Automated Task and Motion Plan Generation
for Multi-Robot Systems from Complex Specifications

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
How do you make the programming of Mainstream Robots easier?
Research Goal

To develop a programming framework and runtime system for multi-robot application

- Scales up for many robots
- Deals with complex specifications
- Deals with complex dynamics of robots
Example

- (Functional spec) Pick up items from $X$ and bring them to $O$
- (Safety spec) Avoid colliding with obstacles and other robots
Example

- (Functional spec) Pick up items from X and bring them to O
- (Safety spec) Avoid colliding with obstacles and other robots

- (functional spec) Pick $p_1$ and $p_2$, but once picked, drop the object at $d_1$ or $d_2$ respectively, before picking anything else

- (functional spec) Measure sensor values at locations $m_1$, $m_2$, and $m_3$ simultaneously, and report the result at one of the report locations $g_1$, $g_2$, $g_3$
State-of-the-art

- **Multi-robot system**, but a user has to reference robots individually

- **Plan for robots**, but the dynamic constraints of the robots are often ignored

- **End-to-end guarantees**, but under the (unrealistic) assumption of a perfect world
What do we need?

- A multi-robot management system that takes care of “robotics infrastructure” and hides the details from a user
- A multi-robot motion planner that respects the dynamic constraints of the robots
- A multi-robot runtime system that can take care of some issues during execution (Separation of concerns)
Antlab: A Multi-Robot Task Server

Antlab: A Multi-Robot Task Server

Declarative task

Antlab

Task & Path Planner
Should be expressive to capture temporal relationships among the events

**Example:** Visit area $R_2$, then area $R_3$, then area $R_4$, and finally, return and remain in region $R_1$ while avoiding areas $R_2$ and $R_3$
**Linear Temporal Logic (LTL)**

**LTL Grammar:**

\[ \phi ::= \pi | \neg \phi | \phi \land \phi | \diamond \phi | \square \phi | \lozenge \phi | \phi U \phi \]

\(\pi\) - set of atomic propositions

Example: \(\pi_1\) - The robot is in Room 1

(next) \[\diamond \phi \rightarrow \phi\]

(always) \[\square \phi \rightarrow \phi \rightarrow \phi \rightarrow \phi \rightarrow \phi \rightarrow \phi\]

(eventually) \[\lozenge \phi \rightarrow \phi\]

(until) \[\phi_1 U \phi_2 \rightarrow \phi_1 \rightarrow \phi_1 \rightarrow \phi_1 \rightarrow \phi_2\]
Examples of LTL Specifications

1. **Reachability**
   \[ \varphi = \Diamond \pi_2 \]

2. **Coverage**
   \[ \varphi = \Diamond \pi_2 \land \Diamond \pi_3 \land \Diamond \pi_4 \]

3. **Sequencing**
   \[ \varphi = \Diamond(\pi_2 \land \Diamond \pi_3) \]

4. **Reachability with avoidance**
   \[ \varphi = (\neg \pi_2 \land \neg \pi_3) \mathcal{U} \pi_4 \]

5. **Recurrent sequencing**
   \[ \varphi = \square \Diamond(\pi_2 \land \Diamond \pi_3) \]

Visit area \( R_2 \), then area \( R_3 \), then area \( R_4 \), and finally, return and remain in region \( R_1 \) while avoiding areas \( R_2 \) and \( R_3 \)

\[ \varphi = \Diamond(\pi_2 \land \Diamond(\pi_3 \land \Diamond(\pi_4 \land (\neg \pi_2 \land \neg \pi_3) \mathcal{U} \square \pi_1)))) \]
How do we capture the following requirements?

- Any of the available robots can satisfy a specification $\psi$
- All robots should satisfy a specification $\psi$

Extend the basic logic by allow outermost quantifiers over the set of robot identifiers

$$\psi ::= \varphi \mid \exists r.\psi \mid \forall r.\psi$$

In a model with a fixed set of robots, the quantifiers can be de-sugared into *disjunctions* or *conjunctions* over the set of robots.
Antlab: A Multi-Robot Task Server

Declarative task

Antlab

Task & Path Planner
Antlab: A Multi-Robot Task Server

Declarative Requests

Server

Antlab

System State

- Configuration Database
- Occupancy Grid

Runtime System

- Service Manager
  - Robot Manager
  - Task & Path Planner
  - Task Organizer

Plan

Robot

Task and motion planning for multi-robot systems
Plan Synthesis for Robots

- Specification (LTL formula)
- System Information (Robot Dynamics)
- Other inputs (workspace, initial state)

Plan Synthesis Tool

Program (Task and Motion Plan)
Existing Solutions for LTL Motion Planning

General Idea:

- Generate a finite abstraction for the robot dynamics
- Generate a finite model for the LTL specification
- Apply a game theoretic algorithm to generate a high level plan
- Generate low level control signals to realize the high level plan

Work by Kress-Gazit, Fainekos, Pappas, Karaman, Frazzoli, Kavraki, Verdi, Topcu, Murray, Belta, Rus and others..

Limitations:

- Discretization of continuous dynamics is expensive
- Existence of low-level controllers is not guaranteed
Motion Primitives

Short, kinematically feasible motions forming the basis of movements of the robot

Components:

- $u$ - a precomputed control input
- $\tau$ - the duration for which the control signal is applied
- $q_i$ - initial velocity configuration
- $q_f$ - final velocity configuration
- $X_{rf}$ - relative final position
- $W$ - the set of relative blocks through which the robot may pass
- \textit{cost} - an estimated energy consumption for executing the control law

Note: Motion Primitives are position oblivious
An input problem instance $\mathcal{P} = \langle R, I, PRIM, Workspace, \xi, L \rangle$

- $R$ - The set of robots
- $I$ - Initial state of the group of robots
- $PRIM = [PRIM_1, PRIM_2, \ldots, PRIM_{|R|}]$
- Workspace - Workspace dimension, position of obstacles
- $\xi$ - Specification given in Linear Temporal Logic
- $L$ - Number of hops in the trajectory

Definition (Motion Planning Problem)

Given an input problem $\mathcal{P}$, synthesize a trajectory of length $L$
Trajectory of a Multi-Robot System

State of a robot $i$: $\phi_i = \langle q, X \rangle$

- $q$ - Velocity configuration
- $X$ - Position

State of the multi-robot system:
$\phi = [\phi_1, \ldots, \phi_{|R|}]$

Input problem instance: $\mathcal{P} = \langle R, I, PRIM, Workspace, \xi, L \rangle$

Trajectory: sequence of states $\Phi = (\Phi(0), \Phi(1), \ldots, \Phi(L))$

$\Phi(0) \in I$

$\Phi(0) \xrightarrow{Prim_1} \Phi(1) \xrightarrow{Prim_2} \Phi(2) \ldots \Phi(L - 1) \xrightarrow{Prim_L} \Phi(L)$

where $Prim_i = [\gamma_{1i}, \ldots, \gamma_{|R|i}]$, where $\gamma_{ji} \in PRIM_j$
Program Synthesis - SMT Solver

Synthesis Tool

(SMT Solver)

Specification
System Information
Program
Program Synthesis - SMT Solver

Specification

System Information

Synthesis Tool

Program

(SMT Solver)

Z3

Yices

GCC

MathSAT
**SMT Solver**

**SAT solver:** Checks satisfiability of Boolean formulas

Example: $b_1 \land b_2 \land (b_2 \rightarrow \neg b_1)$

**SMT Solver:** SAT solver empowered with Theory solvers

Example Theories: LRA (Linear Real Arithmetic), EUF (Equality with Uninterpreted Functions), …

Example: $x_0 = 0 \land y_0 = 0 \land f(2, 1) = true \land$

$$((x_1 = x_0 + 2) \land (y_1 = y_0 + 1)) \lor ((x_1 = x_0 + 1) \land (y_1 = y_0 + 2)) \land$$

$$((x_2 = x_1 + 2) \land (y_2 = y_1 + 1)) \lor ((x_2 = x_1 + 1) \land (y_2 = y_1 + 2)) \land$$

$f(x_1, y_1) \neq true \land x_2 \geq 4 \land y_2 \geq 3$

Formula is satisfiable $\Rightarrow$ generates a **model**

Formula is unsatisfiable $\Rightarrow$ generates an **unsatisfiable core**

An SMT solver can be used as an **optimization engine**
- Iteratively search for better solution using binary search
Architecture of a Program Synthesis Tool

- Specification
- System Information
- Constraint Generator
- Constraints
- SMT Solver
- Unsat Core
- Model
- Specification Refinement Tool
- Program Generator
- Program
Complan\(^2\)
(COMpositional Motion PLANner)
http://www.cse.iitk.ac.in/~isaha/complan.shtml

\[
\begin{align*}
\Phi(0) & \xrightarrow{\text{Prim}_1} \Phi(1) \xrightarrow{\text{Prim}_2} \Phi(2) \ldots \Phi(L-1) \xrightarrow{\text{Prim}_L} \Phi(L)
\end{align*}
\]

Constraints:
\[(\Phi(0) \in I) \land \|\text{Transition}\| \land \|\text{Specification}\| \land \|\text{Cost}\|\]

Boolean combination of constraints from Linear Arithmetic and Equality with Uninterpreted Functions theories

Complan solves for the \(L\) motion primitives using an SMT solver

Transition Constraints

\[ \Phi_1 = [\phi_{11}, \ldots, \phi_{1N}], \Phi_2 = [\phi_{21}, \ldots, \phi_{2N}] \]

\[ Prim = [\gamma_1, \ldots, \gamma_N], \text{ where } \gamma_i \in PRIM_i. \]

A transition

\[ \Phi_1 \xrightarrow{Prim} \Phi_2 \]

is associated with the following constraints:

- \( \forall i \in \{1, \ldots, N\} : \phi_{1i}.q = \gamma_i.q_i \)
- \( \forall i \in \{1, \ldots, N\} : \phi_{2i}.q = \gamma_i.q_f \)
- \( \forall i \in \{1, \ldots, N\} : \phi_{2i}.X = \phi_{1i}.X + \gamma_i.X_{rf} \)
- \textit{obstacle avoidance}(\Phi_1, \Phi_2, Prim, OBS)
- \textit{collision avoidance}(\Phi_1, \Phi_2, Prim)

Collision avoidance constraints are expensive
- ignored during plan generation
Trajectory:

\[ \Phi(0) \xrightarrow{Prim_1} \Phi(1) \xrightarrow{Prim_2} \Phi(2) \ldots \Phi(L-1) \xrightarrow{Prim_L} \Phi(L) \]

Specification: \( \varphi \cup \psi \)

The Specification Constraints:

- \( \Phi(L) \models \psi \)
- \( \forall i \in \{0, \ldots, (L-1)\} : \Phi(i) \models \varphi \)
Cost Constraints

Trajectory:

\[ \Phi(0) \xrightarrow{\text{Prim}_1} \Phi(1) \xrightarrow{\text{Prim}_2} \Phi(2) \ldots \Phi(L - 1) \xrightarrow{\text{Prim}_L} \Phi(L) \]

where \(\text{Prim}_i = [\gamma_{1i}, \ldots, \gamma_{|R|i}]\), where \(\gamma_{ji} \in \text{PRIM}_j\)

Cost of the trajectory:

\[
\text{cost}(\Phi(0) \ldots, \Phi(L)) = \sum_{t=1}^{L} \sum_{r \in |R|} \text{cost}(\gamma(r)(t))
\]

Cost constraint:

\[
\text{cost}(\Phi(0), \ldots, \Phi(L)) < \text{cost\_bound}
\]
Example: Multi-Robot Motion Planning

Goal: $I_1 \rightarrow F_1$, $I_2 \rightarrow F_2$, $I_3 \rightarrow F_3$, $I_4 \rightarrow F_4$

Requirements:
- Maintain a rectangular formation
- Maintain a precedence relationship
- Maintain a minimum distance
Example: Satisfy Invariants before Reaching Goal

**Goal:**
- \((I_1 \text{ and } I_2) \rightarrow B\)
- \((I_3 \text{ and } I_4) \rightarrow A\)

**Invariants:**
- Maintain a **rectangular** or **linear** formation
- Maintain a minimum distance
Example: Satisfy Invariants before Reaching Goal

Goal: (I1 and I2) → B
     (I3 and I4) → A

Invariants:
- Maintain a **rectangular or linear** formation
- Maintain a minimum distance

No motion plan that satisfies the formation constraint exists

Unsatisfiable Core helps us refining the specification
Example: Satisfy Invariants before Reaching Goal

**Goal:** (I1 and I2) $\rightarrow$ B
(I3 and I4) $\rightarrow$ A

**Invariants:**
- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
Example: Satisfy Invariants before Reaching Goal

Goal: \((I_1 \text{ and } I_2) \rightarrow B\)
\((I_3 \text{ and } I_4) \rightarrow A\)

Invariants:
- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
Finding Optimal Trajectory

- Find the minimum value of $L$ for which we can generate a valid trajectory
- Among all trajectories of length $L$, find the one that incurs the least cost
Example: Satisfy Invariants before Reaching Goal

Goal: (I1 and I2) $\rightarrow$ B
(I3 and I4) $\rightarrow$ A

Invariants:
- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
Example: Satisfy Invariants before Reaching Goal

**Goal:**

(I1 and I2) $\rightarrow$ B  
(I3 and I4) $\rightarrow$ A

**Invariants:**

- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
Dealing with Arbitrary LTL property

An infinite trajectory satisfying an LTL formula can be represented as a finite trace without or with a loop.

\[ \Phi(0) \rightarrow \Phi(L+1) \rightarrow \Phi(0) \rightarrow \Phi(k) \rightarrow \Phi(L+1) \]

Examples: \( \Diamond \xi, \xi_1 U \xi_2 \)  \( \Box \Diamond \xi, \Box \Diamond (\xi_1 \land \Diamond \xi_2) \)

Systematic generation of constraints for full LTL syntax

Example: \( [\xi_1 U \xi_2]_i \equiv [\xi_2]_i \lor ([\xi_1]_i \land [\xi_1 U \xi_2]_{i+1}) \)

Two new variables to deal with loops:

- **loopexists**: Boolean variable that is true if the trajectory is of infinite length
- **loopbeginning**: Integer variable indicating where the loop begins if loopexists = true

and associated constraints
Each quadrotor has to repeatedly gather data from some data gathering location and upload the gathered data at a data upload location.

\[ \xi_1 := \forall r. (\Box ((r_{X\text{gather}} \land (\Diamond r_{X\text{upload}})))) \]
Each quadrotor has to repeatedly gather data from some data gathering location and upload the gathered data at a data upload location.

\[ \xi_1 := \forall r. (\Box (rX_{gather} \land (\Diamond rX_{upload}))) \]
Specification: $\neg \text{obstacles} \cup \text{reach}$
Motion Plan Synthesis for a Swarm of Robots

Specification: $\neg obstacles \cup reach$

Main Idea:
- Synthesize optimal trajectory for each robot independently
Specification: $\neg obstacles \cup reach$

Main Idea:
- Synthesize optimal trajectory for each robot independently
Motion Plan Synthesis for a Swarm of Robots

Specification: $\neg \text{obstacles} \cup \text{reach}$

Main Idea:
- Synthesize optimal trajectory for each robot independently
- Find a feasible ordering for the robots
Motion Plan Synthesis for a Swarm of Robots

**Specification:** \( \neg obstacles U reach \)

**Main Idea:**
- Synthesize optimal trajectory for each robot independently
- Find a feasible ordering for the robots
- Synthesize final trajectories according to the assigned priorities
  - Treat the robots with higher priorities as dynamic obstacles
  - Introduce minimum delay to execute the optional trajectory to avoid collision
Motion Plan Synthesis for a Swarm of Robots

Specification: $\neg obstacles \cup reach$

**Implan**$^3$

(Incremental Motion PLANner)

**Main Idea:**

- Synthesize optimal trajectory for each robot independently
- Find a feasible ordering for the robots
- Synthesize final trajectories according to the assigned priorities
  - Treat the robots with higher priorities as dynamic obstacles
  - Introduce minimum delay to execute the optional trajectory to avoid collision

---

Motion Plan Synthesis for a Swarm of Robots

Ordering:
1: $R_3$  
2: $R_6$  
3: $R_2$  
4: $R_4$  
5: $R_1$  
6: $R_5$
Finding Robot Ordering

Based on a priority assignment algorithm

**Priority Assignment:**

Input: Optimal trajectories of the robots, the initial and final positions of the robot

Output: A ordering that allows all robots to reach their destination

Algorithm:

- If the initial location of Robot $q$ blocks the optimal trajectory of Robot $p$:
  \[ \text{prio}(q) > \text{prio}(p) \]
- If the final location of Robot $q$ blocks the optimal trajectory of Robot $p$:
  \[ \text{prio}(q) < \text{prio}(p) \]
- Solve the constraints using an SMT solver
- If the solver generates an unsat core, assign same priority to all the robots involved in unsat core

\[ \text{prio}(R2) > \text{prio}(R1) \]
\[ \text{prio}(R1) > \text{prio}(R2) \]
Example: Motion Planning in a Compact Workspace

25 quadrotors moving in a closed place

Specification:
\( \neg \text{obstacles} \cup \text{reach} \)
Dynamism in the environment may invalidate synthesized paths

- dynamic obstacles
- potential collisions with other robots

Potential inter-robot collisions are resolved with the help of a local collision avoidance protocol

In cases where local motion planning and obstacle avoidance cannot find a feasible plan, a recovery behavior is triggered
Is Joint Task and Path Planning Necessary?

Figure: arena: shoreline
Figure: arena: artificial floor
Figure: arena: maze

<table>
<thead>
<tr>
<th>Arena name</th>
<th>Planning Time (sec)</th>
<th>Plan Execution Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shoreline</td>
<td>floor</td>
</tr>
<tr>
<td>Heuristic assignment</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SMT-based assignment and planning</td>
<td>11.56</td>
<td>44.63</td>
</tr>
</tbody>
</table>

Table: Effect of joint task assignment and planning
Response Time with Concurrency

Figure: Execution time vs number of concurrent batches for a system with 8 robots

Response time depends on several factors including

- Number of available robots
- Nature of the workspace
Looking Ahead
Enhancing Scalability of Planning Algorithms and Expressiveness of Specifications

- Tens of robots each having their own complex LTL specification

- More expressive logic
  
  e.g. timing specification
  
  Robot a should visit region B within 10 sec of visiting region A
**Scenario:** A group of robots have a search a partially known environment
- boundaries are known
- obstacle locations are unknown
- points of interest are unknown

**Approach:** Receding horizon exploration

**Challenge:**
- How to assign responsibilities to the robots in an optimal way?
- How many robots should be active at any point of time?
**Goal:** Devise tools and techniques for synthesizing motion planner that is

- **Reactive**
  
  self-driven car, robocup

- **Robust**
  
  a drone flying in the presence of gust of wind

- **Distributed**
  
  autonomous delivery drones

**Challenges:**

- How to make existing solvers more amenable to solve planning problems?
- How to build domain specific solvers? (e.g., a solver that has control theoretic intelligence)
Thank You!!

http://www.cse.iitk.ac.in/~isaha