Noname manuscript No.
(will be inserted by the editor)

Collision Avoidance for Mobile Robots with Limited Sensing and Limited Information about Moving Obstacles

Dung Phan · Junxing Yang · Radu Grosu · Scott A. Smolka · Scott D. Stoller

Received: date / Accepted: date

Abstract This paper addresses the problem of safely navigating a mobile robot with limited sensing capability and limited information about stationary and moving obstacles. We consider two sensing limitations: blind spots between sensors and limited sensing range. We study three notions of safety: (1) *static safety*, which ensures collision-freedom with respect to stationary obstacles, (2) *passive safety*, which ensures collision-freedom while the robot is moving, and (3) *passive friendly safety*, which ensures the robot leaves sufficient room for obstacles to avoid collisions. We present a runtime approach, based on the Simplex architecture, to ensure these safety properties. To obtain the switching logic for the Simplex architecture, we identify a set of constraints on the sensor readings whose satisfaction at time t guarantees that the robot will still be able to ensure the safety property at time $t + \Delta t$, regardless of how it navigates during that time interval. Here, Δt is the period with which the switching logic is executed and is bounded by a function of the maximum velocity and braking power of the robot and the range of the sensors. To the best of our knowledge, this work is the first that provides runtime assurance that an autonomous mobile robot with limited sensing can navigate safely with

Dung Phan
Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
E-mail: dphan@cs.stonybrook.edu

Junxing Yang
Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
E-mail: junyang@cs.stonybrook.edu

Radu Grosu
Department of Computer Science, Vienna University of Technology, Vienna, Austria
E-mail: radu.grosu@tuwien.ac.at

Scott A. Smolka
Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
E-mail: sas@cs.stonybrook.edu

Scott D. Stoller
Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
Tel.: +1-631-632-1627
Fax: +1-631-632-8334
E-mail: stoller@cs.stonybrook.edu

limited information about obstacles. The limited information about obstacles is used to derive an over-approximation of the set of nearby obstacle points.

Keywords Mobile robots · Simplex architecture · Collision avoidance · Blind spots

1 Introduction

Autonomous mobile robots are becoming increasingly popular. They are used in homes, warehouses, hospitals and even on the roads. In most applications, collision avoidance is a vital safety requirement. Ideally, the robots would have 360° field-of-view. One approach to achieve this is to closely place a sufficient number of sensors (e.g., infrared, laser, or ultrasound) on the robot. The biggest problem with this approach is interference between sensors. It is difficult to install the sensors close enough to achieve 360° sensing while at the same time avoiding interference.[1] In addition, the use of numerous sensors increases cost, power consumption, weight, and size of the robot. Another option is to use sensors that have wide angle of observation, such as the Hokuyo URG-04LX laser range finder with 240° range. This approach, however, adds thousands of dollars to the cost. Due to these difficulties, 360° sensing capability is often not a practical option. Consequently, many well-known cost-effective mobile robots, such as E-puck, Khepera III, Quickbot and AmigoBot, lack this capability. These robots have a small number of narrow-angle infrared or ultrasound sensors that do not provide 360° field-of-view. The resulting blind spots between sensors make the robot vulnerable to collision with undetected obstacles that are narrow enough to fit in the blind spots.

One approach to prevent such collisions is for the robot to repeatedly stop or slow down (depending on the sensor range), rotate back and forth to sweep its sensors across the original blind spots, and then continue (this assumes the robot can rotate without moving too much). This approach, however, is inefficient: it significantly slows the robot and wastes power. A similar approach is to mount the sensors so that they can rotate relative to the robot. Unfortunately, this approach adds hardware and software complexity, increases power usage, and limits the maximum safe speed of the robot (depending on the rotation speed of the sensors).

In this paper, we present a runtime approach based on the Simplex architecture [18,19] to ensure safety of robots with limited field-of-view and limited sensing range in environments where obstacles are polyhedral and satisfy reasonable assumptions about minimum internal angle and minimum edge length. One example of such an environment is an automated warehouse, where some information is known about the shapes and sizes of shelving racks, pallets, etc. Our work is also applicable to robots designed with 360° sensing capability that temporarily acquire blind spots due to of one or more sensor failures. Our approach does not suffer from the above disadvantages; i.e., the robot does not have to rotate the sensors or repeatedly stop and rotate itself. The trade-off is that our approach requires some assumptions, albeit weak ones, about obstacles.

[1] Cameras, i.e., sensing based on computer vision, do not interfere with each other but are less common as a basis for navigation due to other disadvantages: cameras depend on good lighting; accurate ranging from stereoscopic vision is impossible on small robots and is generally less accurate than and requires significantly more computational power than ranging from lasers, ultrasound, infrared, *etc.*

We consider three safety properties: (1) *static safety*, which ensures collision-freedom with respect to stationary obstacles; (2) *passive safety*, which ensures collision-freedom while the robot is moving; and (3) *passive friendly safety*, which ensures the robot leaves sufficient room for obstacles to avoid collisions. The notions of passive safety and passive friendly safety we use are adopted from those introduced in [10]. Passive safety ensures that if a collision is imminent, the robot can brake and stop before the collision occurs. It is *passive* in the sense that the robot never actively collides with an obstacle. While the robot may stop in time, it may also create a situation in which the obstacle is left with no choice but to collide with the robot. Passive friendly safety prevents these situations by ensuring that after the robot stops, the obstacle always has enough braking distance to avoid a collision.

Our approach allows one to determine the information needed about obstacles to guarantee a desired safety property. Specifically, to guarantee static safety, no information about an obstacle's movement is needed. If one wants to guarantee passive safety w.r.t. moving obstacles, then an upper bound on the overall velocity of obstacle points is required. If passive friendly safety is desired, then a lower bound on the braking power and an upper bound on the reaction time of obstacles are also needed.

Many navigation algorithms have been proposed for autonomous mobile robots. Few of these algorithms, however, have been verified to ensure the safety of the robot. One consequence of this situation is that potentially superior but uncertified navigation algorithms are not deployed in safety-critical applications. The Simplex architecture allows these uncertified algorithms, which in Simplex terms are called *advanced controllers* (ACs), to be used along side a pre-certified controller, called the *baseline controller* (BC). The Simplex architecture contains a *decision module* that periodically executes *switching logic* that determines whether to leave the AC in control for another time step or switch control to the BC.

To obtain the switching logic, we identify a set of conservative constraints on sensor readings whose satisfaction at time t guarantees that, regardless of what the AC does between time t and time $t+\Delta t$, the specified safety property holds during that period, and the BC will be able to ensure that it continues to hold after that, if it is given control at time $t+\Delta t$. The *decision period* Δt is the period with which the decision module makes the switching decision. The constraints are obtained under assumptions about minimum internal angle and minimum edge length of polyhedral obstacles as well as the limits on the maximum velocity, maximum reaction time and minimum braking power of obstacles for passive safety and passive friendly safety. Our simulation results illustrate how our design avoids collisions.

Another distinguishing feature of our work is the manner in which the switching condition is computed, using extensive geometric reasoning. Existing approaches to computation of switching condition are based on Lyapunov stability theory (e.g., [18,19]) or, more recently, state-space exploration (e.g., [3]). These existing approaches cannot be applied to the problem at hand, because of the incomplete knowledge of the shapes and locations of the obstacles in the robot's environment. To the best of our knowledge, our study is the first to provide runtime assurance that a mobile robot with limited sensing can safely navigate in such an environment.

A preliminary version of this paper appeared in [14]. This is the full version and has the following significant differences.

- In Section 4.1, we extend [14] to take into account the limits on how rapidly the robot can accelerate and turn. We describe how to use this information to calculate a more accurate region that the robot can reach within Δt time units. This results in a tighter switching condition, allowing the robot to go past obstacles that would make the robot stop under the switching condition in [14]. In Section 5, we illustrate this difference via simulation results involving a ground-rover case study.
- In [14], we assume that the robot can stop instantaneously. In Section 4.2, we revise the switching logic to take the robot's finite braking power into account. This makes the approach more realistic.
- In addition to the notion of safety with respect to stationary obstacles considered in [14], we now also investigate the switching conditions needed to ensure safety with respect to moving obstacles.

The rest of the paper is organized as follows. Section 2 considers related work on provable collision avoidance. Section 3 provides background on the Simplex architecture. Section 4 contains a detailed derivation of the switching condition. Section 5 discusses our implementation and experimental results. Section 6 offers our concluding remarks and directions for future work.

2 Related Work

Collision avoidance is a well-studied problem. We refer the reader to a recent survey [8] for an extensive review of the literature on collision avoidance for both static and moving obstacles. In this section, we consider related work on provable collision avoidance, i.e., algorithms whose collision-freedom property is formally proven. Prior work [1,4,12,13] has focused on establishing collision-freedom for specific navigation algorithms. In contrast, we employ the Simplex architecture to ensure the safety of the robot in the presence of any navigation algorithm, however faulty it may be. We consider each of these approaches in turn.

Theorem-proving techniques are used in [12] to establish two safety properties of the Dynamic Window algorithm for collision avoidance: passive safety and passive friendly safety, both of which apply to stationary and moving obstacles. Infinite sensor detection range and 360° sensing are assumed. Our approach, in contrast, accounts for blind spots between sensors and limited sensing range. Our assumptions about the movement of the obstacles for ensuring passive safety and passive friendly safety are similar to [12].

In [4], the authors present the PassAvoid navigation algorithm, which avoids "braking-inevitable collision states" to achieve passive safety. In [17], a biologically inspired navigation algorithm for a unicycle-like robot moving in a dynamic environment is presented. Both algorithms assume 360° sensing capability. We do not make this assumption, and instead rely on certain weak assumptions about the shapes of obstacles.

In [1], the authors propose an algorithm that constrains the velocity of a mobile robot moving on a known trajectory such that it stops before colliding with moving obstacles. They assume 360° field-of-view and a pre-planned trajectory that guides the robot through an environment with known static obstacles. We do not make any of these assumptions.

A method is presented in [13] for computing a smooth, collision-free path from a piecewise linear collision-free trajectory produced by sampling-based planners. They assume the given sampling-based trajectory is collision-free and use cubic B-splines to generate a smooth trajectory that guarantees collision-freedom. We do not make any assumptions about robot trajectories.

A reachable-set approach is presented in [20] to assure that a robot safely visits a sequence of targets while avoiding moving obstacles. The robot and obstacles are treated as points. The initial locations, dynamics, and the set of control inputs of the obstacles are assumed known. The robot pre-computes a time-optimal path that remains unchanged during the mission. As such, this approach does not need to consider sensor readings and the computed path is necessarily conservative.

3 The Simplex Architecture

The Simplex architecture [18,19] was developed to allow sophisticated control software to be used in safety-critical systems. This sophisticated software, called an *advanced controller*, is designed to achieve high performance according to specified metrics (e.g., maneuverability, fuel economy, mission completion time). As a result, it might be so complex that it is difficult to achieve the desired level of safety assurance in all possible scenarios. Its complexity might also prevent it from achieving required certifications (e.g., RTCA DO-178C for flightworthiness). The Simplex architecture allows such advanced controllers to be used safely, by pairing them with a simpler *baseline controller* for which the desired level of safety assurance can be achieved, and with a *decision module* that determines which controller is in control of the plant.

While the system is under the control of the advanced controller, the decision module monitors the system state and periodically checks whether the system is in imminent danger of violating a given safety requirement. If so, the decision module switches control of the system from the advanced controller to the baseline controller. The period with which the decision module makes the switching decision is called the *decision period* and denoted Δt. The condition on the system state that it evaluates to determine whether to switch to the baseline controller is called the *switching condition*. The switching condition depends on the safety requirements, the system dynamics, and the decision period. A state is *correct* if it satisfies the given safety requirements. A state is *recoverable* if, starting from that state, the baseline controller can ensure that the system remains correct; i.e., remains in correct states.

The correctness requirement for the switching condition is: If the switching condition is false (i.e., "don't switch"), then the system is guaranteed to remain in recoverable states for the next Δt time units, regardless of the control inputs to the plant produced by the advanced controller during that interval. The quantification over all possible control inputs to the plant is needed because we make no assumptions about the advanced controller's behavior. If the baseline controller and switching condition are correct, then correctness of the system is ensured, regardless of the advanced controller's behavior.

4 Switching Logic

Our approach uses the Simplex architecture with a baseline controller that immediately applies full braking power to stop the robot. To simplify the derivation of the switching condition slightly, we make the following assumptions: (1) the execution time of the decision module is negligible; (2) the switching latency is negligible (i.e., the baseline controller can take over immediately); (3) the robot's shape is a single point, as in [12]. The robot point is the center of motion of the robot, which is the midpoint between the left and right driving wheels in the case of robots with two-wheel differential-drive. None of these assumptions are essential. Our derivation can easily be extended to eliminate them: (1) and (2) can be eliminated by simply adding DM's execution time and switching latency to Δt; (3) can be eliminated, for instance, by using the approach given in [11] to take the shape of the rover into account.

We assume that the robot starts in a state in which the switching condition is false. At time t, if DM lets the advanced controller remain in control for Δt time, it must ensure that the robot will not collide with an obstacle during this period of time, and that if the decision module switches to the baseline controller at the next time step, the baseline controller will be able to bring the robot to a full stop without any collisions. Based on this reasoning, our switching logic comprises the following steps:

1. Determine the set of positions that the robot can reach during the next time step. We call this set the *reachable region*.
2. Expand the reachable region by some margin to account for braking distance and moving obstacles. We call the expanded reachable region the *safety region*, denoted S_{safe}.
3. Find the set of possible obstacle points that are closest to the robot, based on the current sensor readings. We denote this set $S_{obstacle}$.
4. The switching condition is then a check of whether the safety region contains any possible obstacle points, i.e., whether $S_{safe} \cap S_{obstacle} \neq \emptyset$.

Steps 1 and 2 can be carried out statically. Steps 3 and 4 need to be performed at runtime. Note that steps 1 and 2 can be done at runtime provided the time it takes to finish these tasks is acceptable and does not exceed the decision period if we take DM's execution time into account. In steps 1, 2, and 3, the regions are over-approximations of the points in question. In the following, we describe these steps in more detail. In doing so, we show that the more we know about the robot, the more accurate and complex the switching condition is. Additionally, if we do not know anything about movement of obstacles, then we can only guarantee collision-freedom w.r.t. static obstacles.

If we have an upper bound on the overall velocity, i.e., taking into account both translational and rotational motions, of every obstacle point, then we can additionally guarantee passive safety w.r.t. moving obstacles. This means that the robot will be at rest if and when a collision occurs. If we also have a lower bound on the braking power and an upper bound on the reaction time of the obstacles, then we can offer the stronger passive friendly safety for moving obstacles. This means that when the robot stops, it leaves sufficient room for obstacles to avoid colliding with the robot. This hierarchical approach to defining safety allows one to design a safety guarantee based on how much information about the environment is available and to construct a suitable switching logic to ensure that safety property.

4.1 Step 1: Determine the Reachable Region

Depending on how much knowledge we have about the robot, we can determine the reachable region to different levels of accuracy. We consider two cases: (1) we know only the upper bound on the linear velocity of the robot; (2) we know the upper bounds on the linear velocity, linear acceleration, angular velocity, and angular acceleration of the robot.

4.1.1 Case 1

If we do not assume any limits on how rapidly the robot can turn or accelerate, the robot may immediately move in any direction at its maximum speed v_{max}. Velocity bound v_{max} and the decision period Δt define the robot's reachable region, a circular disk with radius $R = v_{max}\Delta t$ centered at the robot. While this reachable region is conservative, its simple shape allows us to derive a simple switching condition, as discussed in Section 4.4.

4.1.2 Case 2

If we have upper bounds on the linear velocity, linear acceleration, angular velocity and angular acceleration, then we can compute the reachable region from the current state and the kinematics of the robot. Eq. 1 provides general kinematics for mobile robots.

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{v} &= a \\ \dot{\theta} &= \omega \\ \dot{\omega} &= \gamma \end{cases} \tag{1}$$

where x and y are the Cartesian coordinates of the robot position, θ is the heading angle, v and a are the linear velocity and linear acceleration, and ω and γ are the angular velocity and angular acceleration of the robot. In Eq. 1, a and γ are implicitly functions of time. We want to find the reachable states based on Eq. 1 augmented with the constraints $\|v\| \leq v_{max}$, $\|a\| \leq a_{max}$, and $\|\gamma\| \leq \gamma_{max}$. Thus, a non-trivial region is reachable even from a single initial state.

Reachability analysis tools can use Eq. 1 to find the region reachable within Δt time units from a given initial state or range of initial states. We use HyCreate [2] for this purpose. HyCreate outputs the reachable region as a set of overlapping rectangles. We use Matlab to merge these rectangles into an equivalent polygon. Note that Eq. 1 does not have any constraints that relate v and ω. Many robots, however, have such constraints. For example, two-wheel differential-drive robots, like the one used in our study, can turn only by making one wheel rotate slower than the other. That means the bound on linear velocity is smaller than v_{max} while the robot is turning. The above computation ignores this constraint and hence produces an over-approximation of the reachable region. To capture this constraint, consider the kinematics of the two-wheel differential drive robot used in our study. The equations are:

$$\begin{cases} \dot{x} &= \frac{(\omega_l+\omega_r)r}{2}\cos\theta \\ \dot{y} &= \frac{(\omega_l+\omega_r)r}{2}\sin\theta \\ \dot{\theta} &= \frac{(\omega_r-\omega_l)r}{l} \\ \dot{\omega_l} &= \gamma_l \\ \dot{\omega_r} &= \gamma_r \end{cases} \tag{2}$$

where x and y are the Cartesian coordinates of the robot position, θ is the heading angle, ω_l and γ_l are the angular velocity and angular acceleration of the left wheel, ω_r and γ_r are the angular velocity and angular acceleration of the right wheel, r is the radius of the wheels, and l is the distance between the centers of the two wheels.

Compared to Eq. 1, linear velocity v and angular velocity ω of the robot are coupled because they are both expressed in terms of ω_l and ω_r. This eliminates infeasible combinations of values of v and ω and yields a more accurate reachable region. Note that we only use this specific model of differential-drive robots to get an accurate reachable region. We are not forced to use Eq. 2 for other purposes such as designing a baseline controller that can do more than just stop the robot. For those purposes, we can use Eq. 1 or any suitable kinematic models.

To pre-compute the reachable regions using HyCreate, we partition the state space into regions and compute a reachable region for each of these regions. We use initial values of $(0, 0, 0)$ for (x, y, θ). We do not need to consider different initial values for (x, y, θ) because the set of possible obstacle points $S_{obstacle}$ is computed in the robot's frame of reference. We partition the range of ω_l and ω_r into small uniform intervals. The Cartesian product of these intervals induces a grid over the state space. We use HyCreate to find the reachable region for each grid block based on Eq. 2 augmented with the constraints $\|\omega_l\| \leq \omega_{max}$, $\|\omega_r\| \leq \omega_{max}$, $\|\gamma_l\| \leq \gamma_{max}$, and $\|\gamma_r\| \leq \gamma_{max}$. When evaluating switching condition at runtime, we determine the region (grid block) corresponding to the current values of ω_l and ω_r and then look up the associated pre-computed reachable region.

4.2 Step 2: Expand the Reachable Region

Depending on what assumptions we have about the robot and the environment (obstacles), we can obtain different safety regions and offer different levels of safety. We prove in Appendix A that the safety regions discussed below are sound.

4.2.1 Braking Power

Let b be the braking power of the robot, i.e., the maximum magnitude of deceleration. Let v^{RR}_{max} be the maximum velocity reachable in Δt time units. If we do not assume a limit on how rapidly the robot can accelerate then $v^{RR}_{max} = v_{max}$. If we assume a maximum acceleration a_{max} then from a range of initial values $[v_{0min}, v_{0max}]$ of velocity, the robot can achieve a maximum velocity $v^{RR}_{max} = min\,(v_{max}, v_{0max} + a_{max}\Delta t)$ after Δt time units. In our case, because we use

Eq. 2 to compute the reachable region, v_{max}^{RR} must be inferred from ω_{max}, γ_{max} and the ranges of initial values of ω_l and ω_r. Suppose the ranges of initial values of ω_l and ω_r used for computing the reachable region are $[\omega_{l0min}, \omega_{l0max}]$ and $[\omega_{r0min}, \omega_{r0max}]$, respectively. Eq. 3 shows how we compute v_{0max}, a_{max}, v_{max} and v_{max}^{RR}. Note that we assume the wheels can freely move backward or forward.

$$\begin{cases} v_{0max} &= max\left(\left|\omega_{l0min}+\omega_{r0min}\right|, \left|\omega_{l0max}+\omega_{r0max}\right|\right) r/2 \\ a_{max} &= \gamma_{max} r \\ v_{max} &= \omega_{max} r \\ v_{max}^{RR} &= min\left(v_{max}, v_{0max}+a_{max}\Delta t\right) \end{cases} \tag{3}$$

The worst case is when the robot starts braking from v_{max}^{RR}. In that case, the braking time is $t_b = v_{max}^{RR}/b$ and the braking distance is $d_b = \left(v_{max}^{RR}\right)^2/2b$. That means that if the robot applies braking power b, it will be able to come to a full stop within distance d_b. The reachable region must be expanded by d_b in all directions to take braking distance into account. If the reachable region is circular, we expand it simply by increasing the radius by d_b. If the reachable region is a polygon, we use Matlab to expand it.

4.2.2 Stationary Obstacles

If we assume all obstacles are stationary, then the safety region is the expanded reachable region obtained above. We use S_{SS} to denote the safety region for static safety w.r.t. stationary obstacles.

4.2.3 Passive Safety for Moving Obstacles

Passive safety ensures collision-freedom while the robot is moving, which means the robot must be at rest before a collision occurs. While this is a weak safety guarantee, it helps reduce the effects of a collision, such as the force of a head-on collision. If we assume a known upper bound V on the overall velocity of every obstacle point, then in the worst-case, the closest point of the obstacle approaches the robot in a straight line at maximum velocity V. Within $\Delta t + t_b$ time units, the maximum distance that the obstacle can travel is $d_o = V(\Delta t + t_b)$. The reachable and braking region must be expanded by d_o so that the robot can stop before a collision occurs. This expanded region, denoted by S_{PS}, is the safety region for ensuring passive safety in the presence of moving obstacles.

4.2.4 Passive Friendly Safety for Moving Obstacles

Passive safety may create a situation in which the robot stops at a position that leaves the obstacle no choice but to collide with it. *Passive friendly safety* requires that the robot leaves enough room for the obstacle to avoid the collision. Informally, if a collision occurs after the robot stops, the obstacle is to blame. To ensure passive friendly safety, we further assume a known lower bound b_o on the braking power and a known upper bound τ on the reaction time of obstacles. In the worst-case, the closest obstacle approaches the robot on a straight line at maximum speed V during Δt time units; it continues at that speed during the braking time v_{max}^{RR}/b of the robot; after the robot has stopped, it then starts braking with braking power b_o after its maximum reaction time τ has passed. The worst-case braking distance of the obstacle is then $d_{bo} = V^2/2b_o$. The safety region for passive safety must be expanded by $V\tau + d_{bo}$ to account for reaction time and braking distance. We use S_{PFS} to denote the safety region for ensuring passive friendly safety.

Fig. 1 illustrates the relationship between the three safety regions S_{SS}, S_{PS}, and S_{PFS}.

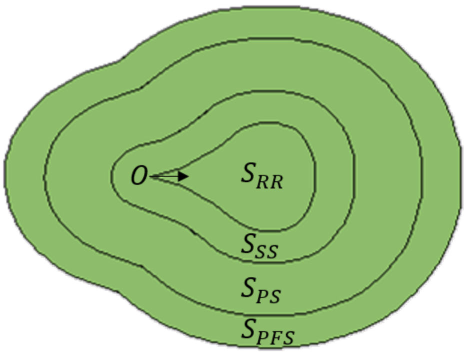


Fig. 1 Relationship between safety regions. The robot currently at O and heading in the direction of the arrow. S_{RR} is the region reachable within Δt time units. S_{SS}, S_{PS}, S_{PFS} are safety regions for ensuring static safety, passive safety, and passive friendly safety, respectively

4.3 Step 3: Find the set of possible obstacle points

The robot is equipped with N distance sensors with angle of detection β_s and maximum range R_s, as shown in Fig. 2. For simplicity, we assume the sensors are evenly spaced; it is easy to analyze other spacings in a similar way. The angle (in radians) of the gap between the fields-of-view of adjacent sensors is $\beta_g = (2\pi - N\beta_s)/N$. We assume N and β_s are such that $\beta_g > 0$ and $\beta = \beta_g + 2\beta_s \leq \pi/3$.

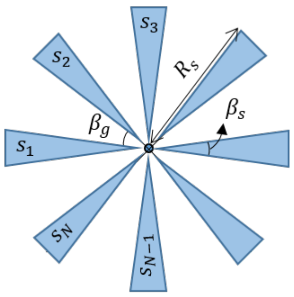


Fig. 2 The robot has N evenly spaced sensors $s_1, s_2, ..., s_N$ with angle of detection β_s and maximum range R_s. The angle of the gap between two adjacent sensors is β_g

When an obstacle intersects a sensor's cone of observation at multiple distances, depending on the exact nature of the sensor, it may report the closest distance to the obstacle, the farthest distance, or something in between. Our derivation of the switching condition is based on the worst-case (from the perspective of collision avoidance) assumption about sensor behavior, namely, that the sensor reports the farthest distance from the obstacle.

We derive the set of possible obstacle points closest to the robot from sensor readings under the following assumptions about obstacles: (1) obstacles are polyhedra; (2) there is a known lower bound α on the internal angles between edges and $\alpha > \beta$, where β is the angle between two adjacent sensors including the sensors' detection angles (i.e., $\beta = \beta_g + 2\beta_s$); (3) there is a known lower bound l_{min} on obstacle edge lengths and $l_{min} \geq L$, where L is defined below; and (4) the separation between obstacles is such that whenever two adjacent sensors detect an obstacle, they are detecting the same obstacle. Intuitively, the lower bound on internal angles ensures that vertices of obstacles are wide enough so that they will be detected by the robot's sensors despite blind spots.

Denote $E^{\alpha}_{AB} = \{P \mid \angle APB = \alpha\}$ the α-*equiangular arcs* of AB, i.e., the locus of points that see the line segment AB under angle α. Geometrically, E^{α}_{AB} forms two circular arcs that pass through A and B, shown as the boundary of the shape in Fig. 3. Let S^{α}_{AB} be the set of points that lie within the area enclosed by α-equiangular arcs of AB including the boundary. It is easy to show that $S^{\alpha}_{AB} = \{C \mid \angle ACB \geq \alpha\}$, which means S^{α}_{AB} is the locus of all possible vertices with angle at least α such that one edge passes through A and the other edge passes through B.

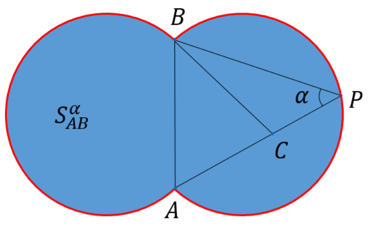


Fig. 3 Illustration of S^{α}_{AB}. The α-equiangular arcs of AB is the boundary

For sensor s_i, $i = 1..N$, we define two points A^-_i and A^+_i on the left and right edges of s_i's cone of detection such that $OA^-_i = OA^+_i = l_{min}$ if s_i does not detect any obstacle, and $OA^-_i = OA^+_i = \min\{OB_i, l_{min}\}$ if s_i detects an obstacle at B_i. Consider a sensor $s_{i'}$, where $i' = (i \bmod N) + 1$, that is adjacent to s_i. We define $A_i = A^-_i$ and $A_{i'} = A^+_{i'}$. That means A_i is on the left edge of s_i's cone while $A_{i'}$ is on the right edge of $s_{i'}$'s cone. The definition of A_i and $A_{i'}$ in conjunction with the assumption $\beta \leq \pi/3$ implies there is at most one obstacle vertex inside triangle $OA_iA_{i'}$. Because of the assumption that minimum internal angle is α, $S^{\alpha}_{A_iA_{i'}}$ contains all possible obstacle points between s_i and $s_{i'}$ and are closest to the robot. However, $S^{\alpha}_{A_iA_{i'}}$ is the union of two identical circular disks that mirror each other over the line $A_iA_{i'}$. As such, $S^{\alpha}_{A_iA_{i'}}$ also contains possible obstacle points that are not of our interest, that is the points farther away from the robot than $min(OA_i, OA_{i'})$. To be less conservative, we only consider the circular disk that is closer to the robot, denoted $C^{\alpha}_{A_iA_{i'}}$. Algorithm 1 finds the center and radius of $C^{\alpha}_{A_iA_{i'}}$. The union of $C^{\alpha}_{A_iA_{i'}}$ for all pair of adjacent sensors $(s_i, s_{i'})$ gives us the set of all possible obstacle points closest to the robot.

```
Input: $OA_i$, $OA_{i'}$, $\alpha$, $\angle A_iOA_{i'}$
// Distance between points $A_i$ and $A_{i'}$
$A_iA_{i'} = \sqrt{OA_i^2 + OA_{i'}^2 - 2 \cdot OA_i \cdot OA_{i'} \cdot \cos \angle A_iOA_{i'}}$;
// Radius of the $\alpha$-equiangular arcs for $A_iA_{i'}$, i.e., points $P$ such that
   $\angle A_iPA_{i'} = \alpha$
$R_{arc} = (A_iA_{i'}/2)/\sin\alpha$;
// Find the centers of those two arcs. Their position is defined by the
   following geometric constraints, whose solution amounts to finding the
   third vertex of a triangle, given the other two vertices (namely, $A_i$ and
   $A_{i'}$) and the internal angle at the third vertex $\angle A_iOA_{i'}$.
$O_{arc,1}$, $O_{arc,2}$ = the points $O_{arc}$ satisfying $O_{arc}A_i = O_{arc}A_{i'} \wedge \angle A_iO_{arc}A_{i'} = 2\alpha$;
// Between those two points, choose the one corresponding to the arc that is
   closer to the robot.
$O_{arc} = \alpha \leq \pi/2$ ? $\min\{OO_{arc,1}, OO_{arc,2}\}$ : $\max\{OO_{arc,1}, OO_{arc,2}\}$;
```

Algorithm 1: Computation of the center and radius of the α-equiangular arc of $A_iA_{i'}$ that is closer to the robot

4.4 Step 4: Check if the safety region contains any possible obstacle points

Let S_{safe} be the safety region for ensuring the safety property of interest. S_{safe} could be S_{SS}, S_{PS}, or S_{PFS} as described in Section 4.2. Let $S^{ii'}_{safe}$ be the set of points in the safety region and in or between the cones of observation of sensors s_i and $s_{i'}$, as illustrated in Fig. 4.

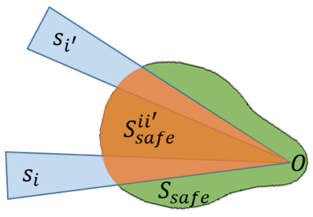


Fig. 4 Illustration of $S^{ii'}_{safe}$, the set of points in the safety region and in or between the cones of observation of sensors s_i and $s_{i'}$

The switching condition is simply a check of whether the safety region contains any possible obstacle points. We prove in Appendix A that the constraints $S^{\alpha}_{A_iA_{i'}} \cap S^{ii'}_{safe} = \emptyset$ for $i = 1..N$, where $i' = (i \bmod N) + 1$, imply $S_{obstacle} \cap S_{safe} = \emptyset$ and hence guarantee that after Δt time units, the robot will be in a state where it can still be able to ensure the safety property. For the reason described in Section 4.3, we use $C^{\alpha}_{A_iA_{i'}}$ instead of $S^{\alpha}_{A_iA_{i'}}$ to derive the switching condition.

4.4.1 Circular safety region

If we make no assumptions about how rapidly the robot can turn or accelerate, then the safety region S_{safe} will be a circular disk with radius $R = v_{max}\Delta t + v_{max}/2b$ centered at the robot. We can use an algorithm like Algorithm 2 to check if $C^{\alpha}_{A_iA_{i'}} \cap S^{ii'}_{safe} \neq \emptyset$. However we can obtain a computationally cheaper algorithm by checking the more conservative alternative $C^{\alpha}_{A_iA_{i'}} \cap S_{safe} \neq \emptyset$. The switching condition is then a check of whether two circular disks—S_{safe} and $C^{\alpha}_{A_iA_{i'}}$— overlap. This check can be done simply by checking if the sum of two radii is less than the distance between two centers. We went further and found that in the case when only one of s_i and $s_{i'}$ detects an obstacle within l_{min}, we can derive an even simpler check of whether the distance to the obstacle is less than a constant. We use the following property to derive the switching condition in this case.

Let OX, OY be two readings by sensor s_i such that $OX < OY$. Let OZ be the reading of sensor $s_{i'}$ that is adjacent to s_i.

Property 1 $\left|S^{\alpha}_{XZ} \cap S^{ii'}_{safe}\right| = 1 \rightarrow S^{\alpha}_{YZ} \cap S^{ii'}_{safe} = \emptyset$

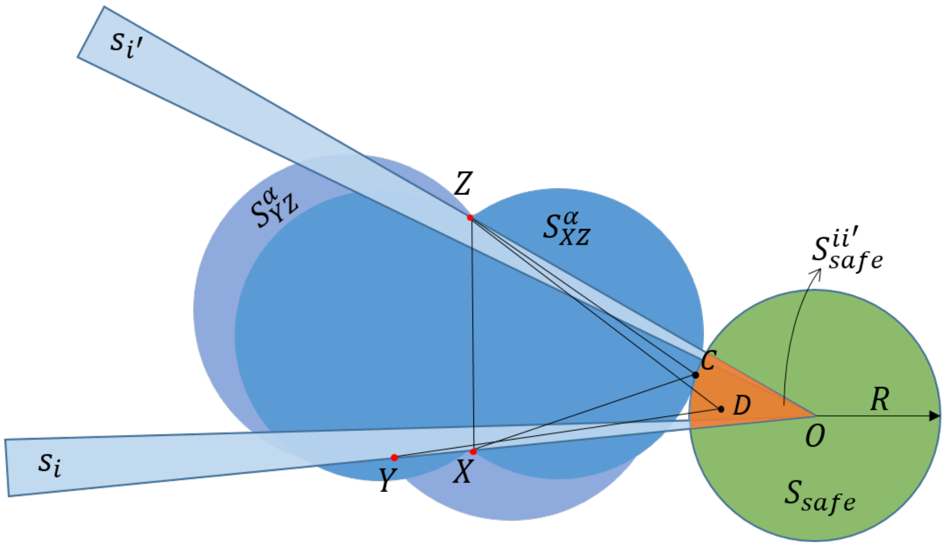


Fig. 5 Illustration of Property 1. S^{α}_{XZ} touches $S^{ii'}_{safe}$ at C. $S^{\alpha}_{YZ} \cap S^{ii'}_{safe} = \emptyset$

Proof By contradiction. Suppose $\left|S^{\alpha}_{XZ} \cap S^{ii'}_{safe}\right| = 1$ and $S^{\alpha}_{YZ} \cap S^{ii'}_{safe} \neq \emptyset$. Let $C \in S^{\alpha}_{XZ} \cap S^{ii'}_{safe}$ as shown in Fig. 5 (C is the point where S^{α}_{XZ} touches $S^{ii'}_{safe}$). Since C lies on the boundary of S^{α}_{XZ}, we have $\angle XCZ = \alpha$. Let $D \in S^{\alpha}_{YZ} \cap S^{ii'}_{safe}$. Because

OY is strictly greater than OX, the geometry implies $\angle XDZ > \angle YDZ \geq \alpha$. This means $D \in S^{\alpha}_{XZ}$ and $D \not\equiv C$, therefore $\left| S^{\alpha}_{XZ} \cap S^{ii'}_{safe} \right| > 1$, a contradiction.

Suppose sensor s_1 detects an obstacle at point A_1, where $OA_1 = d_1$, and adjacent sensors do not detect any obstacle within distance l_{min}, as shown in Fig. 6. In this case, we assume the adjacent sensor s_2 detects an obstacle at distance $OA_2 = l_{min}$, as described above. The switching condition $\phi_1(s_1)$ in this case is of the form $d_1 \leq d_{1switch}$, for the threshold $d_{1switch}$ defined below.

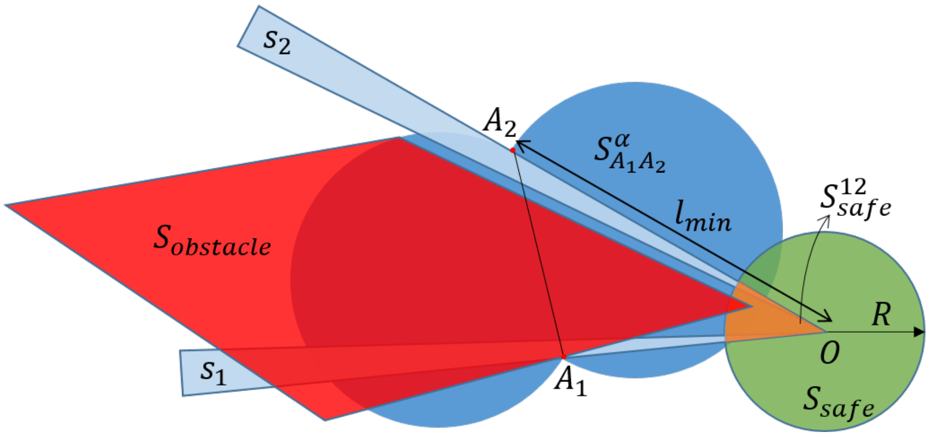


Fig. 6 Illustration of case 1. Sensor s_1 detects an obstacle at distance $OA_1 < l_{min}$. Adjacent sensor s_2 does not detect any obstacle within distance l_{min} so we assume $OA_2 = l_{min}$

If we can find a point A_T such that $\left| S^{\alpha}_{A_T A_2} \cap S_{safe} \right| = 1$ (i.e., $S^{\alpha}_{A_T A_2}$ touches S_{safe}), then by Property 1, we can let $d_{1switch} = OA_T$. This switching condition is more conservative than the constraint $S^{\alpha}_{A_1 A_2} \cap S^{12}_{safe} = \emptyset$ because there are some cases when $S^{\alpha}_{A_T A_2}$ touches S_{safe} at a point outside the wedge S^{12}_{safe}. The benefit is that the switching threshold $d_{1switch} = OA_T$ can be computed statically, resulting in a very simple switching condition.

The point A_T must satisfy the following equations, where O_{arc} is the center of the α-equiangular arc of $A_T A_2$ as shown in Fig. 7.

$$A_T A_2 = \sqrt{OA_T^2 + l_{min}^2 - 2 \cdot OA_T \cdot l_{min} \cdot \cos\beta} \tag{4}$$
$$R_{arc} = (A_T A_2/2)/\sin\alpha \tag{5}$$
$$OO_{arc} = R_{arc} + R \tag{6}$$

Given l_{min}, α, β and R, all of which are known statically, the switching threshold OA_T can be obtained by straightforward solution of these equations using algebraic geometry. We use Matlab to automate this.

4.4.2 Polygonal safety region

If the safety region is polygonal, then Algorithm 2 is used to check whether $C^{\alpha}_{A_i A_{i'}} \cap S^{ii'}_{safe} \neq \emptyset$. Note that we use the algorithm in [9], which is $O(n)$ where n is the

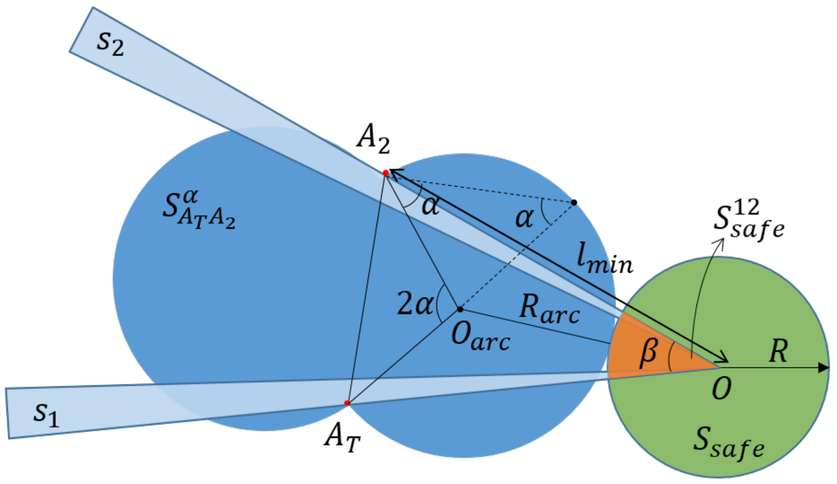


Fig. 7 Illustration of switching threshold OA_T calculation

number of vertices of the polygon, because the reachable regions outputted by HyCreate are not convex as shown in Fig. 11. If safety regions are convex, we can use a $O(\log n)$ algorithm (e.g., [15]) instead.

Algorithm 2 checks whether a circle overlaps with a polygon based on [5]. A circle C is defined by its center $C.O$ and its radius $C.R$. A polygon is defined by a list of vertices $[P_0, P_1, ..., P_N]$. A point P is defined by its x and y coordinates $P.x$ and $P.y$. A circle overlaps a polygon if one of the following condition is satisfied:

1. the shortest distance between any edge of the polygon and the circle's center is less than its radius;
2. the center of the circle is in the polygon.

4.4.3 Lower bounds on α and l_{min}

The assumption $\alpha > \beta$ is needed because if $\alpha \le \beta$, then $\angle A_i O A_{i'} = \beta \ge \alpha$, i.e., $O \in C^{\alpha}_{A_i A_{i'}}$ for any pair $A_i, A_{i'}$. That means $C^{\alpha}_{A_i A_{i'}}$ always intersects the safety region and we cannot guarantee the safety of the robot.

Fig. 8 shows the lower bound L on l_{min}. Let O_{arc} be the center of the α-equiangular arc of $A_{iL}A_{i'L}$ as shown in Fig. 8. Let R be the radius of the circle centered at the robot and enclosed the largest safety region. Similar to the computation of OA_T described above, L can be derived from the following equations.

$$A_{iL}A_{i'L} = \sqrt{2 \cdot L^2 - 2 \cdot L^2 \cdot \cos\beta} \tag{7}$$

$$R_{arc} = (A_{iL}A_{i'L}/2)/\sin\alpha \tag{8}$$

$$OO_{arc} = R_{arc} + R \tag{9}$$

The assumption $l_{min} \ge L$ ensures that if adjacent sensors s_i and $s_{i'}$ both detect an obstacle at distances greater than l_{min}, then no obstacle point appears within the wedge $S^{ii'}_{safe}$. We prove this in Appendix A (case 1c).

```
Input: Circle $C$, Polygon $Poly$
return $EdgeInCircle(C, Poly) \vee PointInPolygon(C.O, Poly)$;

// Check if there is an edge of the polygon whose shortest distance to the
   center of the circle is less than its radius.
Function EdgeInCircle(Circle $C$, Polygon $Poly$)
    foreach edge $P_iP_j$ of $Poly$ do
        // The following code computes the shortest distance $d$ between $C.O$
           and the edge $P_iP_j$, using the algorithm in [6].
        $t = \frac{(P_j-P_i)\cdot(C.O-P_i)}{||P_j-P_i||^2}$;
        if $t \le 0$ then
            $d = ||C.O - P_i||$
        else if $t \ge 1$ then
            $d = ||C.O - P_j||$
        else
            $d = ||C.O - (P_i + t \cdot (P_j - P_i))||$
        end
        if $d \le C.R$ then
            return True
        end
    end
    return False

// Use even-odd rule [9] to check if a point is inside a polygon. The idea
   is to draw a ray that starts from the point and goes in any direction,
   and count how many times it intersects the edges of the polygon. The
   number is odd when the point is inside the polygon.
Function PointInPolygon(Point $P$, Polygon $Poly$)
    isInside = false;
    foreach edge $P_iP_j$ of $Poly$ do
        if $((P_i.y > P.y) \neq (P_j.y > P.y)) \wedge (P.x < \frac{(P_j.x-P_i.x)\cdot(P.y-P_i.y)}{(P_j.y-P_i.y)} + P_i.x)$ then
            isInside = ¬isInside
        end
    end
    return isInside;
```

Algorithm 2: Checking whether a circle overlaps with a polygon.

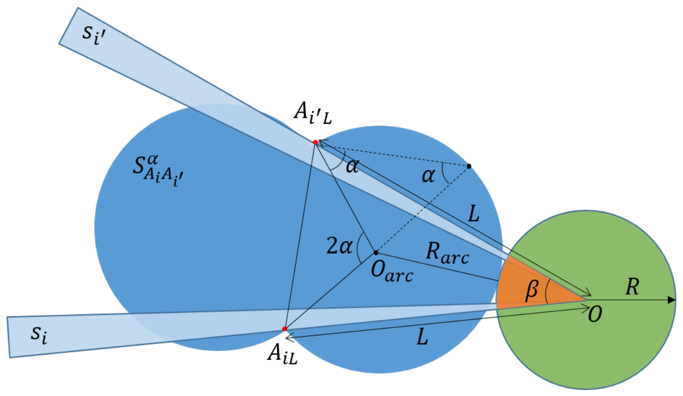


Fig. 8 Lower bound L on l_{min} such that the α-equiangular arcs of $A_{iL}A_{i'L}$ touch the circle that enclosed the safety region

5 Implementation and Experimental Results

We implemented the Simplex architecture with the baseline controller and switching conditions described in Section 4 in the Matlab simulator for the Quickbot ground robot [16]. The robot has sensor architecture as in Fig. 2 with the following parameters: (1) number of sensors $N = 8$; (2) angle of detection of the sensors $\beta_s = 5^o$; (3) maximum range of the sensors $R_s = 0.8m$; (4) maximum rotational speed of the wheels $\omega_{max} = 7\pi\ rad/s$; (5) maximum rotational acceleration of the wheels $\gamma_{max} = 16\pi\ rad/s^2$; (6) braking power $b = 30m/s^2$; (7) radius of the wheels $r = 0.0325m$; (8) distance between the centers of the two wheels $l = 0.09925m$; (9) maximum linear velocity $v_{max} = \omega_{max}r = 0.715m/s$; (10) maximum acceleration $a_{max} = \gamma_{max}r = 1.634m/s^2$, and decision period $\Delta t = 0.1s$.

5.1 Circular Safety Region

The radius of the circular safety region is $R = v_{max}\Delta t + v_{max}^2/2b = 0.08m$. We tested the switching condition in the following two scenarios; snapshots from simulations of these scenarios appear in Fig. 9. Both scenarios involve an obstacle with lower bound on internal angles $\alpha = 70^\circ$. For the scenario in Fig. 9(a), we place the obstacle such that when the robot approaches the obstacle and the vertex with angle α is about to enter the safety disk, only one sensor detects an edge with l_{min} and the other edge barely misses the cone of observation of an adjacent sensor. This is the worst-case scenario for the case when only one sensor detects an obstacle at distance less than l_{min} as described in Section 4.4. For the scenario in Fig. 9(b), we place the obstacle such that when the robot approaches the obstacle and the vertex with angle α is about to enter the safety disk, the vertex is in the gap of two adjacent sensors and both sensors detect an edge of the obstacle within l_{min}.

The snapshots in Fig. 9 show the moment when the switching condition becomes true and the robot stops. One observation is that, in both scenarios, the switching condition is *correct*: the obstacle does not enter the safety disk. Of course, this is expected. A more interesting observation is that, in both scenarios, the switching condition is *tight* (not unnecessarily conservative): the robot does not stop until the obstacle is about to enter the safety disk. The actual simulations leading to these snapshots can be viewed at `https://www.youtube.com/watch?v=iQpOZgYyhqQ`

Fig. 10 shows how the switching threshold OA_T depends on various parameters in the case when only one sensor detects an obstacle at distance less than l_{min} . Fig. 10(a) shows how OA_T decreases as α increases. It is clear from the worst-case scenario of case 1 that when an obstacle with a sharper corner, i.e., a smaller α, touches the safety disk, the sensor detects its edge at a greater distance than one with a flatter corner, and this necessitates a larger OA_T. Fig. 10(b) shows how OA_T increases as β increases. Intuitively, a larger β means a larger gap between the cones of observation of two adjacent sensors, so the edge of the obstacle is detected at a larger distance when the vertex is at the boundary of the safety disk. Fig. 10(c) shows how OA_T decreases as l_{min} increases. This can be seen from the worst-case scenario: the edge of the obstacle that is not detected within l_{min} will make a smaller angle with the edge of the cone if l_{min} is larger, so the other edge is detected at a smaller distance. Fig. 10(d) shows how OA_T increases as R increases

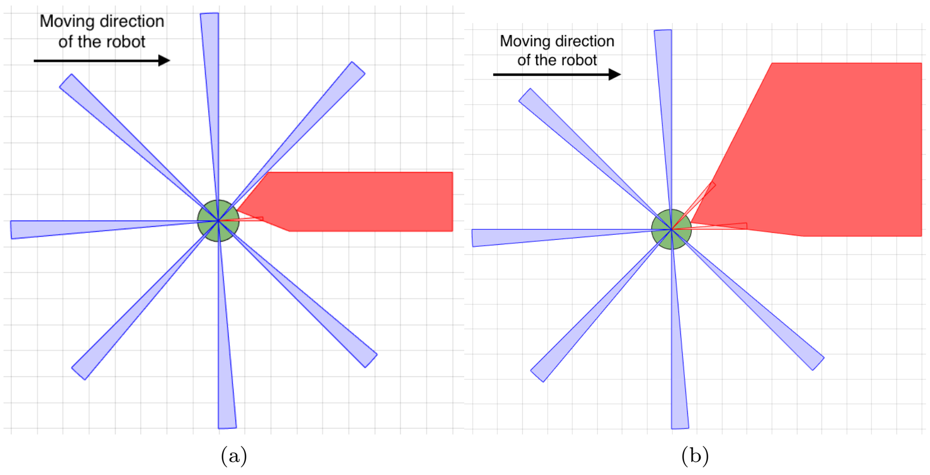


Fig. 9 Snapshots from simulations showing the robot correctly stops to ensure no obstacles in the safety disk. The green disk around the robot represents the safety disk. The red region represents the obstacle. The blue wedges represent the robot's cones of observation. (a) Snapshot from scenario for case 1: a sensor detects an obstacle within l_{min}; adjacent sensors do not. (b) Snapshot from scenario for case 2: two adjacent sensors detect an obstacle within l_{min}

(note: it doesn't matter whether the increase in R is due to an increase in v_{max} or Δt, or a decrease in b). This directly reflects the fact that a robot with a larger safety disk needs to stop farther from obstacles.

5.2 Polygonal Safety Region

Fig. 11 shows an example of the reachable region obtained from HyCreate and the three safety regions we get by expanding the reachable region. We compare the switching conditions of using circular safety region and polygonal safety region; snapshots from simulations of the comparison appear in Fig. 12. We consider a static obstacle with lower bound on internal angles $\alpha = 70°$. For the scenario of using circular safety region in Fig. 12(a), we place the obstacle such that when two sensors both detect an edge, the α-equiangular arc will intersect the safety disk so that the robot has to stop. Note that one sensor detects an edge at a distance greater than l_{min}, and we use l_{min} as the detected distance of the edge. The snapshot shows the moment when the switching condition becomes true and the robot stops. For the scenario of using polygon safety region in Fig. 12(b), the placement of the obstacle is the same as in the previous scenario. The snapshot shows the moment when the sensors detect the obstacle at the same distances as in Fig. 12(a). In this case, however, the α-equiangular arc does not intersect the polygonal safety disk so that the switching condition remains false and the robot keeps moving. The robot moves from the left to the right in this scene without stopping. The comparison shows that using polygonal safety region instead of circular safety region leads to a less conservative switching condition. The actual simulations leading to these snapshots can be viewed at `https://youtu.be/dZewtOPO_KI`.

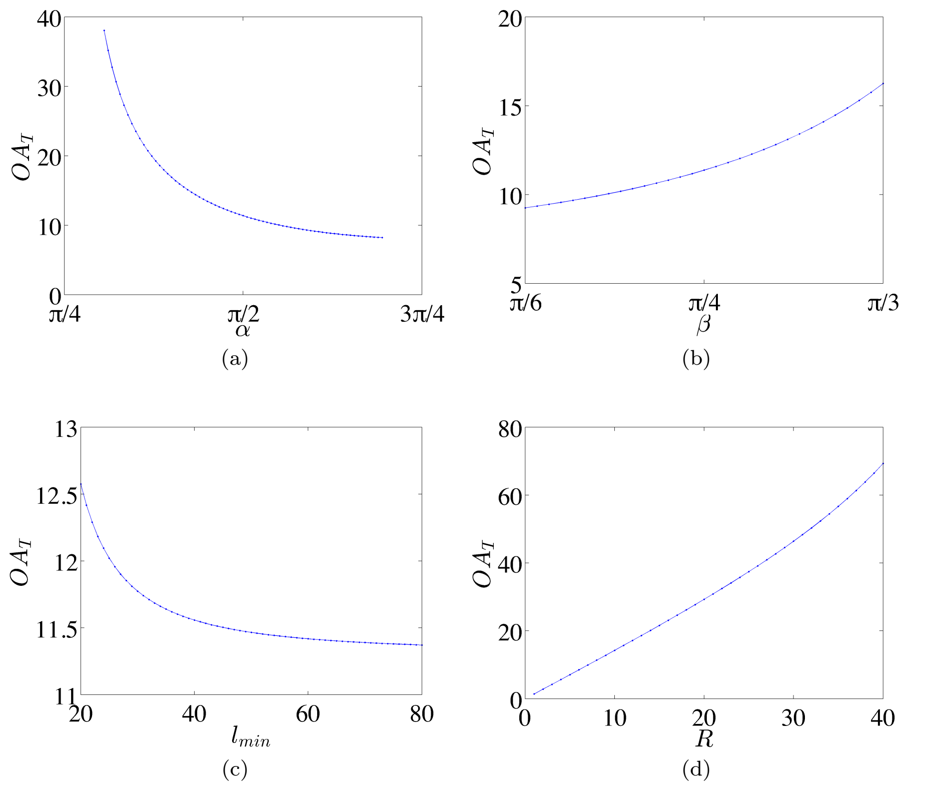


Fig. 10 Graphs of the switching threshold OA_T as a function of various parameters. (a) OA_T as a function of α, with $\beta = \pi/4$, $l_{min} = 80$ and $R = 8$. (b) OA_T as a function of β, with $\alpha = \pi/2$, $l_{min} = 80$ and $R = 8$. (c) OA_T as a function of l_{min}, with $\alpha = \pi/2$, $\beta = \pi/4$, and $R = 8$. (d) OA_T as a function of R, with $\alpha = \pi/2$, $\beta = \pi/4$, and $l_{min} = 80$

Additionally, a simulation of passive-safety for moving obstacles is available at `https://youtu.be/SLB1hME3Z10`.

6 Conclusions

In this paper, we have shown how it is possible to use the Simplex architecture, equipped with a sophisticated geometric-based switching condition, to ensure at runtime that mobile robots with limited field-of-view and limited sensing range navigate safely with only limited information about moving obstacles.

Future work includes extending our approach to take into account the size and shape of the robot and the minimum detection distance of the sensors. We will also consider more powerful baseline controllers. In particular, we plan to extend the Dynamic Window algorithm (DWA) [7] with our geometric analysis. DWA has been verified [12] to guarantee collision-freedom under the assumption of 360° sensing. Our geometric analysis would enable DWA to work with limited sensing.

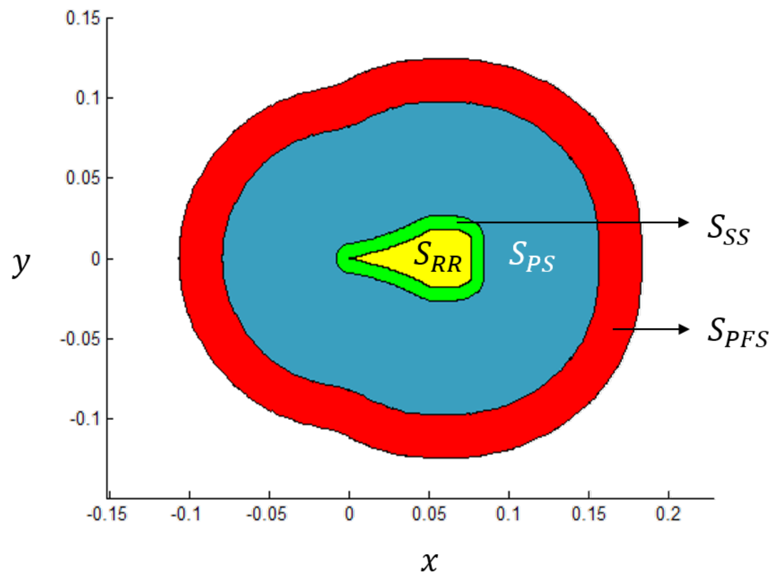


Fig. 11 Illustration of Quickbot's reachable region and different safety regions. The reachable and safety regions are computed with $\omega_{l0} = \omega_{r0} = [5\pi, 7\pi]$ rad/s, $\omega_{max} = 7\pi$ rad/s, $\gamma_{max} = 16\pi$ rad/s^2, $b = 30m/s^2$, $\Delta t = 0.1s$, $V = 0.715m/s$, $\tau = 0.02s$, $b_o = 20m/s^2$, where ω_{l0} and ω_{r0} are the range of initial values of ω_l and ω_r for computing the reachable region

We also plan to investigate an extension of the Simplex architecture that would allow control of the plant to be given back to AC after a failover to BC has occurred, thereby allowing AC's performance benefits to come back into play. Such a "reverse switching condition" would most likely need to take into account the following: (1) the AC-to-BC switching condition is false, and (2) the AC-to-BC switching condition will not be true anytime soon, to avoid excessive switching. Another direction of future work is to perform a theoretical analysis of the conservativeness of the over-approximations of the reachable region and the set of nearby obstacle points.

We also plan to develop algorithms that would allow the robot to learn about its environment, enabling it to replace worst-case assumptions with more detailed information about obstacles it has encountered; this in turn would allow tighter switching conditions. The geometric analysis that we developed to derive and verify the switching condition can also be used as a basis for the design of collision-avoidance logic in navigation algorithms for mobile robots.

Acknowledgements We thank Denise Ratasich for her helpful comments on earlier drafts of the manuscript. We also thank our anonymous reviewers for their comments that help to improve the manuscript. This material is based upon work supported in part by AFOSR Grant FA9550-14-1-0261, NSF Grants IIS-1447549, CNS-1421893, CNS-1446832, CCF-1414078, ONR Grant N00014-15-1-2208, and Artemis EMC2 Grant 3887039. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these organizations.

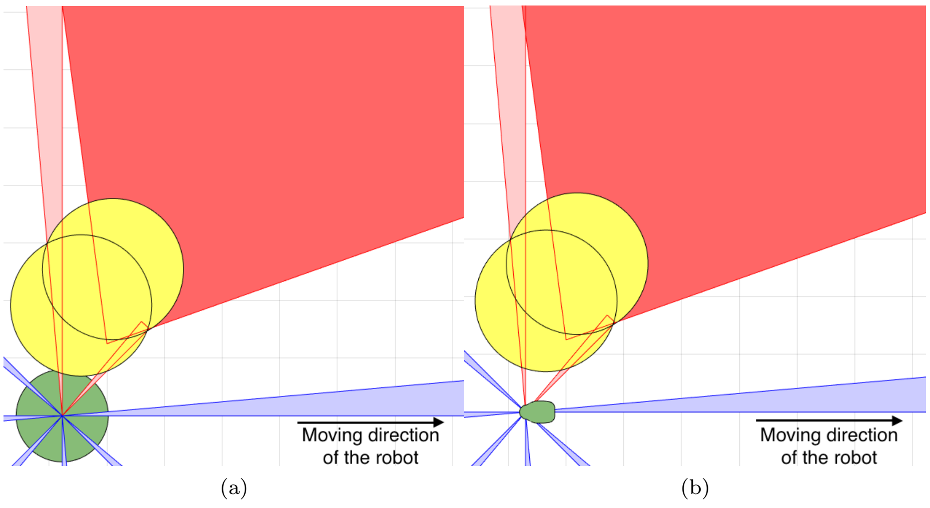


Fig. 12 Snapshots from simulations showing the comparison of using circular safety region vs. polygonal safety region. The red region represents the obstacle. The blue wedges represent the robot's cones of observation. The yellow circles represent the α-equiangular arcs. (a) Snapshot of using circular safety disk represented by the green disk. The robot stops at this moment as the α-equiangular arc intersects the safety region. (b) Snapshot of using polygonal safety disk represented by the green polygon. The robot does not stop at this moment as the α-equiangular arc does not intersect the safety region

References

1. Alami, R., Krishna, K.M.: Provably safe motions strategies for mobile robots in dynamic domains. In: in Autonomous Navigation in Dynamic Environment: Models and Algorithms. in C. Laugier, R. Chatila (Eds.), Springer Tracts in Advanced Robotics (2007)
2. Bak, S.: Hycreate: A tool for overapproximating reachability of hybrid automata (2013). URL `http://stanleybak.com/projects/hycreate/hycreate.html`
3. Bak, S., Manamcheri, K., Mitra, S., Caccamo, M.: Sandboxing controllers for cyber-physical systems. In: Proc. 2011 IEEE/ACM International Conference on Cyber-Physical Systems ICCPS, pp. 3–12. IEEE Computer Society (2011)
4. Bouraine, S., Fraichard, T., Salhi, H.: Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. Autonomous Robots **32**(3), 267–283 (2012). DOI 10.1007/s10514-011-9258-8. URL `https://hal.inria.fr/hal-00733913`
5. Chen, Y., Smith, T.R.: Finitely representable spatial objects and efficient computation. In: Algorithms and Computation, pp. 181–189. Springer (1994)
6. Eberly, D.: Distance between point and line, ray, or line segment. Geometric Tools (1999). URL `http://www.geometrictools.com/Documentation/DistancePointLine.pdf`
7. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics Automation Magazine **4**(1), 23–33 (1997). DOI 10.1109/100.580977
8. Hoy, M., Matveev, A.S., Savkin, A.V.: Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. Robotica **33**(3), 463–497 (2015). DOI 10.1017/S0263574714000289
9. Hughes, J.F., Van Dam, A., Foley, J.D., Feiner, S.K.: Computer graphics: principles and practice. Pearson Education (2013)
10. Macek, K., Vasquez Govea, D.A., Fraichard, T., Siegwart, R.Y.: Towards Safe Vehicle Navigation in Dynamic Urban Scenarios. Automatika (2009). URL `https://hal.inria.fr/inria-00447452`
11. Minguez, J., Montano, L., Santos-Victor, J.: Abstracting vehicle shape and kinematic constraints from obstacle avoidance methods. Autonomous Robots **20**(1), 43–59 (2006). DOI 10.1007/s10514-006-5363-5. URL `http://dx.doi.org/10.1007/s10514-006-5363-5`

12. Mitsch, S., Ghorbal, K., Platzer, A.: On provably safe obstacle avoidance for autonomous robotic ground vehicles. In: P. Newman, D. Fox, D. Hsu (eds.) Robotics: Science and Systems. Berlin, Germany (2013)
13. Pan, J., Zhang, L., Manocha, D.: Collision-free and smooth trajectory computation in cluttered environments. Int. J. Rob. Res. **31**(10), 1155–1175 (2012). DOI 10.1177/0278364912453186. URL `http://dx.doi.org/10.1177/0278364912453186`
14. Phan, D., Yang, J., Ratasich, D., Grosu, R., Smolka, S., Stoller, S.D.: Collision avoidance for mobile robots with limited sensing and limited information about the environment. In: Proc. 15th International Conference on Runtime Verification (RV 2015), Lecture Notes in Computer Science. Springer-Verlag (2015)
15. Preparatat, F., Shamos, M.: Computational geometry: An introduction. pp. 41–67. Springer-Verlag (1985)
16. QuickBot MOOC v2 (2014). URL `http://o-botics.org/robots/quickbot/mooc/v2/`
17. Savkin, A.V., Wang, C.: A reactive algorithm for safe navigation of a wheeled mobile robot among moving obstacles. In: Proceedings of the 2012 IEEE International Conference on Control Applications (CCA), pp. 1567–1571. IEEE (2012)
18. Seto, D., Krogh, B., Sha, L., Chutinan, A.: The Simplex architecture for safe online control system upgrades. In: Proc. 1998 American Control Conference, vol. 6, pp. 3504–3508 (1998). DOI 10.1109/ACC.1998.703255
19. Sha, L.: Using simplicity to control complexity. IEEE Software **18**(4), 20–28 (2001). DOI 10.1109/MS.2001.936213
20. Takei, R., Huang, H., Ding, J., Tomlin, C.J.: Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 323–329 (2012). DOI 10.1109/ICRA.2012.6225074

A Proof of Correctness of the Switching Conditions

We prove the following:

- P1: If no obstacle points are inside the safety region described in Section 4.2 then after Δt time units, the robot will still be able to guarantee the corresponding safety property. Specifically:
  - P1.1: Static safety property: after Δt time units, the robot will be able to brake to a full stop without colliding with any static obstacles.
  - P1.2: Passive safety property: after Δt time units, the robot will be able to brake to a full stop before any collisions with an obstacle can happen.
  - P1.3: Passive friendly safety: after Δt time units, the robot will be able to brake to a full stop before any collision with an obstacle can happen, and after the robot stops, any moving obstacle can brake to a full stop without colliding with the robot.
- P2: If the constraints $S^{\alpha}_{A_iA_{i'}} \cap S^{ii'}_{safe} = \emptyset$, for $i = 1..N$ where $i' = (i \bmod N) + 1$, hold, then there are no obstacle points inside the safety region S_{safe}. In other words, the switching conditions in Section 4.4 are sound.

We define $C_{P,r}$ as the set of points that are within Euclidean distance r from a point P, i.e., $C_{P,r} = \{Q \mid PQ \leq r\}$. In other words, $C_{P,r}$ is a circular disk of radius r centered at P. It is trivial to show that $Q \in C_{P,r} \rightarrow P \in C_{Q,r}$.

A.1 Proof of P1

P1 means that if no obstacle points are inside the safety region corresponding to the safety property of interest, then the robot does not need to begin braking now; it has enough room to start braking at the next time step if necessary to ensure the safety property.

In the following proofs, we assume that the computed reachable region, denoted S_{RR} is a conservative approximation to the actual reachable region. This is justified when we use safety disk with radius $v_{max}\Delta t$, whose correctness follows immediately from the definitions, and when we use the region computed by HyCreate, which uses a sound algorithm based on face lifting.

A.1.1 Proof of P1.1

As described in Section 4.2, the safety region S_{SS} corresponding to this safety property is obtained by expanding the reachable region S_{RR} by the worst-case braking distance $d_b = v_{max}/2b$. Since there are no obstacle points inside the reachable region, the robot never comes into contact with any obstacles during Δt time units. We prove by contradiction that, if the robot starts braking at the next time step, it will come to a full stop without colliding with any static obstacles.

By definition, the S_{SS} is obtained by expanding S_{RR} by d_b. That means $C_{A,d_b} \subseteq S_{SS} \quad \forall A \in S_{RR}$.

Suppose after Δt time units, the robot reaches point $A \in S_{RR}$ then starts braking at maximum braking power b and collides with an obstacle point $B \notin S_{SS}$, meaning B is on the braking trajectory of the robot. In the worst-case scenario, the robot starts braking from its maximum speed v_{max} and comes to a full stop after traveling a distance $d_b = v_{max}/2b$. That means all possible braking trajectories are contained inside C_{A,d_b}. Since obstacle point B is a point on the braking trajectory, we have $B \in C_{A,d_b}$. But $C_{A,d_b} \subset S_{SS}$, therefore $B \in S_{SS}$, a contradiction.

A.1.2 Proof of P1.2

By definition, the safety region S_{PS} for ensuring passive safety for moving obstacles is obtained by expanding S_{SS} by $d_o = V(\Delta t + t_b)$, where V is the maximum speed of obstacles, and t_b is the worst-case braking time of the robot. That means $C_{A,d_o} \subseteq S_{PS} \quad \forall A \in S_{SS}$.

We will prove that the robot can start to brake after Δt time units and then come to a full stop without colliding with any obstacles. Let T be the set of all trajectories on which the robot move during the next time step and then brake to a full stop. The definition of S_{SS} implies it contains T. Suppose there is a collision before the robot can come to a full stop, i.e., there is an obstacle point B outside S_{PS} that collides with the robot at point C after some time $t <= \Delta t + t_b$ while the robot is moving on some trajectory in T. Since maximum speed of obstacles is V, the maximum distance that B can travel in t time units is $Vt \leq d_o$. That means $C \in C_{B,d_o}$, which in turns means $B \in C_{C,d_o}$. Because $C \in S_{SS}$, we have $C_{C,d_o} \subseteq S_{PS}$. Therefore, $B \in S_{PS}$, a contradiction.

A.1.3 Proof of P1.3

By definition, the safety region S_{PFS} for ensuring passive friendly safety for moving obstacles is obtained by expanding S_{PS} by $d_{bo} = V\tau + V^2/2b_o$, where V is the maximum speed of obstacles, τ is the upper bound on the reaction time of the obstacles, and b_o is the lower bound on the braking power of the obstacles. That means $C_{A,d_{bo}} \subseteq S_{PFS} \quad \forall A \in S_{PS}$.

Since S_{PFS} is bigger than S_{PS}, it follows from the passive safety proof that the robot can come to a full stop before a collision can occur. We prove that after the robot comes to a full stop, the obstacle can brake to a full stop without colliding with the robot. Specifically, we show that applying any braking power greater than or equal to b_o brings the obstacle to a complete stop without colliding with the robot.

Suppose the robot stops at point $A \in S_{SS}$. For a collision to occur, A must belong to some trajectory of some obstacle point. Suppose there are no obstacle points inside S_{PFS} and after τ time units following the time when the robot stops, an obstacle starts braking but collides with the robot before the obstacle comes to a full stop. Let B denote the location of the obstacle point at the beginning of the current time step and later collides with the robot. From the assumptions, we have $B \notin S_{PFS}$. The worst-case time for the robot to move during the next time step time units and then brake to a full stop is $\Delta t + t_b$. During the time $\Delta t + t_b$ and the worst-case reaction time τ, the obstacle can travel a maximum distance of $d_o = V(\Delta t + t_b + \tau)$. The worst-case braking distance of the obstacle is $d_{bo} = V^2/2b_o$. That means for A to belong to a trajectory of the obstacle point starting at B, A must be in $C_{B,d_o+d_{bo}}$. That in turns means $B \in C_{A,d_o+d_{bo}}$. But the definition of S_{PFS} implies $S_{A,d_o+d_{bo}} \in S_{PFS}$, which means $B \in S_{PFS}$, a contradiction.

A.2 Proof of P2

The derivation of the switching conditions Section 4 shows that the switching conditions are designed to imply the constraints $S^{\alpha}_{A_i A_{i'}} \cap S^{ii'}_{safe} = \emptyset$ for $i = 1..N$ where $i' = (i \bmod N) + 1$. We prove by contradiction that, if these constraints hold, there are no obstacle points inside the safety region S_{safe}. S_{safe} could be S_{SS}, S_{PS}, or S_{PFS}. The proof relies on the assumptions about obstacles presented in Section 4.

The high-level idea of the proof is that since $S^{\alpha}_{A_i A_{i'}} \cap S^{ii'}_{safe} = \emptyset$, and $S^{\alpha}_{A_i A_{i'}}$ contains all possible vertices whose edges pass through A_i and $A_{i'}$, it would be contradictory if an obstacle point is in $S^{ii'}_{safe}$.

Suppose the above constraints are satisfied and there is an obstacle point in S_{safe}, i.e., $S^{\alpha}_{A_i A_{i'}} \cap S^{ii'}_{safe} = \emptyset$ for $i = 1..N$ where $i' = (i \bmod N) + 1$, and $S_{obstacle} \cap S_{safe} \neq \emptyset$. Let $C \in S_{obstacle} \cap S_{safe}$ be an obstacle point that lies within the safety region. Since the wedges $S^{ii'}_{safe}$ cover the safety region, C must belong to at least one wedge. Without loss of generality, assume $C \in S_{obstacle} \cap S^{12}_{safe}$. We consider three cases, based on the distances OA_1 and OA_2: (1) $OA_1 = OA_2 = l_{min}$; (2) $OA_1 < OA_2 = l_{min} \vee OA_2 < OA_1 = l_{min}$; and (3) $OA_1, OA_2 < l_{min}$. Case 1 covers three sub-cases: (1a) neither sensors detects the obstacle; (1b) exactly one sensor detects the obstacle at a distance of at least l_{min}; and (1c) both sensors detect the obstacle at distances of at least l_{min}. Case 2 covers two sub-cases: (2a) exactly one sensor detects the obstacle at a distance less than l_{min}; (2b) one sensor detects the obstacle at a distance less than l_{min}, the other at a distance of at least l_{min}. Case 3 covers the only case when both sensors detect the obstacle at distances less than l_{min}. In all cases, the proof relies on the fact that there is at most one vertex of the obstacle within the triangle OA_1A_2 (because $OA_1, OA_2 \leq l_{min}$ and $\angle A_1OA_2 < \pi/3$).

A.2.1 Case 1: $OA_1 = OA_2 = l_{min}$

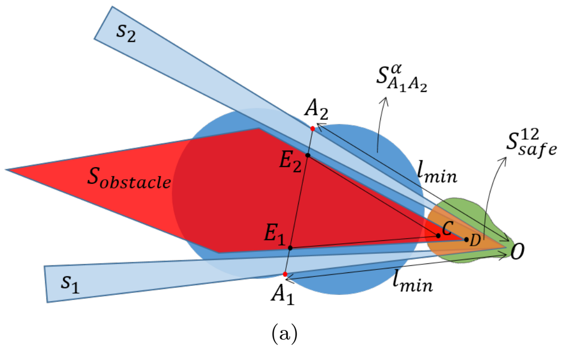
As shown in Fig. 13, case 1 comprises three sub-cases. The proof is the same for these sub-cases. For C to be in S^{12}_{safe}, there must be a vertex that fits between the sensors and intersects the safety disk. Let E_1 and E_2 be the intersections between line segment A_1A_2 and the edges of the obstacle. Let D be the vertex of the obstacle. The geometry implies $\angle A_1CA_2 \geq \angle E_1CE_2 \geq \angle E_1DE_2$. We know from our assumptions that $\angle E_1DE_2 \geq \alpha$, so $\angle A_1CA_2 \geq \alpha$. This means $C \in S^{\alpha}_{A_1A_2}$, therefore $C \in S^{\alpha}_{A_1A_2} \cap S^{12}_{safe}$. This contradicts the assumption that $S^{\alpha}_{A_1A_2} \cap S^{12}_{safe} = \emptyset$.

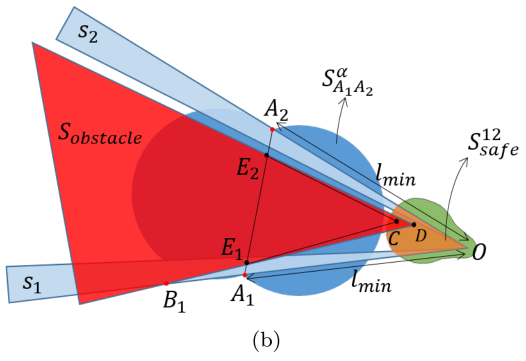
A.2.2 Case 2: $OA_1 < OA_2 = l_{min} \vee OA_2 < OA_1 = l_{min}$

Without loss of generality, assume $OA_1 < OA_2 = l_{min}$. There is a trivial sub-case when A_1 is inside the safety disk. In that case, we can choose C to be A_1 and that leads to $C \in S^{\alpha}_{A_1A_2}$, contradicting the assumption $S^{\alpha}_{A_1A_2} \cap S^{12}_{safe} = \emptyset$. Consider two sub-cases when A_1 is outside the safety disk as shown in Fig. 14. The proof for both sub-cases is as follows. For C to be in S^{12}_{safe}, there must be a vertex D of the obstacle that intersects the safety disk and has one edge that passes through A_1. Let E_2 be the intersection between line segment A_1A_2 and the obstacle's other edge incident on D. We have $\angle A_1CA_2 \geq \angle A_1CE_2 \geq \angle A_1DE_2 \geq \alpha$. Therefore $C \in S^{\alpha}_{A_1A_2}$, contradicting the assumption $S^{\alpha}_{A_1A_2} \cap S^{12}_{safe} = \emptyset$.

A.2.3 Case 3: $OA_1 < l_{min} \wedge OA_2 < l_{min}$

Suppose sensors s_1 and s_2 detect the obstacle at A_1 and A_2, respectively. We consider two sub-cases: (3a) A_1 and A_2 lie on the same edge of the obstacle; and (3b) A_1 and A_2 lie on different edges of the obstacle. It is easy to see the contradiction for case (3a), as shown in Fig. 15. Consider case (3b). As shown in Fig. 16, for C to be in S^{12}_{safe}, there must be a vertex D of the obstacle that intersects the safety disk and has one edge that passes through A_1. Let E be the intersection between A_1A_2 and the obstacle's other edge incident on D. We have





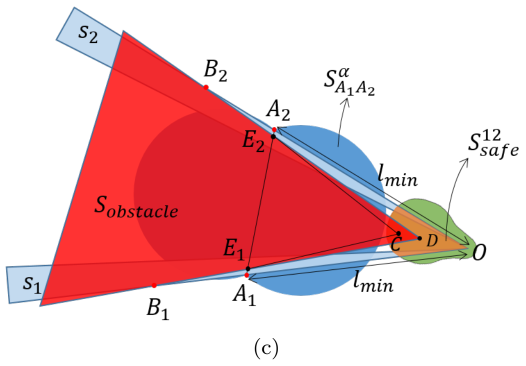
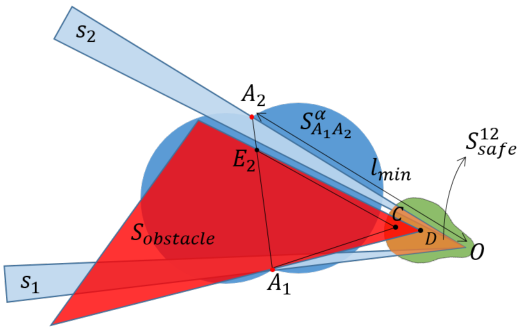
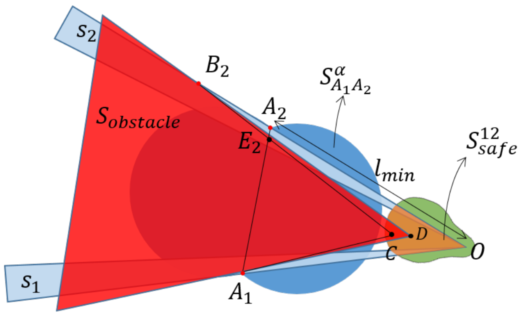


Fig. 13 Illustration of case 1. (a) Neither s_1 nor s_2 detects the obstacle. (b) Exactly one of s_1 and s_2 detects the obstacle at distance greater than l_{min}. (c) Both s_1 and s_2 detect the obstacle at distances greater than l_{min}. OB_1 and OB_2 are the actual distances detected by s_1 and s_2 respectively

$\angle A_1CA_2 \geq \angle A_1CB \geq \angle A_1DB \geq \alpha$. Therefore $C \in S^{\alpha}_{A_1A_2}$, contradicting the assumption that $S^{\alpha}_{A_1A_2} \cap S^{12}_{safe} = \emptyset$.



(a)



(b)

Fig. 14 Illustration of case 2. (a) Exactly one of s_1 and s_2 detects the obstacle within distance l_{min}. (b) One sensor detects the obstacle at distance less than l_{min}, the other detects the obstacle at distance greater than l_{min}. OB_2 is the actual distance detected by s_2

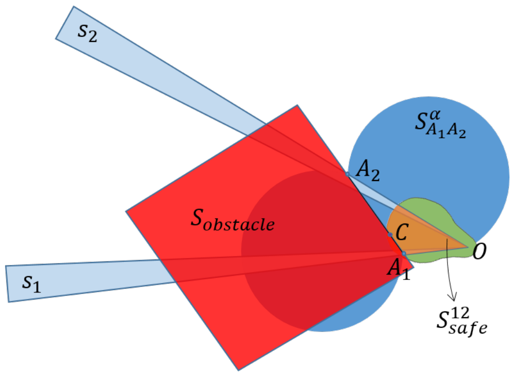


Fig. 15 Sensors s_1 and s_2 detect the obstacle at distances less than l_{min}. A_1 and A_2 both lie on the same edge. For C to be in S^{12}_{safe}, A_1A_2 must intersect S^{12}_{safe}. That means $S^{\alpha}_{A_1A_2} \cap S^{12}_{safe} \neq \emptyset$, a contradiction

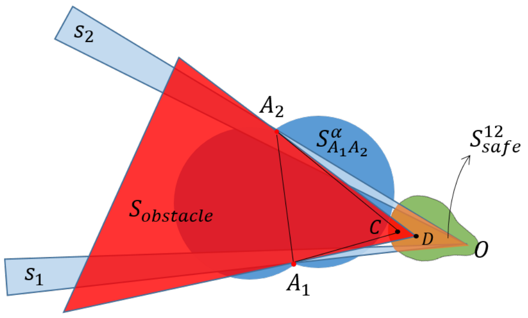


Fig. 16 Sensor s_1 detects the obstacle at A_1, sensor s_2 detects the obstacle at A_2 where $OA_1, OA_2 \leq l_{min}$