACNA approach discussed in Chapter 7 was developed for warehouse inventory applications, it can be readily adapted to other cooperative navigation scenarios. Furthermore, the current implementation of the VACNA approach only supports a single UGV-UAV system. However, the author is exploring options for extending the approach to support single-UGV-multi-UAV settings.

Moreover, the author aims to extend VAN to other robotic systems. For example, visibility-aware planning can be important for autonomous cars while overtaking [103], [104]. Additionally, VAN is also used for navigation in unknown environments, where maintaining visibility of the goal positions can help prevent unexpected collisions [105].

BIBLIOGRAPHY

[1] Boseong Jeon, Yunwoo Lee, and H. Jin Kim. "Integrated Motion Planner for Real-time Aerial Videography with a Drone in a Dense Environment". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1243–1249. DOI: 10.1109/ICRA40945.2020.9196703.

[2] Marios Xanthidis et al. "AquaVis: A Perception-Aware Autonomous Navigation Framework for Underwater Vehicles". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 5410–5417. DOI: 10.1109/IROS51168.2021.9636124.

[3] Shigeru Saito et al. "Indoor Marker-based Localization Using Coded Seamless Pattern for Interior Decoration". In: *2007 IEEE Virtual Reality Conference*. 2007, pp. 67–74. DOI: 10.1109/VR.2007.352465.

[4] Tobias Nägeli et al. "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization". In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1696–1703.

[5] Boseong Felipe Jeon and H. Jin Kim. "Online Trajectory Generation of a MAV for Chasing a Moving Target in 3D Dense Environments". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1115–1121. DOI: 10.1109/IROS40897.2019.8967840.

[6] Sudarshan S. Chawathe. "Marker-Based Localizing for Indoor Navigation". In: *2007 IEEE Intelligent Transportation Systems Conference*. 2007, pp. 885–890. DOI: 10.1109/ITSC.2007.4357816.

[7] Shigeru Saito et al. "Indoor Marker-based Localization Using Coded Seamless Pattern for Interior Decoration". In: *2007 IEEE Virtual Reality Conference*. 2007, pp. 67–74. DOI: 10.1109/VR.2007.352465.

[8] Markus Schreiber, Carsten Knöppel, and Uwe Franke. "LaneLoc: Lane marking based localization using highly accurate maps". In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 449–454. DOI: 10.1109/IVS.2013.6629509.

[9] Davide Falanga et al. "Pampc: Perception-aware model predictive control for quadrotors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2018, pp. 1–8.

[10] Marius Beul et al. "Fast Autonomous Flight in Warehouses for Inventory Applications". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3121–3128. DOI: 10.1109/LRA.2018.2849833.

[11] Haishi Liu et al. "UAV Stocktaking Task-Planning for Industrial Warehouses Based on the Improved Hybrid Differential Evolution

Algorithm". In: *IEEE Transactions on Industrial Informatics* 18.1 (2022), pp. 582–591. DOI: 10.1109/TII.2021.3054172.

[12] Houman Masnavi et al. "Real-Time Multi-Convex Model Predictive Control for Occlusion-Free Target Tracking With Quadrotors". In: *IEEE Access* 10 (2022), pp. 29009–29031. DOI: 10.1109/ACCESS.2022.3157977.

[13] Zdravko I. Botev et al. "Chapter 3 - The Cross-Entropy Method for Optimization". In: *Handbook of Statistics*. Ed. by C.R. Rao and Venu Govindaraju. Vol. 31. Handbook of Statistics. Elsevier, 2013, pp. 35–59. DOI: https://doi.org/10.1016/B978-0-444-53859-8.00003-5. URL: https://www.sciencedirect.com/science/article/pii/B9780444538598000035.

[14] Gordon Wilfong. "Motion planning in the presence of movable obstacles". In: *Annals of Mathematics and Artificial Intelligence* 3 (Jan. 1991), pp. 131–150. DOI: 10.1007/BF01530890.

[15] Alessandro Gasparetto et al. "Path Planning and Trajectory Planning Algorithms: A General Overview". In: *Mechanisms and Machine Science* 29 (Mar. 2015), pp. 3–27. DOI: 10.1007/978-3-319-14705-5_1.

[16] James J Kuffner and Steven M LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[17] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[18] Mohamed Elbanhawi and Milan Simic. "Sampling-based robot motion planning: A review". In: *Ieee access* 2 (2014), pp. 56–77.

[19] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[20] Joshua Bialkowski et al. "Efficient collision checking in sampling-based motion planning". In: *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*. Springer. 2013, pp. 365–380.

[21] Fabian Schwarzer, Mitul Saha, and Jean-Claude Latombe. "Exact collision checking of robot paths". In: *Algorithmic foundations of robotics V* (2004), pp. 25–41.

[22] Stephen R Lindemann and Steven M LaValle. "Current issues in sampling-based motion planning". In: *Robotics Research. The*

Eleventh International Symposium: With 303 Figures. Springer. 2005, pp. 36–54.

[23] Robert Bohlin and Lydia E Kavraki. "Path planning using lazy PRM". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE. 2000, pp. 521–528.

[24] Jinwook Huh and Daniel D Lee. "Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 63–69.

[25] Peter Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/tssc.1968.300136. URL: https://doi.org/10.1109/tssc.1968.300136.

[26] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[27] Oskar Ljungqvist et al. "Lattice-based motion planning for a general 2-trailer system". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 819–824.

[28] Mhand Hifi and Mustapha Michrafy. "A reactive local search-based algorithm for the disjunctively constrained knapsack problem". In: *Journal of the Operational Research Society* 57.6 (2006), pp. 718–726.

[29] Yajue Yang, Jia Pan, and Weiwei Wan. "Survey of optimal motion planning". In: *IET Cyber-Systems and Robotics* 1.1 (2019), pp. 13–19.

[30] Dina Youakim et al. "Multirepresentation, multiheuristic A* search-based motion planning for a free-floating underwater vehicle-manipulator system in unknown environment". In: *Journal of Field Robotics* 37.6 (2020), pp. 925–950.

[31] Nathan Ratliff et al. "CHOMP: Gradient optimization techniques for efficient motion planning". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 489–494.

[32] Mrinal Kalakrishnan et al. "STOMP: Stochastic trajectory optimization for motion planning". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 4569–4574.

[33] Wikipedia. *Trajectory Optimization*. URL: https://en.wikipedia.org/wiki/Trajectory_optimization (visited on 03/31/2020).

[34] Wilko Schwarting et al. "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention". In: *2017 IEEE*

International Conference on Robotics and Automation (ICRA). IEEE. 2017, pp. 1928–1935.

[35] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[36] Zhiwei Zhang et al. "Auto Filmer: Autonomous Aerial Videography Under Human Interaction". In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 784–791. DOI: 10.1109/LRA.2022.3231828.

[37] Junlong Guo et al. "Dynamic Free-Space Roadmap for Safe Quadrotor Motion Planning". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 10523–10528. DOI: 10.1109/IROS47612.2022.9981447.

[38] Jingru Luo and Kris Hauser. "An empirical study of optimal motion planning". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1761–1768.

[39] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[40] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. cite arxiv:1609.04747Comment: Added derivations of AdaMax and Nadam. 2016. URL: http://arxiv.org/abs/1609.04747.

[41] Alex Barclay, Philip E Gill, and J Ben Rosen. "SQP methods and their application to numerical optimal control". In: *Variational Calculus, Optimal Control and Applications: International Conference in Honour of L. Bittner and R. Klötzler, Trassenheide, Germany, September 23–27, 1996*. Springer. 1998, pp. 207–222.

[42] Yuhe Ren, Bo Zhang, and Hong Qiao. "A simple Taylor-series expansion method for a class of second kind integral equations". In: *Journal of Computational and Applied Mathematics* 110.1 (1999), pp. 15–24.

[43] Damián Fernández and Mikhail Solodov. "Stabilized sequential quadratic programming: a survey". In: *Pesquisa Operacional* 34 (2014), pp. 463–479.

[44] Harbir Antil, Drew P Kouri, and Denis Ridzal. "ALESQP: An augmented Lagrangian equality-constrained sqp method for optimization with general constraints". In: *SIAM Journal on Optimization* 33.1 (2023), pp. 237–266.

[45] Cristina Pinneri et al. "Sample-efficient Cross-Entropy Method for Real-time Planning". In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos, and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, Nov.

2021, pp. 1049–1065. URL: https://proceedings.mlr.press/v155/pinneri21a.html.

[46] István Szita and András Lörincz. "Learning Tetris Using the Noisy Cross-Entropy Method". In: *Neural computation* 18 (Jan. 2007), pp. 2936–41. DOI: 10.1162/neco.2006.18.12.2936.

[47] Min Wen and Ufuk Topcu. "Constrained cross-entropy method for safe reinforcement learning". In: *Advances in Neural Information Processing Systems* 31 (2018).

[48] Francis Celeste, Frédéric Dambreville, and J-P Le Cadre. "Optimal path planning using cross-entropy method". In: *2006 9th International Conference on Information Fusion*. IEEE. 2006, pp. 1–8.

[49] Wei Zhang et al. "Applications of the cross-entropy method to importance sampling and optimal control of diffusions". In: *SIAM Journal on Scientific Computing* 36.6 (2014), A2654–A2672.

[50] Rémi Bardenet and Balázs Kégl. "Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm". In: *27th International Conference on Machine Learning (ICML 2010)*. Omnipress. 2010, pp. 55–62.

[51] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. "Model predictive path integral control: From theory to parallel computation". In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 344–357.

[52] Jintasit Pravitra et al. " 1-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 7661–7666.

[53] Hyung-Jin Yoon et al. "Sampling complexity of path integral methods for trajectory optimization". In: *2022 American Control Conference (ACC)*. IEEE. 2022, pp. 3482–3487.

[54] Grady Williams et al. "Aggressive driving with model predictive path integral control". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1433–1440.

[55] Manan S Gandhi et al. "Robust model predictive path integral control: Analysis and performance guarantees". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1423–1430.

[56] Ihab S. Mohamed, Guillaume Allibert, and Philippe Martinet. "Model Predictive Path Integral Control Framework for Partially Observable Navigation: A Quadrotor Case Study". In: *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. 2020, pp. 196–203. DOI: 10.1109/ICARCV50220.2020.9305363.

[57] Taekyung Kim et al. “Smooth Model Predictive Path Integral Control Without Smoothing”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10406–10413. DOI: 10.1109/LRA.2022.3192800.

[58] Nikolaus Hansen and Anne Auger. “CMA-ES: evolution strategies and covariance matrix adaptation”. In: *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*. 2011, pp. 991–1010.

[59] Kouhei Nishida and Youhei Akimoto. “Psa-cma-es: Cma-es with population size adaptation”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2018, pp. 865–872.

[60] Ilya Loshchilov and Frank Hutter. “CMA-ES for hyperparameter optimization of deep neural networks”. In: *arXiv preprint arXiv:1604.07269* (2016).

[61] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)”. In: *Evolutionary computation* 11.1 (2003), pp. 1–18.

[62] Qianhao Wang et al. “Visibility-aware Trajectory Optimization with Application to Aerial Tracking”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 5249–5256. DOI: 10.1109/IROS51168.2021.9636753.

[63] Chong Huang et al. “Act: An autonomous drone cinematography system for action scenes”. In: *IEEE International Conference on Robotics and Automation*. 2018, pp. 7039–7046.

[64] Amirsaman Ashtari et al. “Capturing Subjective First-Person View Shots with Drones for Automated Cinematography”. In: *ACM Transactions on Graphics (TOG)* 39.5 (2020), pp. 1–14.

[65] Boseong Felipe Jeon et al. “Autonomous Aerial Dual-Target Following Among Obstacles”. In: *IEEE Access* 9 (2021), pp. 143104–143120. DOI: 10.1109/ACCESS.2021.3117314.

[66] Yunwoo Lee et al. “Target-visible Polynomial Trajectory Generation within an MAV Team”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 1982–1989. DOI: 10.1109/IROS51168.2021.9636446.

[67] Xinyue Shen et al. “Disciplined multi-convex programming”. In: *2017 29th Chinese Control And Decision Conference (CCDC)*. IEEE. 2017, pp. 895–900.

[68] Prateek Jain, Purushottam Kar, et al. “Non-convex optimization for machine learning”. In: *Foundations and Trends® in Machine Learning* 10.3-4 (2017), pp. 142–336.

[69] Rui Chen et al. “Visibility-Aware Point-Based Multi-View Stereo Network”. In: *IEEE Transactions on Pattern Analysis and Machine*

Intelligence 43.10 (2021), pp. 3695–3708. DOI: 10.1109/TPAMI.2020.2988729.

[70] Zhichao Han et al. "Fast-Tracker: A Robust Aerial System for Tracking Agile Target in Cluttered Environments". In: *arXiv preprint arXiv:2011.03968, to appear in ICRA 2021* (2020).

[71] Christoph Gebhardt et al. "Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 2508–2519.

[72] Patrick Cozzi and Frank Stoner. "Gpu ray casting of virtual globes". In: *ACM SIGGRAPH 2010 Posters*. 2010, pp. 1–1.

[73] Bryan Penin, Paolo Robuffo Giordano, and François Chaumette. "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3725–3732.

[74] Rogerio Bonatti et al. "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming". In: *International Symposium on Experimental Robotics*. Springer. 2018, pp. 119–129.

[75] Tirthankar Bandyopadhyay et al. "Motion planning for people tracking in uncertain and dynamic environments". In: *Workshop on People Detection and Tracking, IEEE International Conference on Robotics and Automation*. 2009.

[76] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. "ACADO toolkit—An open-source framework for automatic control and dynamic optimization". In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312.

[77] Steven G Johnson. *The NLopt nonlinear-optimization package*. 2014.

[78] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. "Online trajectory planning in ROS under kinodynamic constraints with timed-elastic-bands". In: *Robot Operating System (ROS)*. Springer, 2017, pp. 231–261.

[79] Yisha Liu et al. "A novel real-time moving target tracking and path planning system for a quadrotor UAV in unknown unstructured outdoor scenes". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.11 (2018), pp. 2362–2372.

[80] Tom Goldstein and Stanley Osher. "The split Bregman method for L1-regularized problems". In: *SIAM journal on imaging sciences* 2.2 (2009), pp. 323–343.

[81] Sébastien Gros et al. "From linear to nonlinear MPC: bridging the gap via the real-time iteration". In: *International Journal of Control* 93.1 (2020), pp. 62–80.

[82] Thomas Lipp and Stephen Boyd. "Variations and extension of the convex–concave procedure". In: *Optimization and Engineering* 17.2 (2016), pp. 263–287.

[83] Fatemeh Rastgar et al. "A Novel Trajectory Optimization for Affine Systems: Beyond Convex-Concave Procedure". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 1308–1315. DOI: 10.1109/IROS45743.2020.9341566.

[84] Gavin Taylor et al. "Training neural networks without gradients: A scalable admm approach". In: *International conference on machine learning*. PMLR. 2016, pp. 2722–2731.

[85] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[86] Fatemeh Rastgar et al. "GPU Accelerated Convex Approximations for Fast Multi-Agent Trajectory Optimization". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3303–3310. DOI: 10.1109/LRA.2021.3061398.

[87] Mark W Mueller and Raffaello D'Andrea. "A model predictive controller for quadrocopter state interception". In: *2013 European Control Conference (ECC)*. IEEE. 2013, pp. 1383–1389.

[88] Pedro Hespanhol and Rien Quirynen. "A real-time iteration scheme with quasi-Newton Jacobian updates for nonlinear model predictive control". In: *2018 European Control Conference (ECC)*. IEEE. 2018, pp. 1517–1522.

[89] Johannes Meyer et al. "Comprehensive simulation of quadrotor uavs using ros and gazebo". In: *International conference on simulation, modeling, and programming for autonomous robots*. Springer. 2012, pp. 400–411.

[90] Gaël Guennebaud, Benoit Jacob, et al. "Eigen". In: *URl: http://eigen. tuxfamily. org* (2010).

[91] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[92] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018.

[93] Woong Kwon et al. "Robust Autonomous Navigation of Unmanned Aerial Vehicles (UAVs) for Warehouses' Inventory Application". In: *IEEE Robotics and Automation Letters* 5.1 (2020), pp. 243–249. DOI: 10.1109/LRA.2019.2955003.

[94] Sarah Barlow et al. "A Multi-UAS Trajectory optimization Methodology for Complex Enclosed Environments". In: *2019 International*

Conference on Unmanned Aircraft Systems (ICUAS). 2019, pp. 596–604. DOI: 10.1109/ICUAS.2019.8797839.

[95] Diederik P Kingma and Max Welling. "Auto-encoding variational {Bayes}". In: *Int. Conf. on Learning Representations*.

[96] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015.

[97] Charles R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[98] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017.

[99] Renno Raudmäe et al. "ROBOTONT – Open-source and ROS-supported omnidirectional mobile robot for education and research". In: *HardwareX* 14 (2023), e00436. ISSN: 2468-0672. DOI: https://doi.org/10.1016/j.ohx.2023.e00436. URL: https://www.sciencedirect.com/science/article/pii/S2468067223000433.

[100] Cristina Pinneri et al. "Sample-efficient Cross-Entropy Method for Real-time Planning". In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos, and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, 16–18 Nov 2021, pp. 1049–1065. URL: https://proceedings.mlr.press/v155/pinneri21a.html.

[101] Julius Jankowski et al. "VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior". In: *arXiv preprint arXiv:2210.04067* (2022).

[102] Robert Tjarko Lange. "evosax: JAX-based Evolution Strategies". In: *arXiv preprint arXiv:2212.04180* (2022).

[103] Denggui Wang et al. "Occlusion-aware Motion Planning for Autonomous Driving". In: *IEEE Access* (2023), pp. 1–1. DOI: 10.1109/ACCESS.2023.3268072.

[104] Bai Li. "Occlusion-aware on-road autonomous driving: A trajectory planning method considering occlusions of Lidars". In: *Optik* 243 (2021), p. 167347. ISSN: 0030-4026. DOI: https://doi.org/10.1016/j.ijleo.2021.167347. URL: https://www.sciencedirect.com/science/article/pii/S0030402621009906.

[105] Nicholas Mohammad and Nicola Bezzo. “A Robust and Fast Occlusion-based Frontier Method for Autonomous Navigation in Unknown Cluttered Environments”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 6324–6331. DOI: 10.1109/IROS47612.2022.9982059.

ACKNOWLEDGEMENTS

I would like to thank

my supervisors Karl Kruusamäe, Arun Kumar Singh, for their guidance and support not only in this thesis but also in the whole journey of my Ph.D.,

my parents whose unconditional love and profound support give me solace and keeps me going,

Robert Valner who never hesitated to answer my questions or help me whenever I stumbled upon problems,

and my other friends including Iman, Fatemeh, Kaveh, and Mozhgan who helped me get this work done.

SISUKOKKUVÕTE

Nähtavust arvestav navigatsioon

Liikumise planeerimine on robotite ja automatiseerimise põhikomponent, mis hõlmab järjestikuste liikumiste või tegevuste määramist konkreetse eesmärgi saavutamiseks, samal ajal rahuldades erinevaid piiranguid. Käesolev doktoritöö tegeleb liikumise planeerimise probleemi variantiga, mida nimetatakse nähtavust arvestavaks navigatsiooniks (VAN), kus eesmärk on liikuda kahe punkti vahel või jälgida viittraektoori, hoides samal ajal teatud objekte roboti vaateväljas. VAN leiab rakendust erinevates valdkondades, nagu sihtmärgi jälgimine, õhust kinematograafia, märgipõhine lokaliseerimine ja koostööalane laopidamine. Sihtmärgi jälgimisel ja õhus kinematograafias tagab see sihtmärgi selge nähtavuse, vältides takistusi. Olekuhinnangu ja lokaliseerimise korral parandab see täpsust ja vastupidavust, planeerides teid, mis säilitavad märke, nagu märgistusmärgid või märgutuled, nähtavuse. VAN võib samuti luua aluse koostööks arvutusressursside ja tajude poolest rikkaliku maapealse ilma juhitud sõiduki (UGV) ning väikeste mõõtmetega ilma juhitud õhusõiduki (UAV) vahel. Nähtavust arvestav navigatsioon (VAN) võib aidata UGV-l juhtida UAV-d seni, kuni nende vaateväli pole takistatud.

Siiski on VAN-ga seotud kaks põhilist väljakutset. Esiteks tuleb välja töötada robustsed algoritmid, mis suudavad käidelda keerulisi piiranguid, näiteks nähtavust, isegi tihedalt segatud ja dünaamilistes keskkondades. Täpsemalt tuleb navigatsiooni käigus vältida takistuste poolt tekitatud varjatud vaatevälja takistusi. Teiseks tuleks lahendused reaalajas toimima panna sisseehitatud seadmetes, mida saab kanda väikestel robotitel nagu droonid. Käesolev doktoritöö pakub välja uudsed liikumise planeerimise algoritmid nähtavust arvestava navigatsiooni kontekstis, mis suudavad käidelda keerulisi keskkondi staatiliste ja dünaamiliste takistustega ning saavutada samas reaalajas jõudlust piiratud ressurssidega seadmetel. Algoritme on ulatuslikult testitud nii simulatsiooni- kui ka reaalmaailma keskkondades ning võrreldud erinevate uuenduslike lähenemisviisidega. Pakutud lähenemisviisid ületavad uuenduslike meetodite tulemusi erinevates mõõdikutel, nagu varjatud oleku aeg, kontrollivaeva ja arvutusaeg. Lisaks töötavad algoritmid reaalajas sisseehitatud seadmetes, nagu NVIDIA Jetson TX2 ja Xavier.

PUBLICATIONS

CURRICULUM VITAE

Personal data

Name:	Houman Masnavi
Date of birth:	17.01.1995
Contact:	houman.masnavi@ut.ee
Current Position:	Junior Research Fellow in Robotics

Education

2020–2023	Ph.D. Candidate, University of Tartu
2018–2020	MSc. Robotics and Computer Engineering (*cum laude*), University of Tartu
2014–2018	BSc. Computer Engineering (*cum laude*), University of Zabol

Employment

2021–2023	Junior Research Fellow in Robotics, Institute of Technology, University of Tartu
2020–2021	Specialist of Robotics, Institute of Technology, University of Tartu

Scientific work

Main fields of interest:

- Motion Planning and Control
- Optimization
- Machine Learning and Deep Learning

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Houman Masnavi
Sünniaeg:	17.01.1995
E-mail:	houman.masnavi@ut.ee
Praegune positsioon:	robootika nooremteadur

Haridus

2020–2023	Tartu Ülikool, Loodus- ja täppisteaduste valdkond, tehnoloogiainstituut, doktoriõpe
2018–2020	Tartu Ülikool, Loodus- ja täppisteaduste valdkond, tehnoloogiainstituut, robootika ja arvutitehnika, magistriõpe (*cum laude*)
2014–2018	Zaboli Ülikool, arvutitehnika, bakalaureuseõpe, (*cum laude*)

Teenistuskäik

2020–2023	Tartu Ülikool, Loodus- ja täppisteaduste valdkond, tehnoloogiainstituut, robootika nooremteadur
2020–2020	Tartu Ülikool, Loodus- ja täppisteaduste valdkond, tehnoloogiainstituut, robootika spetsialist

Teadustegevus

Peamised uurimisvaldkonnad:

- Liikumise planeerimine ja juhtimine
- Optimeerimine
- Masinõpe ja süvaõpe

DISSERTATIONES TECHNOLOGIAE UNIVERSITATIS TARTUENSIS

1. **Imre Mäger**. Characterization of cell-penetrating peptides: Assessment of cellular internalization kinetics, mechanisms and bioactivity. Tartu 2011, 132 p.
2. **Taavi Lehto**. Delivery of nucleic acids by cell-penetrating peptides: application in modulation of gene expression. Tartu 2011, 155 p.
3. **Hannes Luidalepp**. Studies on the antibiotic susceptibility of *Escherichia coli*. Tartu 2012, 111 p.
4. **Vahur Zadin**. Modelling the 3D-microbattery. Tartu 2012, 149 p.
5. **Janno Torop**. Carbide-derived carbon-based electromechanical actuators. Tartu 2012, 113 p.
6. **Julia Suhorutšenko**. Cell-penetrating peptides: cytotoxicity, immunogenicity and application for tumor targeting. Tartu 2012, 139 p.
7. **Viktoryia Shyp**. G nucleotide regulation of translational GTPases and the stringent response factor RelA. Tartu 2012, 105 p.
8. **Mardo Kõivomägi**. Studies on the substrate specificity and multisite phosphorylation mechanisms of cyclin-dependent kinase Cdk1 in *Saccharomyces cerevisiae*. Tartu, 2013, 157 p.
9. **Liis Karo-Astover**. Studies on the Semliki Forest virus replicase protein nsP1. Tartu, 2013, 113 p.
10. **Piret Arukuusk**. NickFects–novel cell-penetrating peptides. Design and uptake mechanism. Tartu, 2013, 124 p.
11. **Piret Villo**. Synthesis of acetogenin analogues. Asymmetric transfer hydrogenation coupled with dynamic kinetic resolution of α-amido-β-keto esters. Tartu, 2013, 151 p.
12. **Villu Kasari**. Bacterial toxin-antitoxin systems: transcriptional cross-activation and characterization of a novel *mqsRA* system. Tartu, 2013, 108 p.
13. **Margus Varjak**. Functional analysis of viral and host components of alphavirus replicase complexes. Tartu, 2013, 151 p.
14. **Liane Viru**. Development and analysis of novel alphavirus-based multifunctional gene therapy and expression systems. Tartu, 2013, 113 p.
15. **Kent Langel**. Cell-penetrating peptide mechanism studies: from peptides to cargo delivery. Tartu, 2014, 115 p.
16. **Rauno Temmer**. Electrochemistry and novel applications of chemically synthesized conductive polymer electrodes. Tartu, 2014, 206 p.
17. **Indrek Must**. Ionic and capacitive electroactive laminates with carbonaceous electrodes as sensors and energy harvesters. Tartu, 2014, 133 p.
18. **Veiko Voolaid**. Aquatic environment: primary reservoir, link, or sink of antibiotic resistance? Tartu, 2014, 79 p.
19. **Kristiina Laanemets**. The role of SLAC1 anion channel and its upstream regulators in stomatal opening and closure of *Arabidopsis thaliana.* Tartu, 2015, 115 p.

20. **Kalle Pärn**. Studies on inducible alphavirus-based antitumour strategy mediated by site-specific delivery with activatable cell-penetrating peptides. Tartu, 2015, 139 p.
21. **Anastasia Selyutina**. When biologist meets chemist: a search for HIV-1 inhibitors. Tartu, 2015, 172 p.
22. **Sirle Saul**. Towards understanding the neurovirulence of Semliki Forest virus. Tartu, 2015, 136 p.
23. **Marit Orav**. Study of the initial amplification of the human papillomavirus genome. Tartu, 2015, 132 p.
24. **Tormi Reinson**. Studies on the Genome Replication of Human Papillomaviruses. Tartu, 2016, 110 p.
25. **Mart Ustav Jr**. Molecular Studies of HPV-18 Genome Segregation and Stable Replication. Tartu, 2016, 152 p.
26. **Margit Mutso**. Different Approaches to Counteracting Hepatitis C Virus and Chikungunya Virus Infections. Tartu, 2016, 184 p.
27. **Jelizaveta Geimanen**. Study of the Papillomavirus Genome Replication and Segregation. Tartu, 2016, 168 p.
28. **Mart Toots**. Novel Means to Target Human Papillomavirus Infection. Tartu, 2016, 173 p.
29. **Kadi-Liis Veiman**. Development of cell-penetrating peptides for gene delivery: from transfection in cell cultures to induction of gene expression *in vivo*. Tartu, 2016, 136 p.
30. **Ly Pärnaste**. How, why, what and where: Mechanisms behind CPP/cargo nanocomplexes. Tartu, 2016, 147 p.
31. **Age Utt**. Role of alphavirus replicase in viral RNA synthesis, virus-induced cytotoxicity and recognition of viral infections in host cells. Tartu, 2016, 183 p.
32. **Veiko Vunder**. Modeling and characterization of back-relaxation of ionic electroactive polymer actuators. Tartu, 2016, 154 p.
33. **Piia Kivipõld**. Studies on the Role of Papillomavirus E2 Proteins in Virus DNA Replication. Tartu, 2016, 118 p.
34. **Liina Jakobson**. The roles of abscisic acid, CO_2, and the cuticle in the regulation of plant transpiration. Tartu, 2017, 162 p.
35. **Helen Isok-Paas**. Viral-host interactions in the life cycle of human papillomaviruses. Tartu, 2017, 158 p.
36. **Hanna Hõrak**. Identification of key regulators of stomatal CO_2 signalling via O_3-sensitivity. Tartu, 2017, 260 p.
37. **Jekaterina Jevtuševskaja**. Application of isothermal amplification methods for detection of *Chlamydia trachomatis* directly from biological samples. Tartu, 2017, 96 p.
38. **Ülar Allas**. Ribosome-targeting antibiotics and mechanisms of antibiotic resistance. Tartu, 2017, 152 p.
39. **Anton Paier**. Ribosome Degradation in Living Bacteria. Tartu, 2017, 108 p.
40. **Vallo Varik**. Stringent Response in Bacterial Growth and Survival. Tartu, 2017, 101 p.

41. **Pavel Kudrin**. In search for the inhibitors of *Escherichia coli* stringent response factor RelA. Tartu, 2017, 138 p.
42. **Liisi Henno**. Study of the human papillomavirus genome replication and oligomer generation. Tartu, 2017, 144 p.
43. **Katrin Krõlov**. Nucleic acid amplification from crude clinical samples exemplified by *Chlamydia trachomatis* detection in urine. Tartu, 2018, 118 p.
44. **Eve Sankovski**. Studies on papillomavirus transcription and regulatory protein E2. Tartu, 2018, 113 p.
45. **Morteza Daneshmand**. Realistic 3D Virtual Fitting Room. Tartu, 2018, 233 p.
46. **Fatemeh Noroozi**. Multimodal Emotion Recognition Based Human-Robot Interaction Enhancement. Tartu, 2018, 113 p.
47. **Krista Freimann**. Design of peptide-based vector for nucleic acid delivery in vivo. Tartu, 2018, 103 p.
48. **Rainis Venta**. Studies on signal processing by multisite phosphorylation pathways of the *S. cerevisiae* cyclin-dependent kinase inhibitor Sic1. Tartu, 2018, 155 p.
49. **Inga Põldsalu**. Soft actuators with ink-jet printed electrodes. Tartu, 2018, 85 p.
50. **Kadri Künnapuu**. Modification of the cell-penetrating peptide PepFect14 for targeted tumor gene delivery and reduced toxicity. Tartu, 2018, 114 p.
51. **Toomas Mets**. RNA fragmentation by MazF and MqsR toxins of *Escherichia coli*. Tartu, 2019, 119 p.
52. **Kadri Tõldsepp**. The role of mitogen-activated protein kinases MPK4 and MPK12 in CO_2-induced stomatal movements. Tartu, 2019, 259 p.
53. **Pirko Jalakas**. Unravelling signalling pathways contributing to stomatal conductance and responsiveness. Tartu, 2019, 120 p.
54. **S. Sunjai Nakshatharan**. Electromechanical modelling and control of ionic electroactive polymer actuators. Tartu, 2019, 165 p.
55. **Eva-Maria Tombak**. Molecular studies of the initial amplification of the oncogenic human papillomavirus and closely related nonhuman primate papillomavirus genomes. Tartu, 2019, 150 p.
56. **Meeri Visnapuu**. Design and physico-chemical characterization of metal-containing nanoparticles for antimicrobial coatings. Tartu, 2019, 138 p.
57. **Jelena Beljantseva**. Small fine-tuners of the bacterial stringent response – a glimpse into the working principles of Small Alarmone Synthetases. Tartu, 2020, 104 p.
58. **Egon Urgard**. Potential therapeutic approaches for modulation of inflammatory response pathways. Tartu, 2020, 120 p.
59. **Sofia Raquel Alves Oliveira**. HPLC analysis of bacterial alarmone nucleotide (p)ppGpp and its toxic analogue ppApp. Tartu, 2020, 122 p.
60. **Mihkel Örd**. Ordering the phosphorylation of cyclin-dependent kinase Cdk1 substrates in the cell cycle. Tartu, 2021, 228 p.
61. **Fred Elhi**. Biocompatible ionic electromechanically active polymer actuator based on biopolymers and non-toxic ionic liquids. Tartu, 2021, 140 p.

62. **Liisi Talas**. Reconstructing paleo-diversity, dynamics and response of eukaryotes to environmental change over the Late-Glacial and Holocene period in lake Lielais Svētiņu using sedaDNA. Tartu, 2021, 118 p.
63. **Livia Matt**. Novel isosorbide-based polymers. Tartu, 2021, 118 p.
64. **Koit Aasumets**. The dynamics of human mitochondrial nucleoids within the mitochondrial network. Tartu, 2021, 104 p.
65. **Faiza Summer**. Development and optimization of flow electrode capacitor technology. Tartu, 2022, 109 p.
66. **Olavi Reinsalu**. Cancer-testis antigen MAGE-A4 is incorporated into extracellular vesiclesand is exposed to the surface. Tartu, 2022, 130 p.
67. **Tetiana Brodiazhenko**. RelA-SpoT Homolog enzymes as effectors of Toxin-Antitoxin systems. Tartu, 2022, 132 p.
68. **Georg-Marten Lanno**. Development of novel antibacterial drug delivery systems as wound scaffolds using electrospinning technology. Tartu, 2022, 175 p.
69. **Liubov Cherkashchenko**. New insights into alphaviral nsP2 functions. Tartu, 2023, 171 p.
70. **Kristina Kiisholts**. Peptide-based drug carriers and preclinical nanomedicine applications for endometriosis treatment. Tartu, 2023, 138 p.
71. **Kai Rausalu**. Alphaviral nsP2 protease: From requirements for functionality to inhibition. Tartu, 2023, 175 p.
72. **Laura Sandra Lello.** Unraveling the intricate nature of the alphavirus RNA replicase. Tartu, 2023, 219 p.