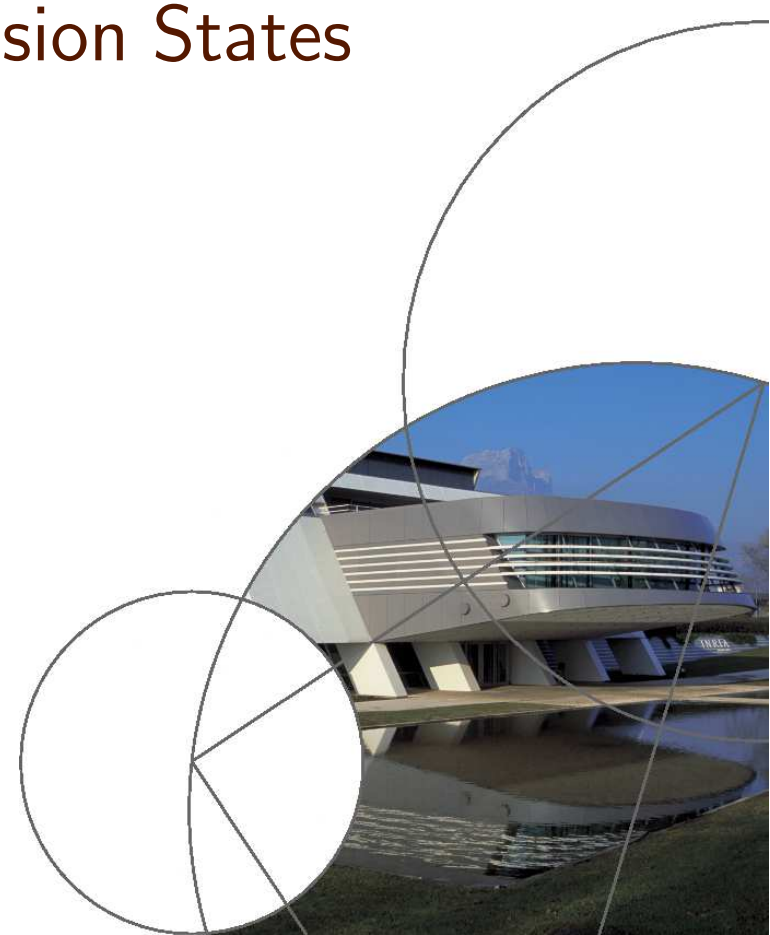
Safe Navigation for Autonomous Vehicles in Dynamic Environments: an Inevitable Collision States Perspective

PhD Defense

Luis Alfredo MARTÍNEZ GÓMEZ
Thesis Advisor: Thierry FRAICHARD



November 5, 2010

CNRS-LIG INRIA UNIVERSITÉ DE GRENOBLE

- Thank you Mr. President.
- This talk is about some of the work I've done during my PhD.
- This work has been done under the supervision of Thierry Fraichard and it is entitled Safe Navigation for Autonomous Vehicles in Dynamic Environments: an ICS Perspective

Motivation

Motion Safety in Dynamic Environments

Outdoor

Indoor

Motivation

Motion Safety in Dynamic Environments

Intelligent Transport Systems

Service Robotics

Challenges of dynamic environments:

1. Understand them $\rightsquigarrow$ model of the environment
2. Safely navigate them

Safe Navigation for Autonomous Vehicles in Dynamic Environments: an ICS Perspective

Motivation

Intelligent Transport Systems

Service Robotics

Challenges of dynamic environments:

1. Understand them $\rightsquigarrow$ model of the environment
2. Safely navigate them

- The main motivation of this work is to address the motion safety probelm in dynamic environments
- Dynamic environments are the typical indoor or outdoor spaces where persons live and work
- If robots are ever going to share this space with us they will have to operate in a safe manner in this type of environment.
- Moving around without causing harm to objects or persons is then one of the key abilities they will need to perform.
- Examples of applications where robots are envisioned to operate side by side with persons are ITS & Service Robotics
- ITS such as autonomous vehicles promise to reduce road accidents by eliminating human errors and also reduce the environmental impact of private cars by for example increasing the throughput of traffic flow
- Service Robotics may be the answer to the problems we will face when the average age of our population inrease. Current tendencies show that the percentage of senior population will be much larger than the younger population.
- This will create a defict of young people that could provide care to the senior citizens.
- To fill this gap robots could provide services to fill the needs of our senior citizens. Examples of such services are to increase their personal mobilty with automatic wheelchairs.
- This applications will need to deal with dynamic environments and the challenges they pose.
- The first challenge has to do with the perception of the environment. Understand what is happening in order to produce a model of it.
- The second challenge once this information or model is available is how the robot use it in order to safely navigate.
- This thesis concentrates in the second challenge: how to safely navigate a dynamic environment.

- So the first and stronger assumption we made is that such model of the environment already exists

Outline

- So the outline of this work is organized as follows:
- First I will start by presenting an analysis of the motion safety problem to highlight the key issues required to address it appropriately and then see if the current navigation methods consider or not these key issues
- I will continue by introducing a concept called Inevitable Collision States that help to deal with the motion safety of systems in dynamic environments.
- The next two sections are focused in putting into practice this concept of Inevitable Collision States
- The first part will describe an algorithm to identify or characterize such states
- While the second will use this characterization inside a simple algorithm to produce a safe navigation scheme.
- Afterwards I will present some of the simulation and experimental result we obtained in this work
- And then finalize with some conclusions and perspectives

Motion Safety Analysis

- Lets start with the Motion Safety Analysis

Safe Navigation for Autonomous Vehicles? Hum. . . Not Yet

DARPA Urban Challenge *[Nov. 2007]*

Safe Navigation for Autonomous Vehicles? Hum. . . Not Yet

DARPA Urban Challenge *[Nov. 2007]*

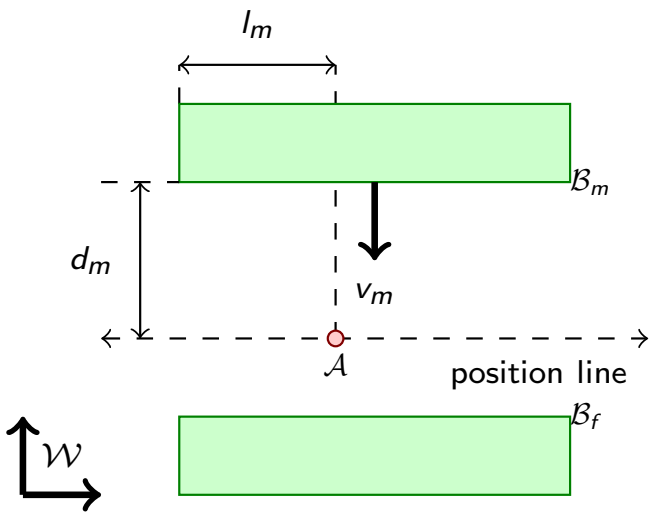
- One thing we can all agree with is that the navigation of autonomous vehicles is not an easy problem
- Although recent and quite successful efforts have shown that the research community have made a lot of progress in this area we can not claim yet that the motion safety problem is solved
- This is quite well illustrated in the pictures shown in the slide. They were taken in the last DARPA Urban Challenge. The challenged called for autonomous vehicles to navigate 96 km through a typical urban environment.
- The challenge was in general very succesful, however several accidents occur. Many reasons can be given for this to happen, but lets take a look of the motion safety problem from a theoretical perspective to see if we can find some reasons there that can explain why this accidents can occur

Moving Safely in Dynamic Environments

Key aspect: **time**

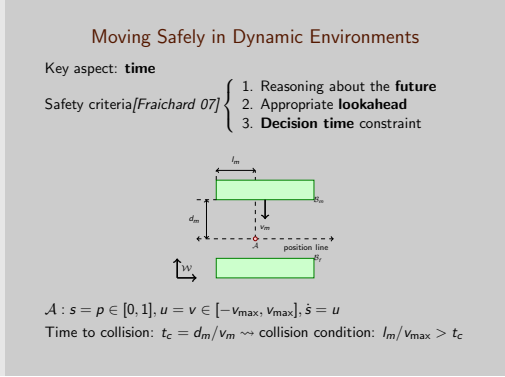
Safety criteria *[Fraichard 07]*

1. Reasoning about the **future**
2. Appropriate **lookahead**
3. **Decision time** constraint



$\mathcal{A} : s = p \in [0, 1], u = v \in [-v_{\max}, v_{\max}], \dot{s} = u$

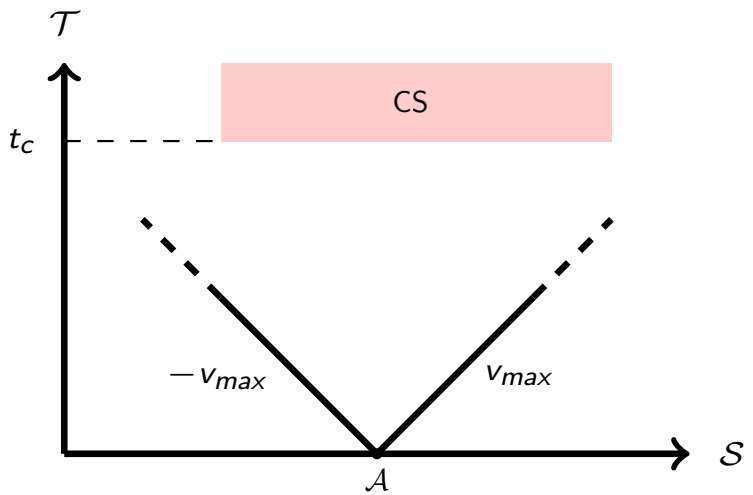
Time to collision: $t_c = d_m / v_m \rightsquigarrow$ collision condition: $l_m / v_{\max} > t_c$



- Moving safely in dynamic environments is considerably much harder than doing it in static environments
- The main difference between a static and a dynamic environment is that in the latter the objects' positions change with time.
- This key aspect of time, was used as main argument in the analysis done by Fraichard, that identified 3 key issues or safety criteria concerning the motion safety of a robotic system in a dynamic environment.
- They are: reasoning about the future, with an appropriate lookahead while respecting a decision time constraint
- To illustrate the relevance of this 3 criteria I'm going to use a simple example
- We call it the "compactor scenario". Imagine you have two objects in the workspace. One fixed in the bottom and one moving in the top. The moving one advance with certain speed, here noted as v_m, towards the fixed one (like closing a sandwich).
- If we have a robotic system in between this two objects is clear that a collision will occur if the robot doesn't move away.
- To simplify we will constraint the robot motion to an horizontal line: so the robot can only move to the left or to the right
- This simplification allows us to define a one dimensional state space that in this case is the position of the system in the line.
- The control of the system is simply a velocity which is constrained to a maximum value and the equation describing the evolution of the system state is just the applied control. When applying a velocity the position of the system changes
- With additional information such as the distance between the current position of the robot to the closest border of the obstacle and the distance that separeates the position line to the moving obstacle
- We can compute the time a collision will occur. Its simply the distance m divided by the obstacle velocity. If this time to collision is shorter than the time it takes the system to move away: l_m / v_{max} then a collision can not be avoided.
- Lets assume this is not the case and that the system can escape a collision.

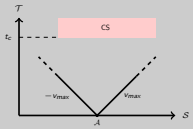
- Now, let's return to our safety criteria and see their impact in the decision procees of the robotic system in

1st Safety Criteria: Reasoning about the future



State-Time Space*[Fraichard 92]*

- Robot's dynamics: reachable states
- Objects' future behaviour: collision states CS
- Model of the environment required



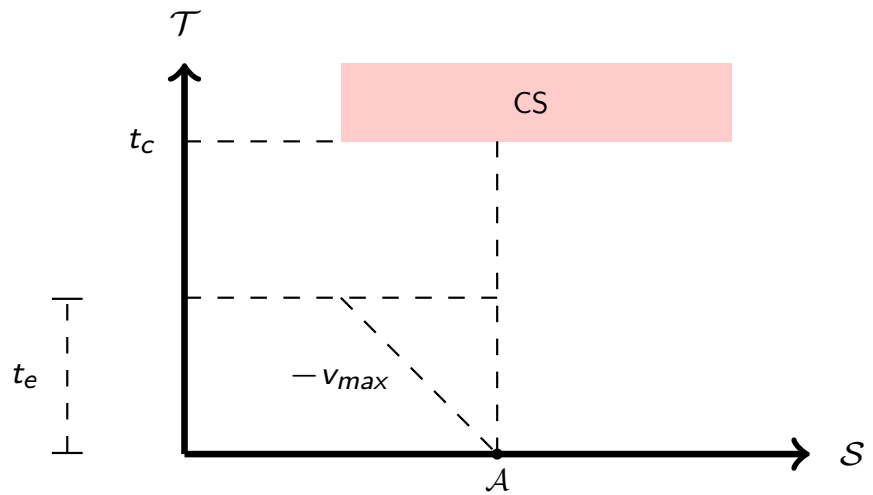
State-Time Space*[Fraichard 92]*

- Robot's dynamics: reachable states
- Objects' future behaviour: collision states CS
- Model of the environment required

- Reasoning about the future means above all to consider the time dimension.
- One way of doing this is by employing a representation called State-Time
- In this representation we add the time dimension to the state space.
- In our compactor scenario which has a 1d state space we end in a 2d state-time space.
- Now, here we can represent our robot capabilities as the set of reachable states.
- In our example such states are a cone with apex in the current state, bounded by the max velocity of the system. Anything inside the cone is possible for the robot, anything outside the cone is impossible.
- Additionally is also possible to represent the future behaviour of the objects in the environment. In particular it permits to represent the collision constraints imposed by the moving objects.
- The movement of the object in the scenario cause the creation of collision states. Those states cover the length of the obstacles and they start at t_c, the time the obstacle arrives at the position line of the system.
- Then a CS is the state where the robot and the object are in collision at a specific time.
- The relevance of reasoning about the future is that this CS appear in the decision process of the system. If it hasn't been done the robot will have never known a collision was coming. This of course has an impact in its safety.

2nd Safety Criteria: Lookahead (t_{la})

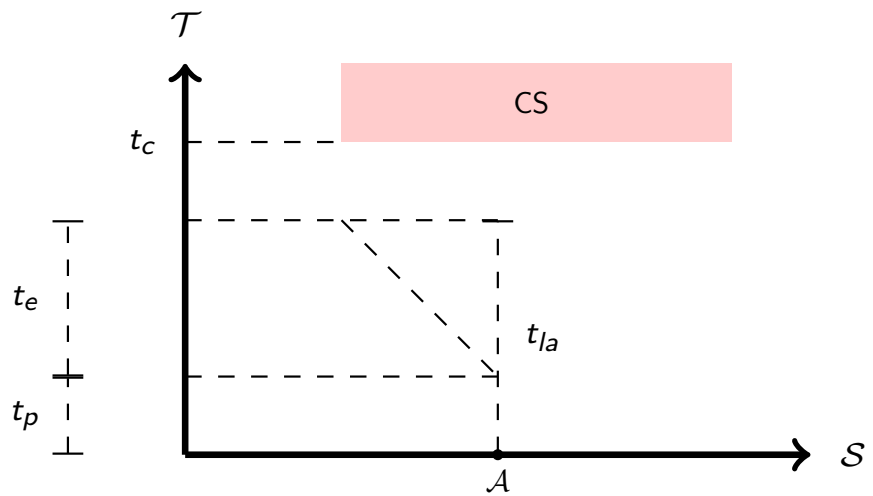
How far into the future should the modeling/reasoning go?



- t_{la} depends on the $\mathcal{A}$'s (dynamics & state) and environment
- $t_e = l_m / v_{max}$
- If $t_{la} = t_e$ then $\mathcal{A}$ will know it needs to escape from collision right away

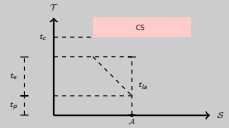
2nd Safety Criteria: Lookahead (t_{la})

How far into the future should the modeling/reasoning go?



- t_{la} depends on the $\mathcal{A}$'s (dynamics & state) and environment
- $\mathcal{A}$ needs t_p to make a plan, then $t_{la} \geq t_e + t_p$
- t_{la} can become arbitrary large, *e.g.*, $v_m \to 0$ and $l_m \to \infty \rightsquigarrow t_e \to \infty$ then $t_{la} \to \infty$

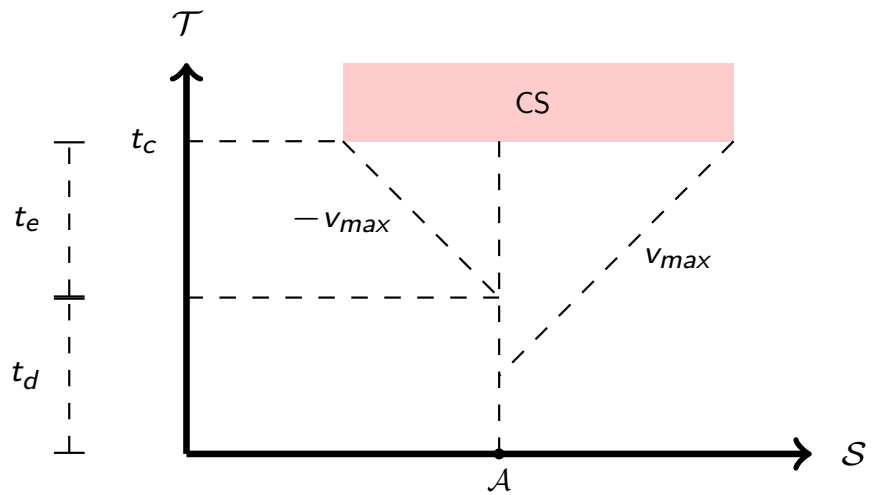
2nd Safety Criteria: Lookahead (t_{la})
How far into the future should the modeling/reasoning go?



- t_{la} depends on the $\mathcal{A}$'s (dynamics & state) and environment
- $\mathcal{A}$ needs t_p to make a plan, then $t_{la} \geq t_e + t_p$
- t_{la} can become arbitrary large, *e.g.*, $v_m \to 0$ and $l_m \to \infty \rightsquigarrow t_e \to \infty$ then $t_{la} \to \infty$

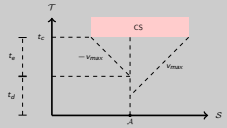
- Now that we know we should consider the future the question is how far in the future should this modeling/reasoning go. The answer to this question is not simple, in fact it depends in the dynamics and state of the system and the environment.
- The intuition is that the lookahead time should go to the point where no more information of the future evolution of the environment is useful to make a decision.
- In our example things are relatively easy.
- Lets denote t_e as the minimum time it takes the system to escape collision from its current state.
- If the lookahead time is set to this value, then the system will look far enough into the future to realize if it needs to escape from collision right away or it can do something else.
- In the case of the example our system can do something else, but if the system is in the same state at a later time in the future then it has no other option that start moving right away
- Now, the robotic system can not make a decision immediately, it needs certain time, lets denote this time t_p, or planning time, then it needs then $t_e + t_p$
- The problem with the lookahead time is that it can become arbitrarly large. Consider $\mathcal{B}_m$ is very long and very slow, then the distance that separates the

3rd Safety Criteria: Decision time constraint (t_d)



- t_d also depends on the $\mathcal{A}$'s (dynamics & state) and the environment
- Can not be arbitrarily long $\rightsquigarrow$ risk of collision
- $t_d = t_c - t_e$
- t_d can become very small, *e.g.*, $l_m \to \infty \rightsquigarrow t_e \to \infty$ then $t_d \to 0$

3rd Safety Criteria: Decision time constraint (t_d)



- t_d also depends on the $\mathcal{A}$'s (dynamics & state) and the environment
- Can not be arbitrarily long $\rightsquigarrow$ risk of collision
- $t_d = t_c - t_e$
- t_d can become very small, *e.g.*, $l_m \to \infty \rightsquigarrow t_e \to \infty$ then $t_d \to 0$

- For our 3rd and last safety criteria we need to consider the upper bound the robotic system has in its planning time.
- We have seen that the planning time was necessary because the robotic system can not make decision instantaneously.
- Now a dynamic environment impose a limit in the available time the robot has to make such decision.
- The value of this upper bound also depends on the current state and of the environment
- If it takes too long a collision may occur
- In our example the system has to possible actions. Take the left or the right
- The minimum time it takes to escape is when it takes the left, does, the upper bound of the planning time, what we call the decision time is then $t_c - t_e$
- Similar to the lookahead time, this value can become arbitrarly small. Consider again the case the moving obstacle becomes arbitrarly large, then the time to escape goes again to a very high value which in turns reduce the upper bound to make a decision since the decision time was $t_c - t_e$

Moving Safely in Dynamic Environments

Q: how to guarantee **motion safety**, *i.e.*, never ending up in a situation where a collision eventually occurs?

A: consider the 3 safety criteria:

1. Reasoning about the future 2. appropriate t_{la} 3. t_d constraint

Does today's navigation schemes consider these safety criteria?

Navigation Method	(1)	(2)	(3)
Roadmap methods	√	×	×
Cell Decomposition	×	×	√
Sampling Methods	√	√	×
Potential Field	×	×	√
Vector Field Histogram	×	×	√
Curvature Velocity	×	×	√
Lane Curvature	×	×	√
Dynamic Window Approach	×	×	√
Time-varying Dynamic Window Approach	√	×	√
Dynamic Velocity Space	√	×	√
Velocity Obstacles	√	×	√
Non-linearVelocity Obstacles	√	×	√
Trajectory Parameter Space	×	×	√

Moving Safely in Dynamic Environments

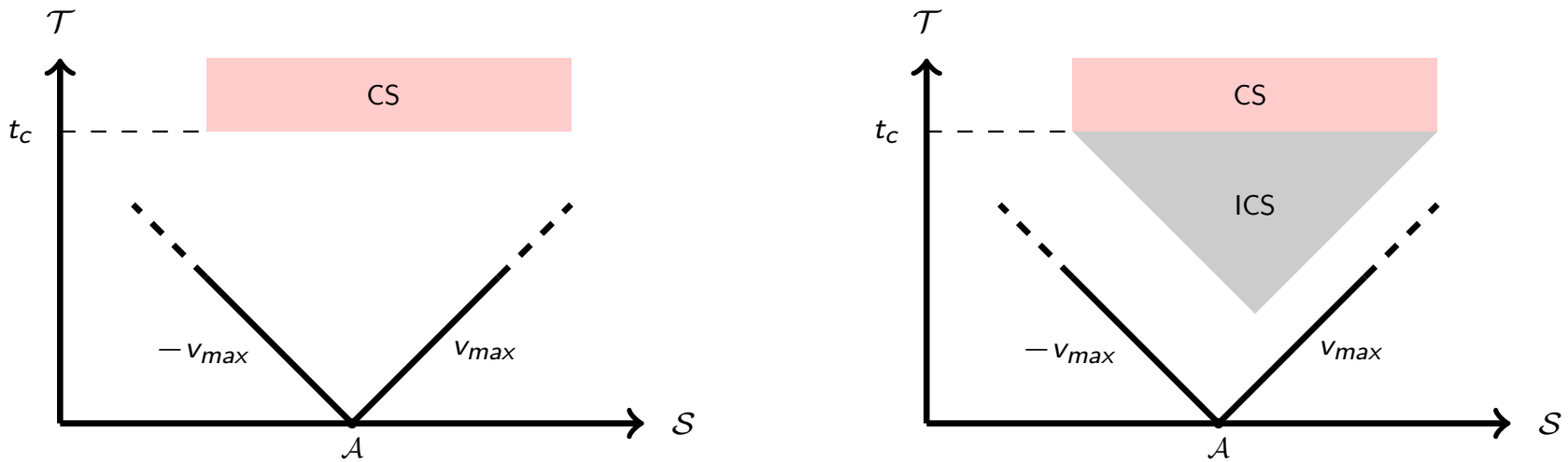
Q: how to guarantee **motion safety**, *i.e.,* never ending up in a situation where a collision eventually occurs?
A: consider the 3 safety criteria:
1. Reasoning about the future 2. appropriate t_{la} 3. t_{cl} constraint

Does today's navigation schemes consider these safety criteria?

Navigation Method	(1)	(2)	(3)
Roadmap methods	√	×	×
Cell Decomposition	×	×	√
Sampling Methods	√	√	×
Potential Field	×	×	√
Vector Field Histogram	×	×	√
Curvature Velocity	×	×	√
Lane Curvature	×	×	√
Dynamic Window Approach	×	×	√
Time-varying Dynamic Window Approach	√	×	√
Dynamic Velocity Space	√	×	√
Velocity Obstacles	√	×	√
Non-linearVelocity Obstacles	√	×	√
Trajectory Parameter Space	×	×	√

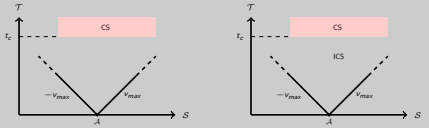
- To summarize we have three safety criteria needed to address properly the motion safety problem.
- Reason about the future, doing it witha appropriate lookahead time while respecting the time decision constraint
- So the question is: how can we guarantee motion safety? that is never ending up in a situation where a collision eventually occurs
- The answer is of course to consider the 3 safety criteria
- But the navigation methods present in the literature address them all?
- In most cases,they don't. They are **unsafe in dynamic environments**: violation of (1) and (2) *[Fraichard 07]*
- However one concept that gives an answer to the motion safety problem when viewed from these criterias is called ICS.

Inevitable Collision States (ICS)*[Fraichard 03]*



ICS: whatever the future trajectory followed by a robotic system is, a collision eventually occurs

Inevitable Collision States (ICS)*[Fraichard 03]*



ICS: whatever the future trajectory followed by a robotic system is, a collision eventually occurs

- ICS are the states from whatever the trajectory of the system is, a collision eventually occur
- In our example we have already identify the states that were collision states. That is the robotic system and an obstacle has already intersect in the workspace.
- But theres also a region that if the system is in it it won't be able to avoid a collision in the future.
- This region is the gray area in the figure.
- If the robot happens to be there, its dynamics constraints will not allow it to avoid a collision in the future, no matter what it does. This is the ICS
- Motion safety means not only to avoid the collision states but also the inevitable collision states

Inevitable Collision States

- Lets review this concept.

Inevitable Collision States (ICS)*[Fraichard 03]*

General concept: $\forall$ robotic system, $\forall$ environment

Notation:

- Robotic system: $\mathcal{A}$
- $\mathcal{A}$'s dynamics: $\dot{s} = f(s, u)$
- State: $s \in \mathcal{S}$
- Control: $u \in \mathcal{U}$
- Control trajectory: $\tilde{u} : [0, \infty] \longrightarrow \mathcal{U}, \tilde{u} \in \tilde{\mathcal{U}}$
- State reached at time t when executing $\tilde{u}$ from state s: $\tilde{u}(s, t)$
- Model of the environment: $\mathcal{B} = \bigcup_i \bigcup_t \mathcal{B}_i(t)$, $\forall i$, $\forall t$

Formal definitions:

$$\begin{aligned} \text{ICS}(\mathcal{B}) &= \{s \in \mathcal{S} | \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \emptyset\} \\ \text{ICS}(\mathcal{B}_i, \tilde{u}) &= \{s \in \mathcal{S} | \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\} \end{aligned}$$

Inevitable Collision States (ICS)*[Fraichard 03]*

General concept: $\forall$ robotic system, $\forall$ environment

Notation:

- Robotic system: $\mathcal{A}$
- $\mathcal{A}$'s dynamics: $\dot{s} = f(s, u)$
- State: $s \in \mathcal{S}$
- Control: $u \in \mathcal{U}$
- Control trajectory: $\tilde{u} : [0, \infty] \longrightarrow \mathcal{U}, \tilde{u} \in \tilde{\mathcal{U}}$
- State reached at time t when executing $\tilde{u}$ from state s: $\tilde{u}(s, t)$
- Model of the environment: $\mathcal{B} = \bigcup_i \bigcup_t \mathcal{B}_i(t)$, $\forall i$, $\forall t$

Formal definitions:

$$\text{ICS}(\mathcal{B}) = \{s \in \mathcal{S} | \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \emptyset\}$$

$$\text{ICS}(\mathcal{B}_i, \tilde{u}) = \{s \in \mathcal{S} | \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\}$$

- The ICS concept is general. It applies to all kind of robotic systems and all kinds of environments
- To arrive to a formal definition of it I will need to review some notation
- The robotic system is denoted by A, and its dynamics are a function that depends in the state and a control. Both of them belonging to the state space and control space respectively.
- Now a control trajectory, noted as u tilde, is a sequence of controls in time. From time zero to the infinite.
- The set of all possible control trajectories is noted as uppercase U tilde.
- When the robotic system uses one of such control trajectories starting from an initial state s. Then it means that at each time a control will be applied to the system and consequently its state will change. If we take for a given time the state reached, we are going to note that as u tilde, s, t
- With respect to the information of the environment we are going to denote it as B. B represents all the information. Which means the union of the information we have of particular objects in the environment along the time
- Then the formal defintion of ICS is a state such as for all control trajectories there is a time where a collision will occur.
- A similar defintion allows to identify the states from where applying a particular control trajectory provoques that the system has a collision with a particular object

- This simpler definition of states denoted as ICS Bi u jay will allow us to build the more general set of ICS

Inevitable Collision States and Motion Safety

s is an **ICS** iff $\forall \tilde{u} : [0, \infty] \longrightarrow \mathcal{U}, \exists t, \mathcal{A}(\tilde{u}(s, t)) \cap \mathcal{B}(t) \neq \emptyset$

1. Future
   - $\mathcal{A}$ (system's dynamics)
   - $\mathcal{B}_i$ (future motion)
2. Lookahead
   - Up to $\infty \rightsquigarrow$ appropriate
3. Time decision constraint

s is an **ICS** iff $\forall \tilde{u} : [0, \infty] \longrightarrow \mathcal{U}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \emptyset$

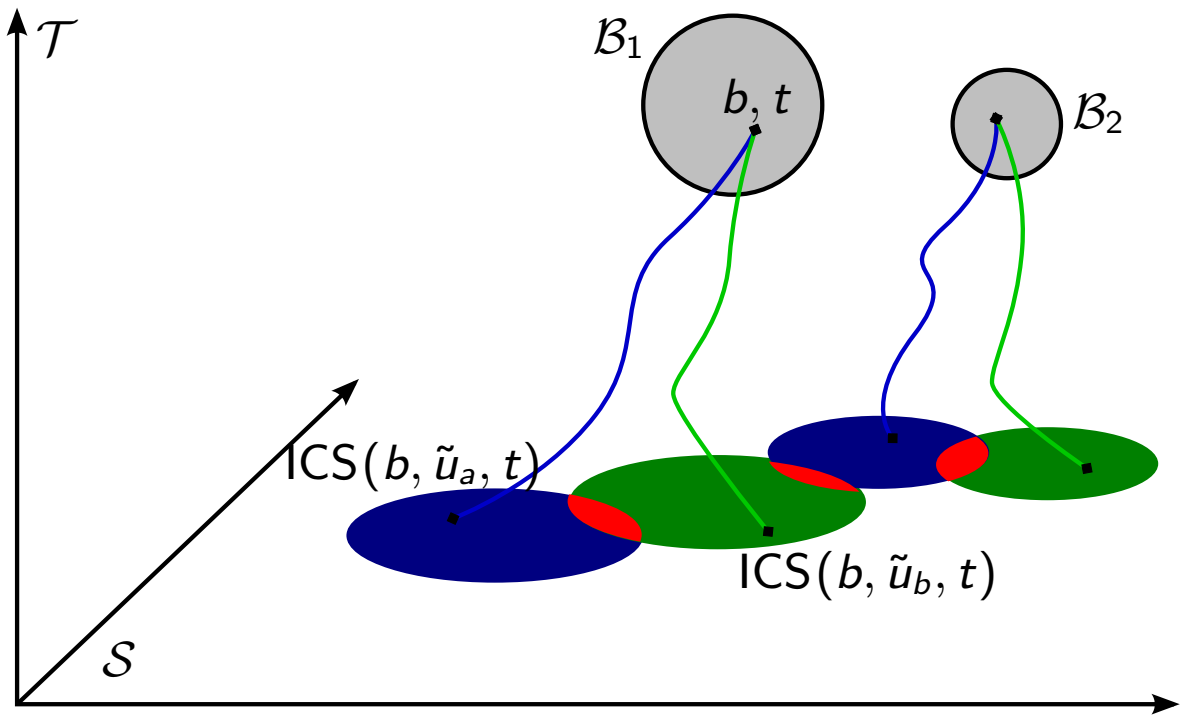
1. Future
   - $\mathcal{A}$ (system's dynamics)
   - $\mathcal{B}_i$ (future motion)
2. Lookahead
   - Up to $\infty \rightsquigarrow$ appropriate
3. Time decision constraint

- Now how does the ICS concept relates to the safety criteria we had review
- From the definition we have that it reasons about the future by considering the dynamics of A and the objects future motion
- And it reasons of the future with a lookahead time up to infinite and thus appropriate.
- But the definition does not allow us to see if this concept will allow to respect the constraint of the decision time.

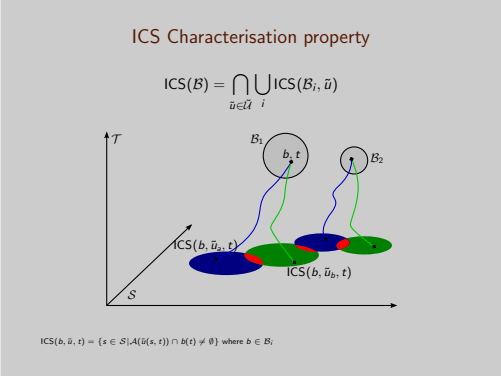
- This of course will depend in how much time it takes to identify or characterize such ICS

ICS Characterisation property

$$\mathrm{ICS}(\mathcal{B}) = \bigcap_{\tilde{u} \in \tilde{\mathcal{U}}} \bigcup_{i} \mathrm{ICS}(\mathcal{B}_i, \tilde{u})$$



$\mathrm{ICS}(b, \tilde{u}, t) = \{s \in \mathcal{S} | \mathcal{A}(\tilde{u}(s, t)) \cap b(t) \neq \emptyset\}$ where $b \in \mathcal{B}_i$



- To characterize them we use a property.
- This property starts from our simpler definition of states from where the robotic system will collide with a particular object when applying a particular control trajectory
- Once this states can be computed all is left to do is to perform a series of union and intersections to arrive to the final ICS set
- The figure illustrate the procedure. Imagine the system has only 2 control trajectories here shown with the blue and green lines.
- Lets fix an arbitrary time t. We know from our model of the environment where the obstacles will be at such time t.
- What, we want to identify are the inital states from where the robotic system will arrive to a collision if it applies one of the control trajectories
- Lets take as example the blue trajectory, u a, we know where obstacle 1 is at time t. If the robotic system is in any of the states here illustrated in blue then a collision will occur at time t if the robot applies the blue control trajectory
- Similar reasoning goes for the second obstacle. If it happens to be in any of the states here, at time t will collide if it applies the blue control trajectory.
- This individual circles are our building blocks up here when we consider all times.
- When we perform a union, here illustrated by using the same color, blue or green for the second trajectory, we arrive to the states where the robotic system will collide with any obstacle in the environment.
- Now if we are in the blue circles and we apply the second trajectory we may not collide. Take this point for example. See, no collision. So we are interested in the states where both control trajectories takes us to collision. That is the intersection. Here illustrates in red
- The problem with all this is that we need to do this for all control trajectories which for even simple system may be infinite

- Thats why we need to make an approximation

ICS Approximation property

Let $\mathcal{E} \subset \tilde{\mathcal{U}}$

$$\text{ICS}(\mathcal{B}) \subset \text{ICS}(\mathcal{B}, \mathcal{E})$$

$$\text{ICS}(\mathcal{B}, \mathcal{E}) = \{s \in \mathcal{S} | \forall \tilde{u} \in \mathcal{E}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \emptyset\}$$

How to choose $\mathcal{E}$? s is not an ICS iff $\exists \tilde{u}, \forall t, \tilde{u}(s,t)$ is collision-free

Common sense answer: $\mathcal{E}$ should contain **evasive manoeuvres**

ICS Approximation property

Let $\mathcal{E} \subset \tilde{\mathcal{U}}$

$$\mathrm{ICS}(\mathcal{B}) \subset \mathrm{ICS}(\mathcal{B}, \mathcal{E})$$

$\mathrm{ICS}(\mathcal{B}, \mathcal{E}) = \{s \in \mathcal{S} | \forall \tilde{u} \in \mathcal{E}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \emptyset\}$

How to choose $\mathcal{E}$? s is not an ICS iff $\exists \tilde{u}, \forall t, \tilde{u}(s,t)$ is collision-free

Common sense answer: $\mathcal{E}$ should contain **evasive manoeuvres**

- The approximation property allow us to make an over approximation of the real set of ICS
- This means that we may classify some states as an ICS when in fact they are not but we will never classify an state as not belonging to the ICS set when it does belong
- The property use a subset of all control trajectories, so, instead of having to verify collision for all possible control trajectories we will do it only for the chosen subset
- The question now is how to select the subset
- The answer is of course to select control trajectories that avoid collision: that is to choose evasive manoeuvres
- An important point here is that the approximation can only get better if evasive manouevres are added,

  when a collision manouevre is chosen of course no improvement in the quality of the approximation will be

  made

General ICS-Checking Algorithm

s_c ICS?
Input: $s_c, \mathcal{B}$
Output: Boolean

---

1. Select $\mathcal{E}$
2. Compute $\text{ICS}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}(\mathcal{B}, \tilde{u}_j) = \bigcup_i \text{ICS}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$
4. Compute $\text{ICS}(\mathcal{B}) = \bigcap_j \text{ICS}(\mathcal{B}, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
5. Check whether $s_c \in \text{ICS}(\mathcal{B})$, return TRUE or FALSE accordingly

---

Key issue:

$\text{ICS}(\mathcal{B}_i, \tilde{u}_j) \rightsquigarrow$ state-time obstacles (**Curse of Dimensionality**)

General ICS-Checking Algorithm

s_c ICS?
Input: $s_c, \mathcal{B}$
Output: Boolean

1. Select $\mathcal{E}$
2. Compute $\text{ICS}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}(\mathcal{B}, \tilde{u}_j) = \bigcup_i \text{ICS}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$
4. Compute $\text{ICS}(\mathcal{B}) = \bigcap_j \text{ICS}(\mathcal{B}, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
5. Check whether $s_c \in \text{ICS}(\mathcal{B})$, return TRUE or FALSE accordingly

Key issue:

$\text{ICS}(\mathcal{B}_i, \tilde{u}_j) \rightsquigarrow$ state-time obstacles (**Curse of Dimensionality**)

- With this background we can spell out the algorithm that allows to make the characterization of ICS
- The input of the algorithm is simply to check if a given state belongs to the ICS set or not
- The output is simple yes or no
- The steps are to select a subset of control trajectories
- To build the set of states from where the robotic system will arrive to collision with a particular object in the environment when using a particular control trajectory
- Then make the union by fixing the control trajectory and grouping the states from where the robotic system will arrive to collision with any object of the environment when using a particualr control trajectory
- And then making the intersection of all control trajectories to arrive to the final ICS set
- Then all its left to do is to verify if the given state belongs or not to this final set
- Although this algorith seems simple remember that all the operations are done in the state time space, which in general has many dimensions and thus may become intractable
- What is needed is then a mechanism to make feasible to verify the icsness of a state in an efficient way in order to be able to respect the time decision constraint.

- This is what we have done. The algorithm is called ICS-CHECK

ICS-CHECK, a 2D ICS-Checking Algorithm

- Lets see how this algorithm works

ICS-CHECK

Assumptions:

- 2D workspace $\mathcal{W}$
- System $\mathcal{A}$: planar, arbitrary shape, arbitrary dynamics $\dot{s} = f(s, u)$
- Objects $\mathcal{B}_i$: arbitrary shapes, deterministic model of their future motions $\mathcal{B}_i(t)$

Principle:

1. 2D reasoning

Efficient Implementation:

1. GPU processing

ICS-CHECK

Assumptions:

- 2D workspace $\mathcal{W}$
- System $\mathcal{A}$: planar, arbitrary shape, arbitrary dynamics $\dot{s} = f(s, u)$
- Objects $\mathcal{B}_i$: arbitrary shapes, deterministic model of their future motions $\mathcal{B}_i(t)$

Principle:

1. 2D reasoning

Efficient Implementation:

1. GPU processing

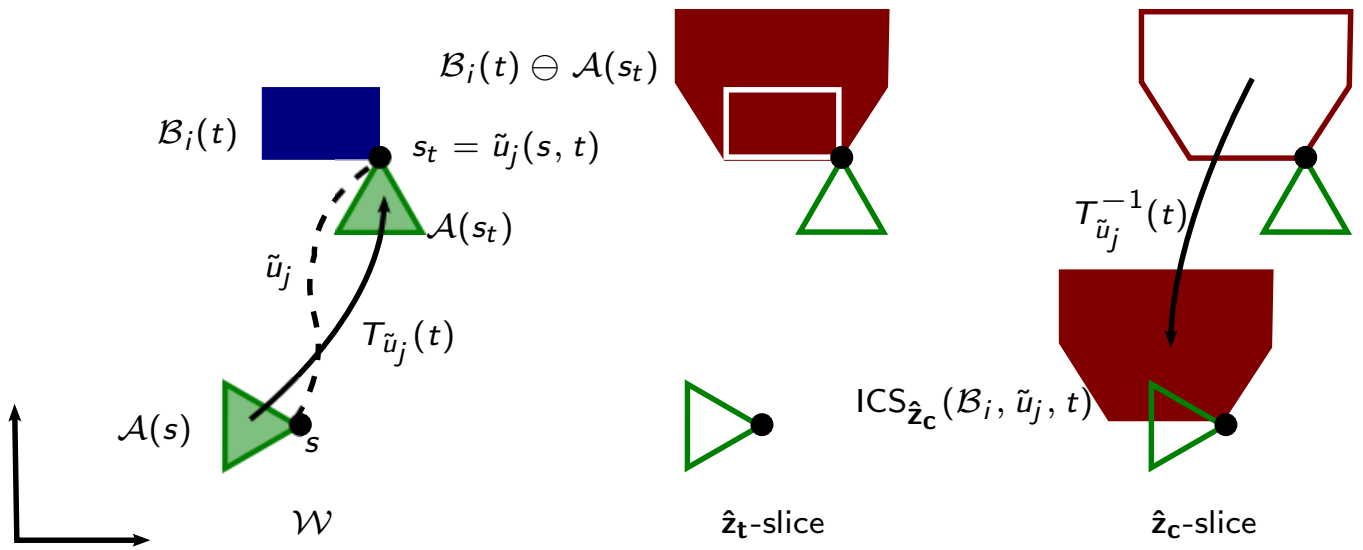
- A

ICS-CHECK Principle: 2D Reasoning

Check $s_c = (x_c, y_c, \hat{\mathbf{z}}_\mathbf{c}) \rightsquigarrow$ **compute ICS set for $\hat{\mathbf{z}}_\mathbf{c}$-slice only**

$\hat{\mathbf{z}}_\mathbf{c}$-slice **diffeomorphic** to $\mathcal{W} \rightsquigarrow \mathcal{B}_i{}^{\mathcal{W}}(t) \equiv \mathcal{B}_i{}^{\hat{\mathbf{z}}_\mathbf{c}}(t)$

$$\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j, t) = \{s \in \hat{\mathbf{z}}_\mathbf{c}\text{-slice} | \mathcal{A}(\tilde{u}_j(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\}$$



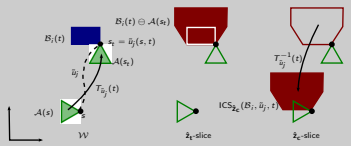
$$\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j, t) = \{s = (x, y, \hat{\mathbf{z}}_\mathbf{c}) | (x,y) \in T_{\tilde{u}_j}^{-1}(t)[\mathcal{B}_i(t) \ominus \mathcal{A}(s_t)]\}$$

ICS-CHECK Principle: 2D Reasoning

Check $s_c = (x_c, y_c, \hat{\mathbf{z}}_c) \rightsquigarrow$ **compute ICS set for $\hat{\mathbf{z}}_c$-slice only**

$\hat{\mathbf{z}}_c$-slice **diffeomorphic** to $\mathcal{W} \rightsquigarrow \mathcal{B}_i^{\mathcal{W}}(t) \equiv \mathcal{B}_i^{\hat{\mathbf{z}}_c}(t)$

$$\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j, t) = \{s \in \hat{\mathbf{z}}_c\text{-slice} | \mathcal{A}(\tilde{u}_j(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\}$$



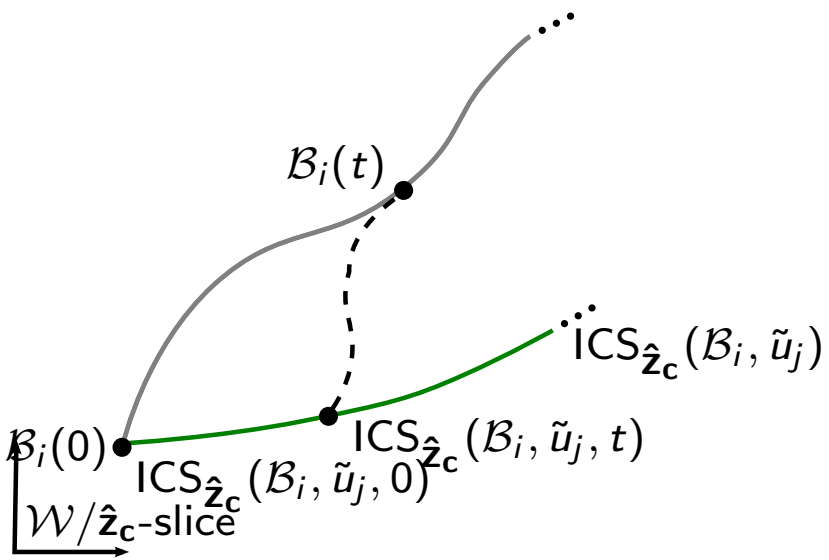
$$\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j, t) = \{s = (x, y, \hat{\mathbf{z}}_c) | (x, y) \in T_{\tilde{u}_j}^{-1}(t)[\mathcal{B}_i(t) \ominus \mathcal{A}(s_t)]\}$$

- A

ICS-CHECK Principle: 2D Reasoning

The region swept by $\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j, t)$ for all time instants:

$$\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_{t\in[0,\infty(} \mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j, t)$$



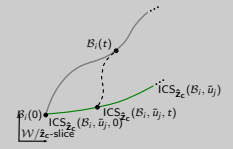
General shape of $\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ is a semi-infinite stripe

2D computations only $(\cup, \cap)$, irrespective of $\dim(\mathcal{S})$

ICS-CHECK Principle: 2D Reasoning

The region swept by $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j, t)$ for all time instants:

$$\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_{t \in [0,\infty[} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j, t)$$



General shape of $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ is a semi-infinite stripe
2D computations only $(\cup, \cap)$, irrespective of $\dim(\mathcal{S})$

- A

ICS-CHECK Algorithm

s_c ICS?
Input: $s_c, \mathcal{B}$
Output: Boolean

---

1. Select $\mathcal{E}$
2. Compute $\text{ICS}_{\hat{\mathbf{Z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}_{\hat{\mathbf{Z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \text{ICS}_{\hat{\mathbf{Z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\text{ICS}_{\hat{\mathbf{Z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \text{ICS}_{\hat{\mathbf{Z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \text{ICS}_{\hat{\mathbf{Z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly

---

ICS-CHECK Algorithm

s_c ICS?
Input: $s_c, \mathcal{B}$
Output: Boolean

1. Select $\mathcal{E}$
2. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly

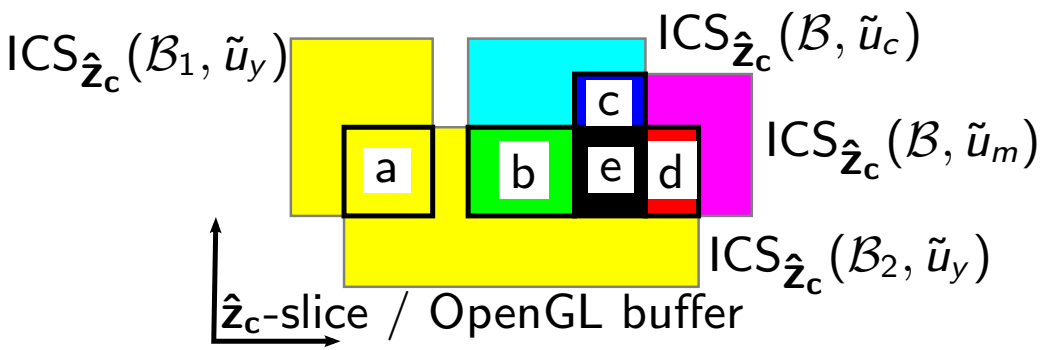
- A

ICS-CHECK GPU Implementation

Using graphics rendering techniques to compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$:

- Assign RGB colour (col_j) to each $\tilde{u}_j \in \mathcal{E}$
- Colour assignment satisfy:

$$\bigvee_{\tilde{u}_j \in \mathcal{E}} col_j = \#000000 \text{ and } \bigvee_{\tilde{u}_j \in \mathcal{E}' \subset \mathcal{E}} col_j \neq \#000000$$



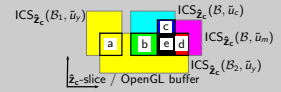
Parallel processing: $T_{\tilde{u}_j}^{-1}(t)[\mathcal{B}_i(t) \ominus \mathcal{A}(s_t)]$

ICS-CHECK GPU Implementation

Using graphics rendering techniques to compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$:

- Assign RGB colour (col_j) to each $\tilde{u}_j \in \mathcal{E}$
- Colour assignment satisfy:

$$\bigvee_{\tilde{u}_j \in \mathcal{E}} col_j = \#000000 \text{ and } \bigvee_{\tilde{u}_j \in \mathcal{E}' \subset \mathcal{E}} col_j \neq \#000000$$



Parallel processing: $T_{\tilde{u}_j}^{-1}(t)[\mathcal{B}_i(t) \ominus \mathcal{A}(s_t)]$

- A

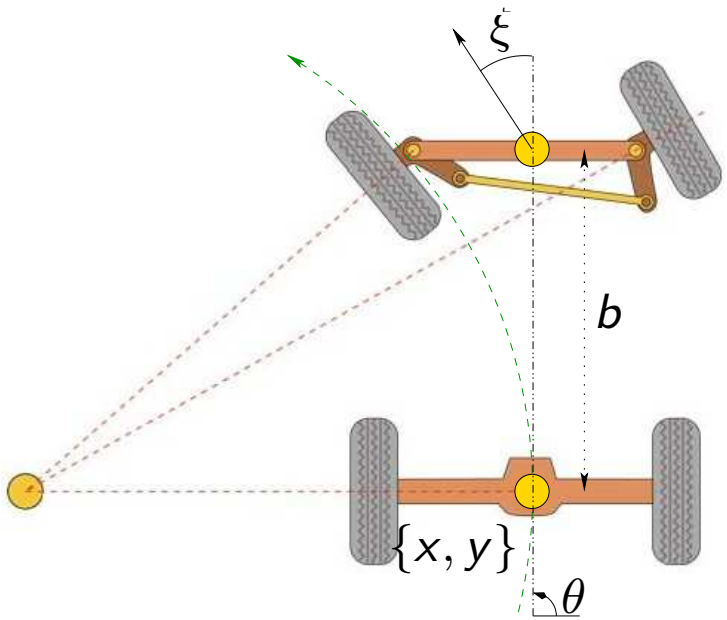
ICS-CHECK: a Case Study

Robotic system $\mathcal{A}$: car-like vehicle

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ v\tan\xi/b \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \gamma$$

$s = (x, y, \theta, v, \xi)$
$\tilde{u} = (\alpha, \gamma)$

$|v| \leq v_{\max}, |\xi| \leq \xi_{\max}$
$|\alpha| \leq \alpha_{\max}, |\gamma| \leq \gamma_{\max}$



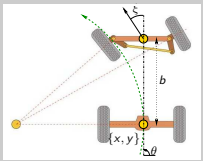
ICS-CHECK: a Case Study

Robotic system $\mathcal{A}$: car-like vehicle

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ v\tan\xi/b \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \gamma$$

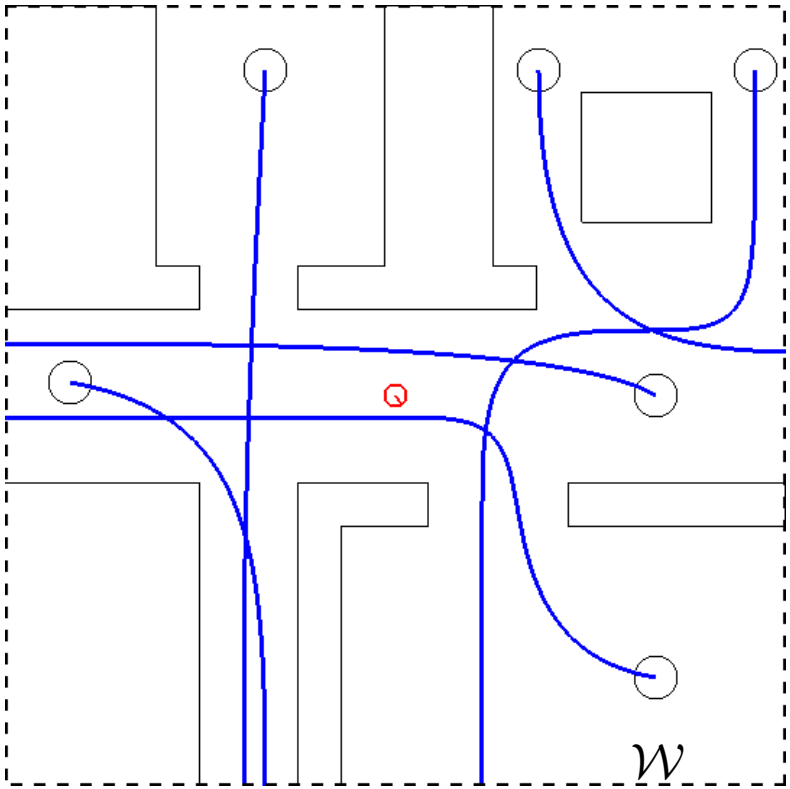
$s = (x, y, \theta, v, \xi)$
$\tilde{u} = (\alpha, \gamma)$

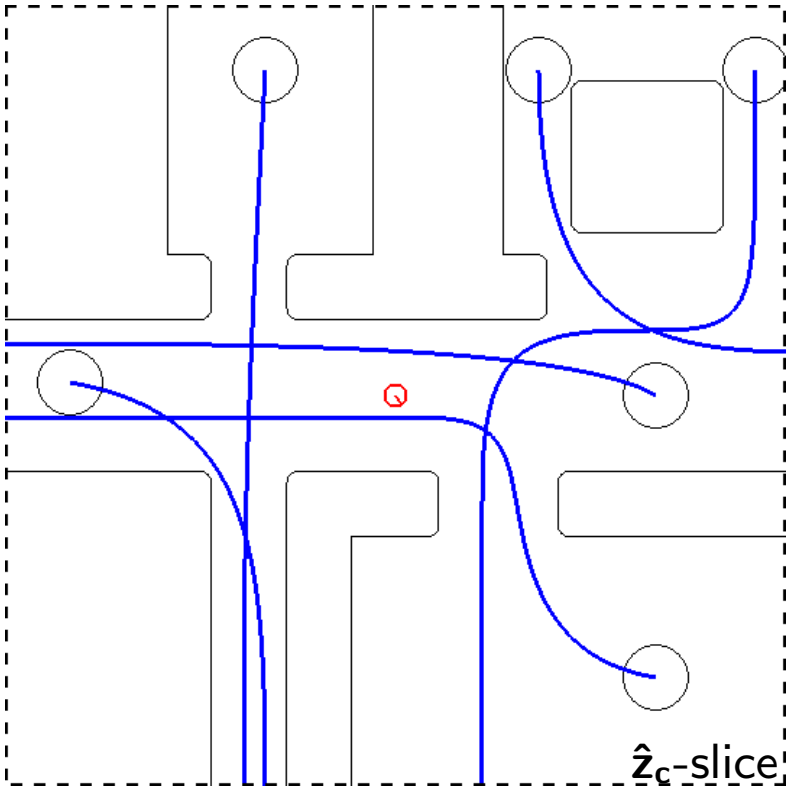
$|v| \leq v_{\max}, |\xi| \leq \xi_{\max}$
$|\alpha| \leq \alpha_{\max}, |\gamma| \leq \gamma_{\max}$



- A

Scenario

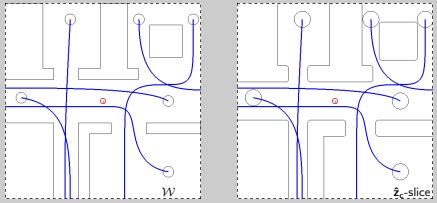




$s_c = (5, -1, -1, 10, 0)$ ICS?

Compute ICS set for $\hat{\mathbf{z}}_\mathbf{c}$-slice with $\hat{\mathbf{z}}_\mathbf{c} = (-1, 10, 0)$

Scenario



$s_c = (5, -1, -1, 10, 0)$ ICS?

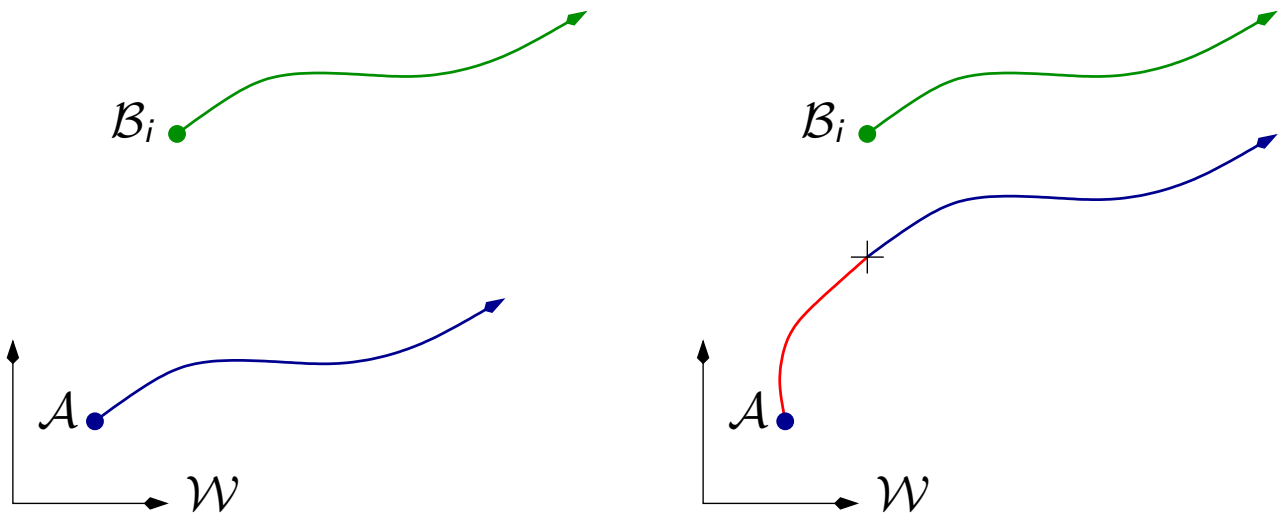
Compute ICS set for $\hat{\mathbf{z}}_c$-slice with $\hat{\mathbf{z}}_c = (-1, 10, 0)$

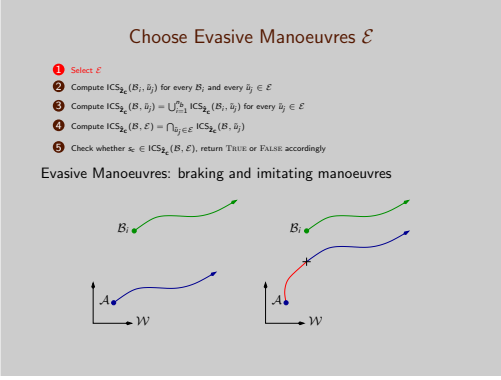
- A

Choose Evasive Manoeuvres $\mathcal{E}$

1. Select $\mathcal{E}$
2. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly

Evasive Manoeuvres: braking and imitating manoeuvres

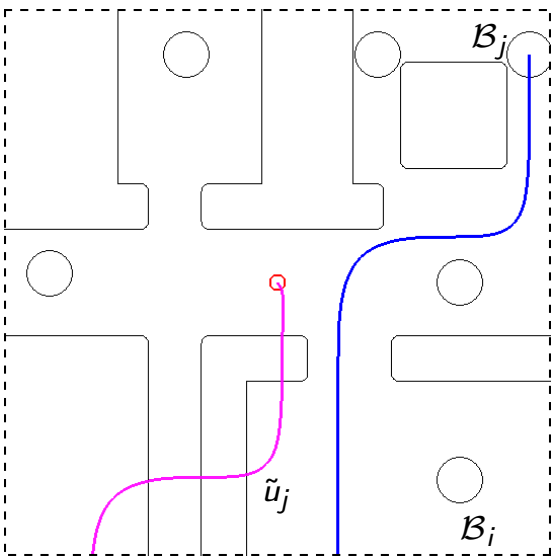


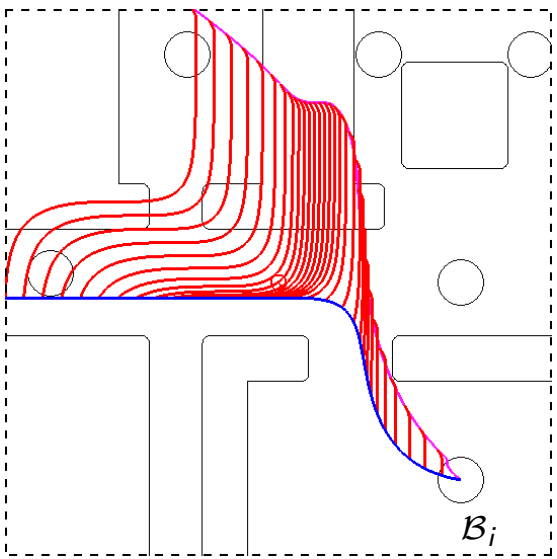

Choose Evasive Manoeuvres $\mathcal{E}$
1 Select $\mathcal{E}$
2 Compute $\mathrm{ICS}_{\mathbf{z}_c}(\mathcal{B}_i, \bar{u}_j)$ for every $\mathcal{B}_i$ and every $\bar{u}_j \in \mathcal{E}$
3 Compute $\mathrm{ICS}_{\mathbf{z}_c}(\mathcal{B}, \bar{u}_j) = \bigcup_{i=1}^{n_b} \mathrm{ICS}_{\mathbf{z}_c}(\mathcal{B}_i, \bar{u}_j)$ for every $\bar{u}_j \in \mathcal{E}$
4 Compute $\mathrm{ICS}_{\mathbf{z}_c}(\mathcal{B}, \mathcal{E}) = \bigcap_{\bar{u}_j \in \mathcal{E}} \mathrm{ICS}_{\mathbf{z}_c}(\mathcal{B}, \bar{u}_j)$
5 Check whether $\mathbf{z}_c \in \mathrm{ICS}_{\mathbf{z}_c}(\mathcal{B}, \mathcal{E})$, return True or False accordingly
Evasive Manoeuvres: braking and imitating manoeuvres
$\mathcal{B}_i$
$\mathcal{A}$
$\mathcal{W}$
$\mathcal{B}_i$
$\mathcal{A}$
$\mathcal{W}$

- A

ICS Set for Evasive Manoeuvre $\tilde{u}_j$

1. Select $\mathcal{E}$
2. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly

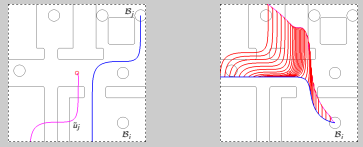




$$\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_t \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j, t)$$

ICS Set for Evasive Manoeuvre $\tilde{u}_j$

1. Select $\mathcal{E}$
2. Compute $\mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly

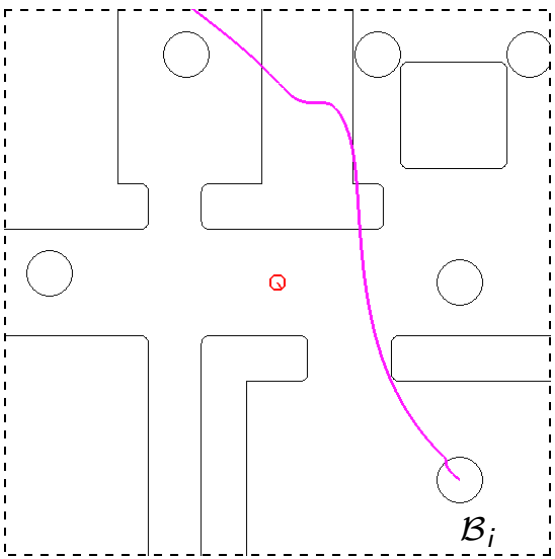


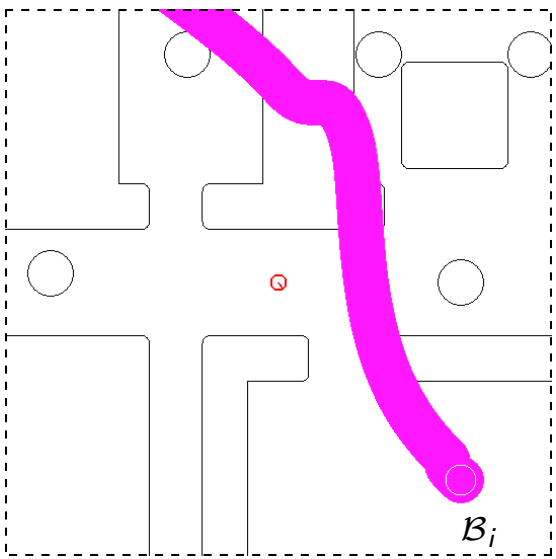
$\mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_t \mathsf{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j, t)$

- A

ICS Set for Evasive Manoeuvre $\tilde{u}_j$

1. Select $\mathcal{E}$
2. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly

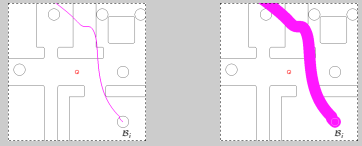




$$\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_t \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j, t)$$

ICS Set for Evasive Manoeuvre $\tilde{u}_j$

1. Select $\mathcal{E}$
2. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $\hat{\mathbf{z}}_c \in \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly

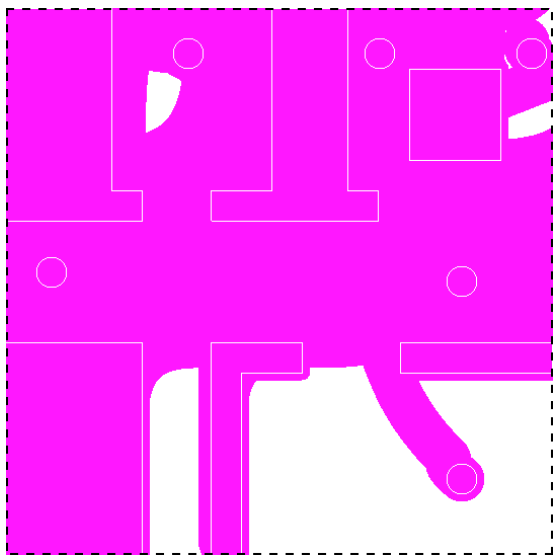


$\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_t \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j, t)$

- A

ICS Set for Evasive Manoeuvre $\tilde{u}_j$

1. Select $\mathcal{E}$
2. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly



$$\mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_i \mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_i \bigcup_t \mathrm{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j, t)$$

ICS Set for Evasive Manoeuvre $\tilde{u}_j$

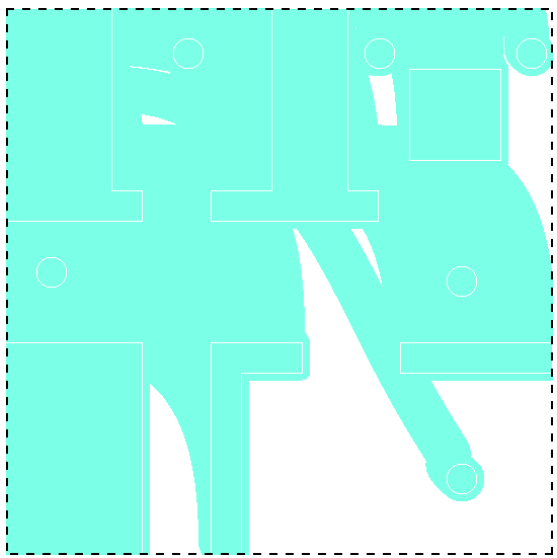
1. Select $\mathcal{E}$
2. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$, return True or False accordingly

$$\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j) = \bigcup_i \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_i \bigcup_t \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j, t)$$

- A

ICS Set for Evasive Manoeuvre $\tilde{u}_k$

1. Select $\mathcal{E}$
2. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly



$$\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_k)$$

ICS Set for Evasive Manoeuvre $\tilde{u}_k$

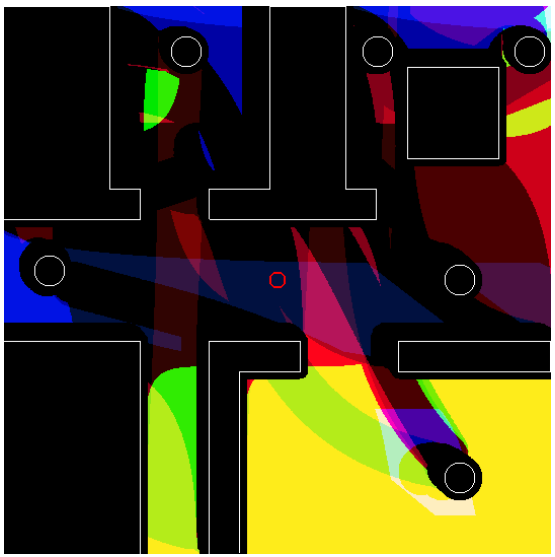
1. Select $\mathcal{E}$
2. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$, return True or False accordingly

$\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_k)$

- A

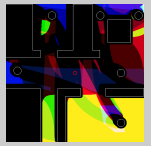
Final ICS Set

1. Select $\mathcal{E}$
2. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_b} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $s_c \in \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE accordingly



$$\text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}) = \bigcap_j \text{ICS}_{\hat{\mathbf{z}}_\mathbf{c}}(\mathcal{B}, \tilde{u}_j)$$
$$\rightsquigarrow s_c \text{ not ICS}$$

Final ICS Set
1. Select $\mathcal{E}$
2. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$
3. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1}^{n_{\mathcal{B}}} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}_i, \tilde{u}_j)$ for every $\tilde{u}_j \in \mathcal{E}$
4. Compute $\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j)$
5. Check whether $\mathbf{s}_c \in \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \mathcal{E})$, return True or False accordingly



$\mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}) = \bigcap_j \mathrm{ICS}_{\hat{\mathbf{z}}_c}(\mathcal{B}, \tilde{u}_j)$
$\rightsquigarrow \mathbf{s}_c$ not ICS

- A

ICS-CHECK Summary

ICS-CHECK is a **generic** and **efficient** ICS-checking algorithm

Generic: arbitrary planar systems, *e.g.*, point mass (4D), car-like (5D), differential drive (5D), spaceship (6D)

Efficient: 2D reasoning + GPU Implementation

Table: ICS-CHECK Performance evaluation

n_b	Time (ms)
13	28.4
14	30.9
15	32.4
16	35.7
17	39.10

First application: **navigation scheme**

- A

ICS-CHECK Summary

ICS-CHECK is a **generic** and **efficient** ICS-checking algorithm

Generic: arbitrary planar systems, *e.g.*, point mass (4D), car-like (5D), differential drive (5D), spaceship (6D)

Efficient: 2D reasoning + GPU Implementation

Table: ICS-CHECK Performance evaluation

n_b	Time (ms)
13	28.4
14	30.9
15	32.4
16	35.7
17	39.10

First application: **navigation scheme**

ICS-AVOID, an ICS-Based Navigation Scheme

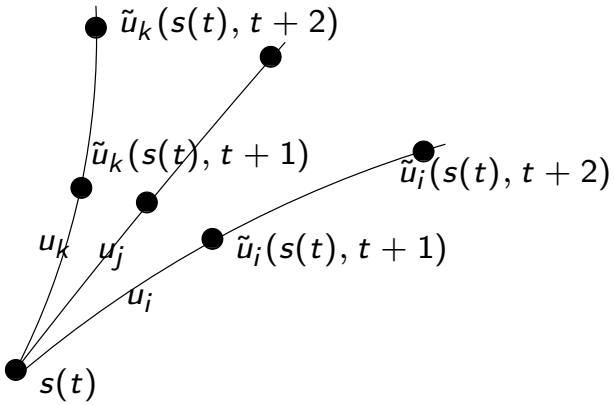
ICS-AVOID

Purpose of ICS-Avoid:

$\forall t,$ compute $u(t)$ such that $s(t+1)$ is not an ICS

Principle:

Sample $\mathcal{U}$ & use ICS-CHECK



Key Feature:

Sampling scheme that **guarantees motion safety**

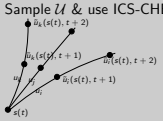
ICS-AVOID

Purpose of ICS-Avoid:

$\forall t$, compute $u(t)$ such that $s(t+1)$ is not an ICS

Principle:

Sample $\mathcal{U}$ & use ICS-CHECK



Key Feature:

Sampling scheme that **guarantees motion safety**

- A

Safe Control Kernel

ICS-Check:

$s(t) \notin \text{ICS}(\mathcal{B}, \mathcal{E}) \Rightarrow \exists$ a collision free $\tilde{u}_j$ in $\mathcal{E}$ (Evasive Manoeuvers)

Build the Safe Control Kernel $K(t)$:

Including first control of every $\tilde{u}_j \in \mathcal{E}$ in $K(t)$

$\rightsquigarrow$ at least one safe control will be in $K(t)$

Sample $\mathcal{U}$:

Include $K(t)$ + sampling strategy

ICS-Check:

$s(t) \notin \mathrm{ICS}(\mathcal{B}, \mathcal{E}) \Rightarrow \exists$ a collision free $\tilde{u}_j$ in $\mathcal{E}$ (Evasive Manoeuvers)

Build the Safe Control Kernel $K(t)$:

Including first control of every $\tilde{u}_j \in \mathcal{E}$ in $K(t)$

$\rightsquigarrow$ at least one safe control will be in $K(t)$

Sample $\mathcal{U}$:

Include $K(t)$ + sampling strategy

- A

Basic ICS-AVOID Algorithm

- Survive is the objective $\Rightarrow$ no interest in going somewhere.

Input: $s(t), \mathcal{E}$
Output: u

1. Compute Safe Control Kernel $K(t) = \bigcup_j \tilde{u}_j(t)$, $\tilde{u}_j \in \mathcal{E}$
2. Build control space sampling set: $\mathcal{J} \subset \mathcal{U}$ and include $K(t)$ into it
3. Select control $u \in \mathcal{J}$
4. Compute $s(t+1) = s(t) + \int_t^{t+1} f(s(t), u)dt$
5. If ICS-Check($s(t+1)$)= True. Go to 3
6. If ICS-Check($s(t+1)$)= False. SUCCESS. Return u.

Other sampling strategies can be employed: target heading, clearance and velocity

- A

Basic ICS-AVOID Algorithm

- Survive is the objective $\Rightarrow$ no interest in going somewhere.

Input: $s(t), \mathcal{E}$
Output: u

1. Compute Safe Control Kernel $K(t) = \bigcup_j \bar{u}_j(t)$, $\bar{u}_j \in \mathcal{E}$
2. Build control space sampling set: $\mathcal{J} \subset \mathcal{U}$ and include $K(t)$ into it
3. Select control $u \in \mathcal{J}$
4. Compute $s(t+1) = s(t) + \int_t^{t+1} f(s(t), u)dt$
5. If ICS-Check($s(t+1)$)= True. Go to 3
6. If ICS-Check($s(t+1)$)= False. SUCCESS. Return u.

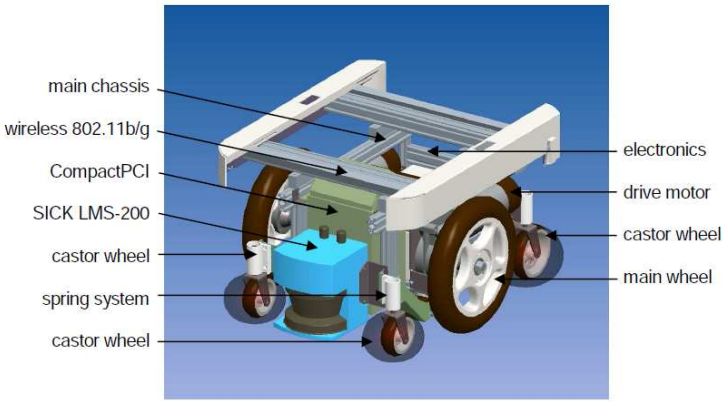
Other sampling strategies can be employed: target heading, clearance and velocity

Experimental and Simulation Results

- We present some of the results in simulation and experimental

Experimental results

Robotic platform:



Environment:

Robotic platform:

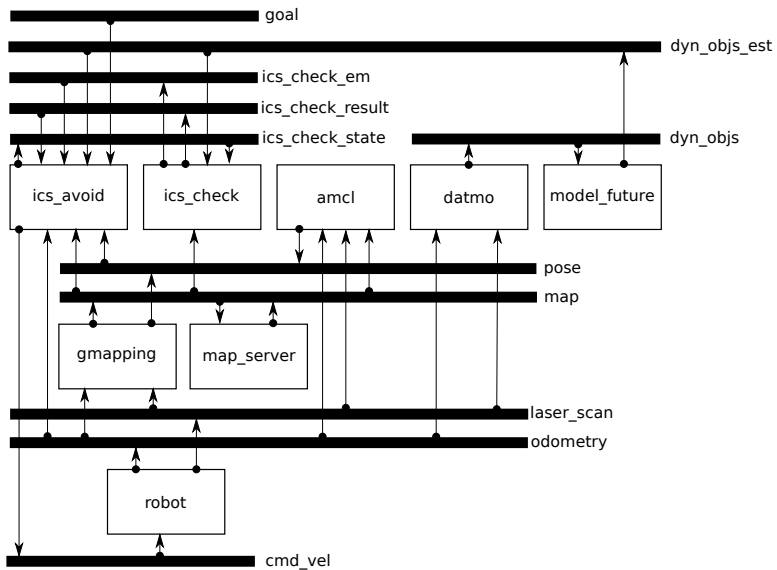
Environment:

- A

Software Architecture

ROS-based (Robotic Open Source - Willow Garage)

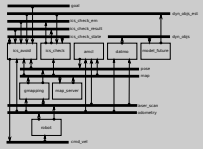
- Model of the environment:
  - SLAM
  - DATMO
- Decision:
  - ICS-AVOID
  - ICS-CHECK
- Execution:
  - Robotic Wheelchair



Software Architecture

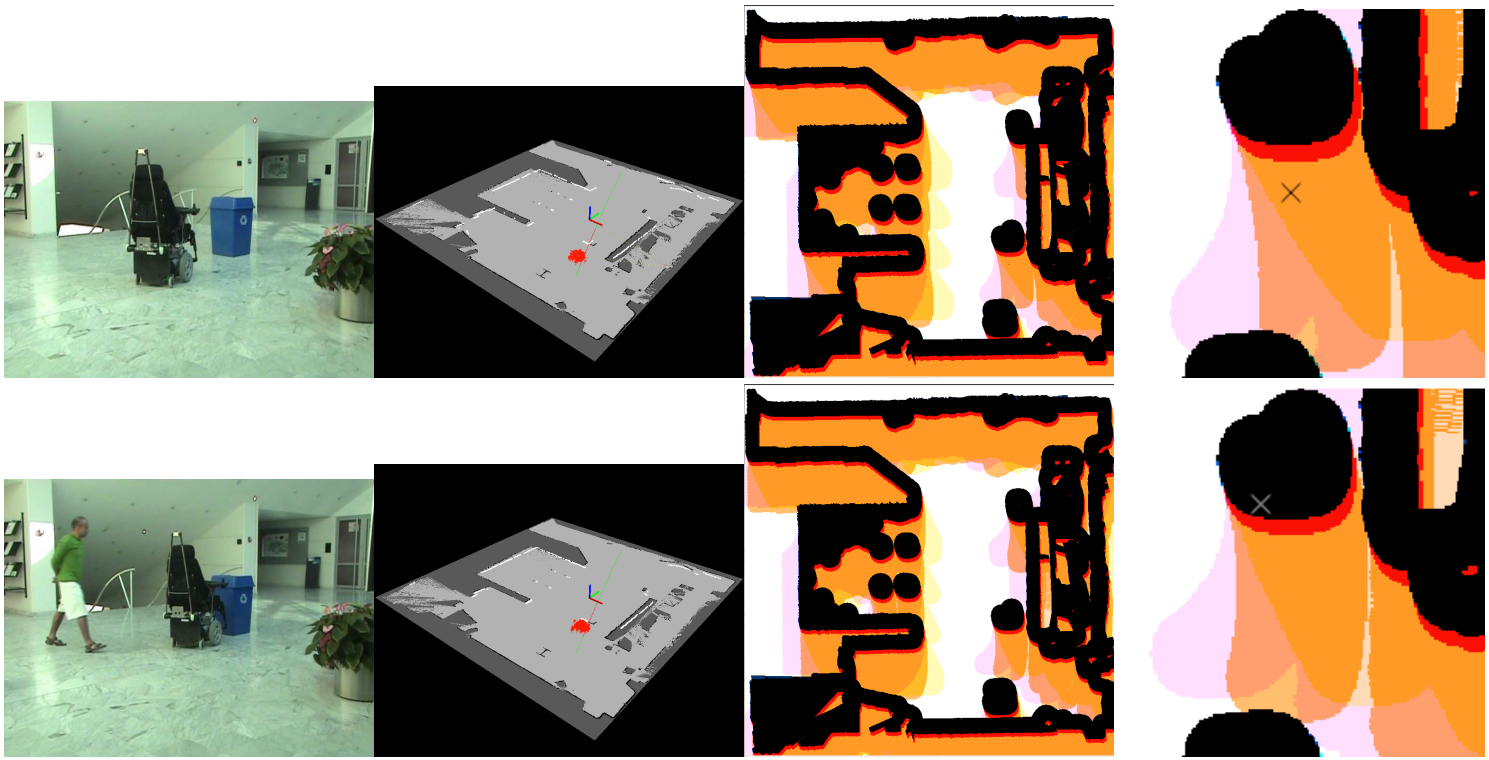
ROS-based (Robotic Open Source - Willow Garage)

- Model of the environment:
  - SLAM
  - DATMO
- Decision:
  - ICS-AVOID
  - ICS-CHECK
- Execution:
  - Robotic Wheelchair



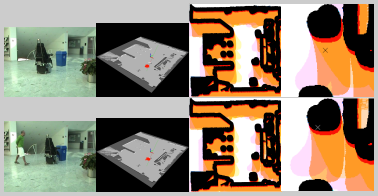
- A

ICS-CHECK Experiment



Video

ICS-CHECK Experiment



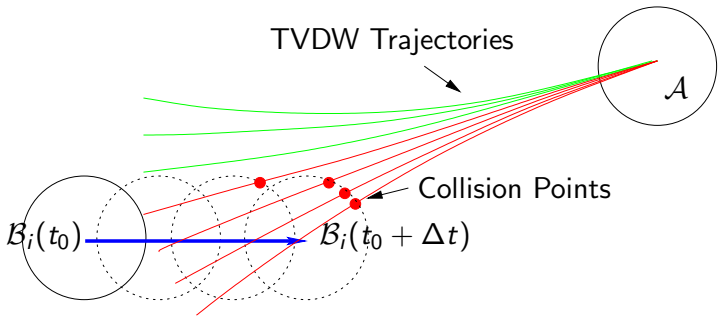
Video

- A

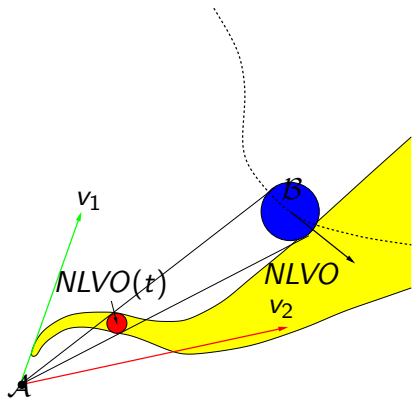
ICS-AVOID Benchmarking

Benchmark with state-of-the-art collision avoidance schemes:

- TVDW *[Seder & Petrovic 07]* t_{la}: braking time

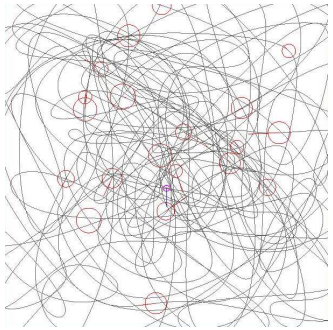


- NLVO *[Large & Shiller 05]* t_{la}: arbitrary

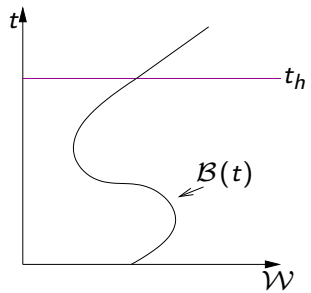


Scenario benchmarking

$\mathcal{W}$: $100m^2$
23 objects $\mathcal{B}_i$, cyclic trajectories



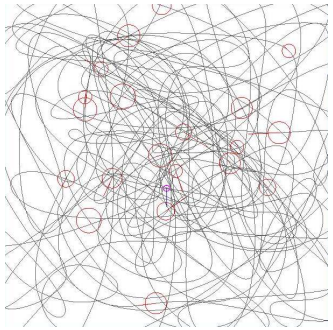
Model of the future with t_h:
Below t_h - Full knowledge
Above t_h - Linear behaviour



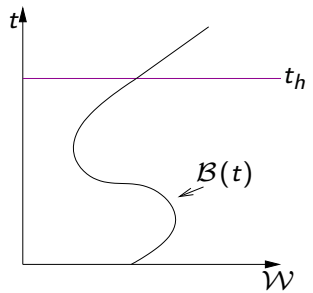
10 min runs Video

Scenario benchmarking

$\mathcal{W}$: $100m^2$
23 objects $\mathcal{B}_i$, cyclic trajectories



Model of the future with t_h:
Below t_h - Full knowledge
Above t_h - Linear behaviour



10 min runs Video

Results:

t_h	TVDW	NLVO	ICS-AVOID
$1s$	9.2	8.0	2
$3s$	4.2	2.2	**0**
$5s$	3.6	0.8	**0**

Scenario benchmarking

$\mathcal{W}$: $100m^2$
23 objects $\mathcal{B}_i$, cyclic trajectories

Model of the future with t_h:
Below t_h - Full knowledge
Above t_h - Linear behaviour

10 min runs Video

Results:

t_h	TVDW	NLVO	ICS-AVOID
1s	9.2	8.0	2
3s	4.2	2.2	**0**
5s	3.6	0.8	**0**

- A

Conclusions and Perspectives

Conclusions

Motion Safety Problem:

- **Inevitable Collision States** provides a good answer
- 3 safety criteria:
  1. Reason about the **future**
  2. Appropriate **lookahead**
  3. **Decision time constraint**

Contribution of this work:

- Efficient Algorithm for the characterisation of ICS.
- Incorporation in a navigation scheme: ICS-AVOID

Motion Safety Problem:

- **Inevitable Collision States** provides a good answer
- 3 safety criteria:
  1. Reason about the **future**
  2. Appropriate **lookahead**
  3. **Decision time constraint**

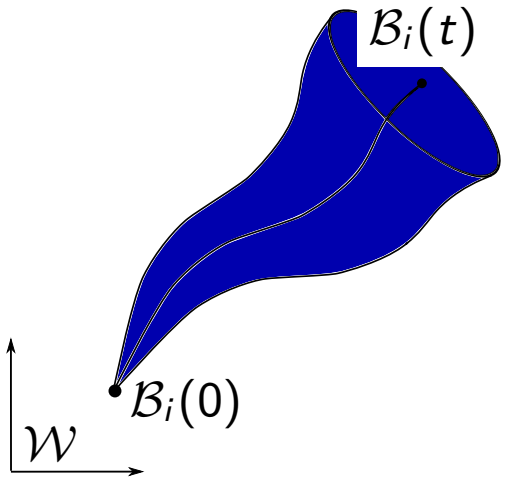
Contribution of this work:

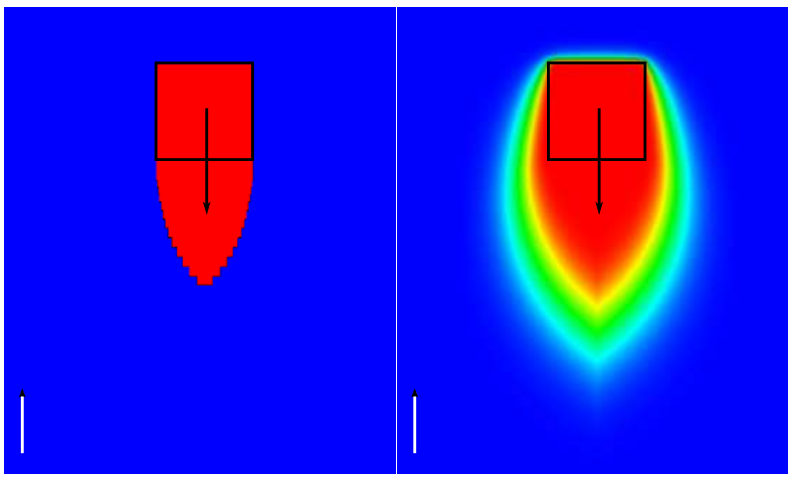
- Efficient Algorithm for the characterisation of ICS.
- Incorporation in a navigation scheme: ICS-AVOID

- A

Perspectives

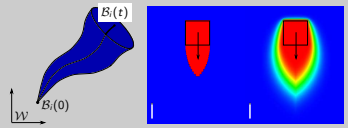
- ICS-CHECK:
  - Model of the future.
  - **Probabilistic ICS** *[Bautin & Martinez 09]*





- ICS-AVOID:
  - Guarantees of converging to a goal.

Perspectives
- ICS-CHECK:
  - Model of the future.
  - **Probabilistic ICS** *[Bautin & Martinez 09]*



- ICS-AVOID:
  - Guarantees of converging to a goal.

- A

Thank you!!