Robust Imaging with Sum of Squares Barrier Verification

Gabriel Margolis

December 12, 2019

1 Introduction

We implement a controller for a camera-equipped Unmanned Aerial Vehicle (UAV) to image some point of interest in a cluttered environment. We model an UAV with nonlinear Dubins dynamics, subject to bounded uncertain disturbances. We apply Sum of Squares (SOS) programming to produce certificates of robust imaging success as well as robust safety. These certificates are used to identify robustly safe motion primitives which can be used for offline or online motion planning.

2 Occlusion Model

We define the occlusion space of an obstacle as the set of points for which some observation point is blocked from view by that obstacle. For any convex obstacle defined by $g(x) \geq 0$ and observation point p, there exists a convex occlusion space represented by a function $h(x) \geq 0$.

If we relax any requirement about whether space inside the obstacle is represented as occluded, we can represent the occlusion space of any convex obstacle as a truncated cone, as follows:

The border of any obstacle consisting of convex set $\mathcal{X} \in R^2$ has exactly two points $T1$, $T2$ whose tangent lines pass through some external point of interest $O \notin \mathcal{X}$. The occlusion space of this obstacle is restricted to the same cone defined by these lines in which the obstacle itself lies. Finally, to account for the fact that some points in this cone are between the obstacle and O and therefore not occluded by the obstacle, we construct a line between $T1$, $T2$ and truncate the cone so that the points in the occluded set lie on the opposite side of this line from O. This is illustrated in Figure 1.

In this work, we consider circular obstacles in R^2 defined by

$$g(x,y) = 1 - (\frac{x - x_C}{r})^2 - (\frac{y - y_C}{r})^2 \geq 0$$

For such an obstacle, the truncated cone defining the occlusion space with respect to observation point $p = (p_x, p_y)$ is analytically derived:

$$d = \sqrt{(x_O - x_C)^2 + (y_O - y_C)^2} \qquad l = \sqrt{d^2 - r^2}$$

$$(x_{T1}, y_{T1}, x_{T2}, y_{T2}) = \text{circcirc}(x_O, y_O, l, x_C, y_C, r)$$

$$h^{(1)}(x,y) = -(x_{T1} - x_o)(y - y_O) + (y_{T1} - y_O)(x - x_O) \geq 0$$
$$h^{(2)}(x,y) = (x_{T2} - x_o)(y - y_O) - (y_{T2} - y_O)(x - x_O) \geq 0$$
$$h^{(3)}(x,y) = (x_{T2} - x_{T1})(y - y_{T1}) - (y_{T2} - y_{T1})(x - x_{T1}) \geq 0$$

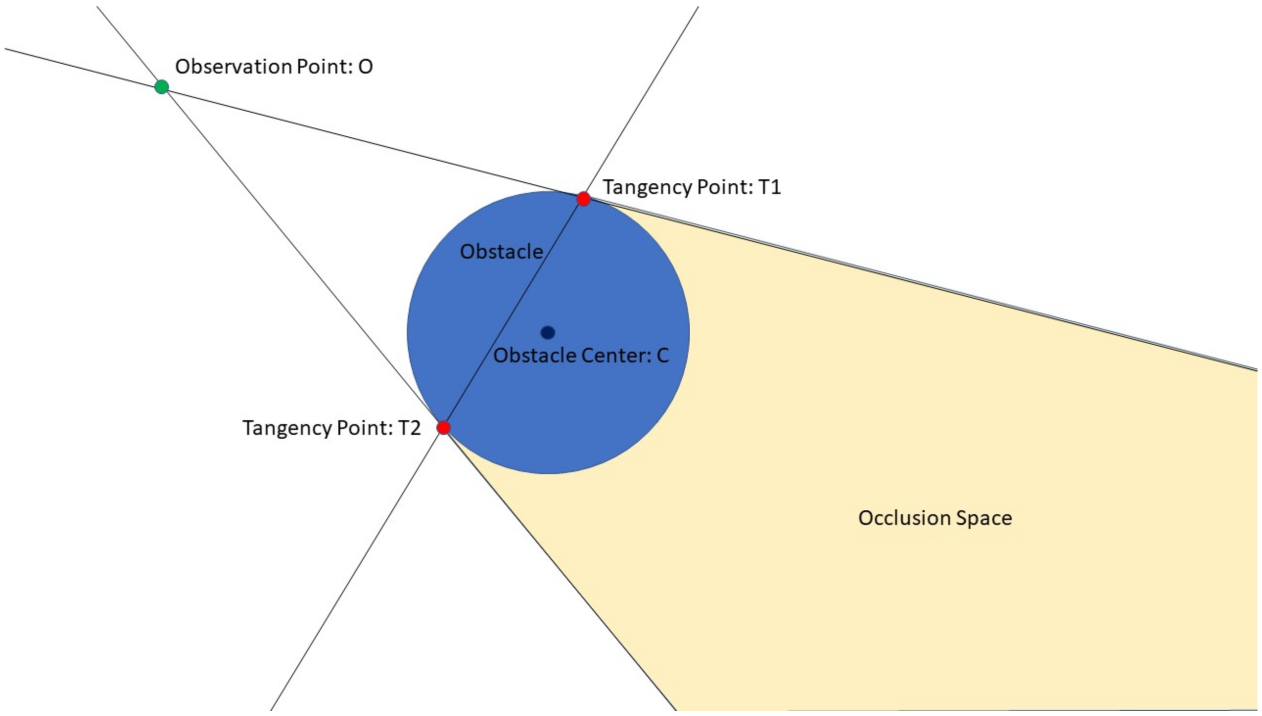

Figure 1: Geometric occlusion model for a circular obstacle.

3 Motion Model

We model our imaging vehicle using a Dubins car model with equations of motion:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -v\sin\phi + w \\ v\cos\phi \\ u \end{bmatrix}$$

where $\mathbf{x}$ is the position and orientation of the vehicle in the environment, $\mathbf{u}$ is the control input, and $\mathbf{w}$ is a stochastic disturbance term.

We consider our random disturbance, $\mathbf{w}$, to be a scalar bounded within the uncertainty set $[-0.03, 0.03]$. This will allow us to define and search for a plan that is robust to collisions or occlusions.

4 Evaluating Motion Primitives

We use SOS programming to evaluate the robustness of motion primitives to two conditions: collision and occlusion.

A Lyapunov Barrier function provides a robust certificate of safety for a dynamical system. The Lyapunov function $V(x)$ guarantees that the system will remain outside the set X_s for all disturbances within the uncertainty set ω if it meets the following conditions:

$$V(x_0) = 0 \qquad V(x) \geq 1, \forall x \in X_s \qquad -\frac{dV}{dt} \geq 0, \forall \omega$$

In the third equation, $\frac{dV}{dt} = \frac{dV}{dx}\frac{dx}{dt}$, which imposes a constraint on V with respect to the system dynamics.

4.1 Collision Safety Verification

Given an obstacle defined by $g(x, y) \geq 0$, we formulate an SOS programming problem to identify a Lyapunov Barrier certificate of robust safety over the set of all states in the neighborhood X:

Find $c(i)$, the coefficients of polynomial V, and functions s_n such that

$$V - 1 - s_1 * g \in SOS \qquad -\frac{dV}{dt} - s_2 * g_\omega - s_3 * X \in SOS \qquad s_1, s_2, s_3 \in SOS \qquad c(1) = 0$$

The SOS program as posed can be efficiently solved using YALMIP and MOSEK in MATLAB. Note that while finding a SOS function V guarantees robust safety, failure to find such a function does not preclude the possibility that a control command is robustly safe. Increasing the degree of V can lead to an improved rate of barrier certificate identification, but yields slower solution times.

4.2 Occlusion Safety Verification

Given an obstacle defined by $g(x, y) \geq 0$ with occlusion space defined by $h^{(1)}, h^{(2)}, h^{(3)} \geq 0$, we formulate an SOS program to search for a Lyapunov Barrier certificate of robust imaging capability over the set of all states in the neighborhood X:

Find $c(i)$, the coefficients of polynomial V, and functions s_n such that

$$V - 1 - s_1 * h^{(1)} - s_2 * h^{(2)} - s_3 * h^{(3)} \in SOS \qquad -\frac{dV}{dt} - s_4 * g_\omega - s_5 * X \in SOS$$
$$s_1, s_2, s_3, s_4, s_5 \in SOS \qquad c(1) = 0$$

The SOS problem as posed can be efficiently solved using YALMIP and MOSEK in MATLAB. Note that while finding a SOS function V guarantees robust imaging success, failure to find such a function does not preclude the possibility that a control command is robustly non-occluded. Increasing the degree of V can lead to an improved rate of barrier certificate identification, but yields slower solution times.

5 Constructing a Robust Imaging Plan

We consider the online selection of robustly collision-free and occlusion-free motions.

Our environment in R^2 contains n circular obstacles, defined by functions $g_1 ... g_n$. An imaging target is stationary at (x_O, y_O). The imaging vehicle's position is denoted as $\mathbf{x} = (x, y, \phi)$, and we select our coordinate frame such that the starting position is $\mathbf{x} = (0, 0, 0)$.

We define motion primitives for our imaging vehicle: $u = -50(\phi - \phi_{des})$ for $\phi_{des} \in [-\frac{\pi}{3}, -\frac{\pi}{4}, -\frac{\pi}{6}, -\frac{\pi}{12}, 0, \frac{\pi}{12}, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}]$. At each timestep, we search for certificates of robust safety and robust imaging success with respect to the obstacles in the environment. If we find a robustly occlusion-free and collision-free primitive, we select it. Otherwise, we select a collision-free primitive.

A sample robust imaging run is shown in Figure . Green circles represent obstacles, the blue line represents the actuated UAV trajectory, and red points represent samples of occluded space for reference. The UAV executes motion primitives which robustly keep it within the visible region for the entire run.

6 Conclusion and Future Work

In this work, we have demonstrated a method for the online selection of safe commands for a nonlinear dynamical system which greedily optimizes for robust imaging success in the presence of convex obstacles and bounded uncertain disturbances. Although we made several simplifying assumptions, we expect that our method will work well when several of these assumptions are relaxed:

- While we considered Dubins vehicle dynamics, our method can be approximately applied to imaging vehicles with arbitrary nonlinear dynamics via Taylor series expansion.

- Our method of occlusion detection, demonstrated here for circular obstacles, is valid for any convex obstacle in two dimensions given a method for calculating the obstacle's relevant tangent points.

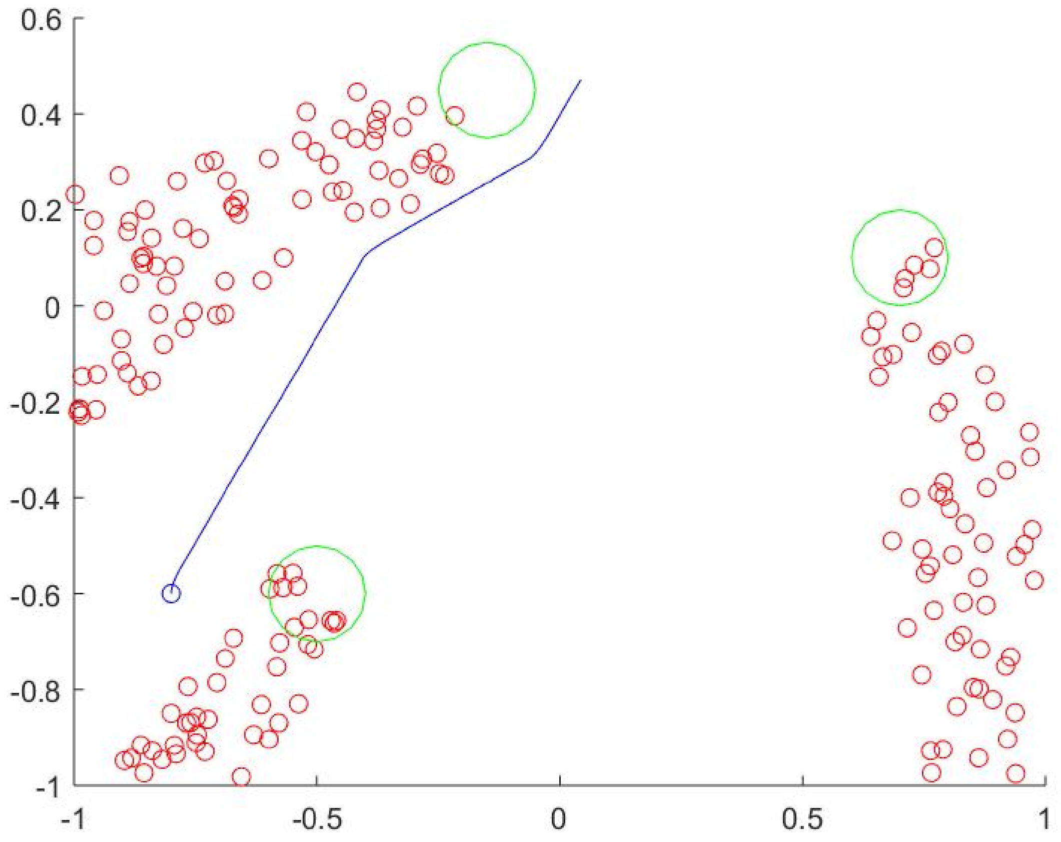

Figure 2: A sample robust imaging plan.

- Uncertainty in the location of the imaging targets or the obstacles can be introduced and modeled within the given framework.

Our proposed method also has several disadvantages which will be worthwhile to address in future research:

- Defining occlusion spaces in n-dimensions will require a more rigorous geometric treatment than presented here. Expressing the occlusion space of a convex obstacle in closed form, possibly as a truncated n-dimensional cone, could be the subject of future work.
- Online planning requires fast decision-making algorithms. While previous work such as [2] has applied SOS programming to online planning, our implementation in MATLAB is not fast enough for real-time application. Using SDSOS programming might allow us to improve runtime without sacrificing much performance.
- Extended-horizon planning may be necessary in some problems to achieve a goal - this is more relevant in the robust imaging scenario, where the vehicle may pass in and out of occluded areas, than the robust obstacle avoidance scenario, where obstacles must be avoided at all costs. Although we considered applying an algorithm for planning with motion primitives to this problem, the presence of continuous state uncertainty makes the application of traditional search-based planning algorithms impossible. A planning framework incorporating state uncertainty will be necessary to solve this type of problem.

References

[1] Ashkan Jasour, *MIT 16.S498 Risk Aware and Robust Nonlinear Planning Course Notes.*

[2] Amir Ali Ahmadi and Anirudha Majumdar, *Some Applications of Polynomial Optimization in Operations Research and Real-Time Decision Making.* arXiv