COMP 650 “Thesis” Presentation

RAM and IJRR/RSS TMKit Papers
Neil T. Dantam et. al., Rice University, 2016

Presented by Cannon Lewis

Images from original papers

Identified Problems

- Task and Motion Planning (TMP) presents challenges in completeness and scalability; existing approaches sacrifice one or the other.
- Existing TMP solutions are specific to certain classes of problems.
- Task planners and motion planners operate on very different principles, so achieving interaction between these layers is difficult.
- Motion planning for complex manipulation tasks requires a non-static configuration space representation.

Specific Contributions

- Useful abstractions of the task planning, motion planning, and TMP problems.
- TMKit, a modular system for performing interleaved TMP.
- The IDTMP algorithm for efficiently combining task planning and motion planning.
- The Amino scene-graph representation which allows for efficient configuration space updates.

Background - Task Planning

Definition 1. Task Language. *The task language is a set of strings of actions, defined by* $\mathfrak{L} = (\mathcal{P}, \mathcal{A}, \mathcal{E}, s^{[0]}, \mathcal{G})$*, where,*

- $\mathcal{P}$ *is the state space ranging over variables* $p_0, \dots, p_n$*,*
- $\mathcal{A}$ *is the set of task operators, i.e., terminal symbols,*
- $\mathcal{E} \subseteq (\mathbb{P}(\mathcal{P}) \times \mathcal{A} \times \mathbb{P}(\mathcal{P}))$ *is the set of* symbolic transitions. *Each* $e_i \in \mathcal{E}$ *denotes transitions* $\mathrm{pre}(a_i) \xrightarrow{a_i} \mathrm{eff}(a_i)$*, where* $\mathrm{pre}(a_i) \subseteq \mathcal{P}$ *is the precondition set,* $a_i \in \mathcal{A}$ *is the operator, and* $\mathrm{eff}(a_i) \subseteq \mathcal{P}$ *is the effect set. We represent a concrete transition on* a_i *at step* k *from state* $s^{[k]}$ *to* $s^{[k+1]}$ *as* $s^{[k+1]} = a_i(s^{[k]})$*, where* $s^{[k]} \in \mathrm{pre}(a_i)$*,* $s^{[k+1]} \in \mathrm{eff}(a_i)$*, and for all state variables* p_j *independent of (i.e., free in)* $\mathrm{eff}(a_i)$*,* ${p_j}^{[k+1]} = {p_j}^{[k]}$*,*
- $s^{[0]} \in \mathcal{P}$ *is the start state,*
- $\mathcal{G} \subseteq \mathcal{P}$ *is the set of accept states, i.e., the task goal.*

Definition 2. Task Plan. *A* task plan $\mathbf{A}$ *is a string in the task language* $\mathfrak{L}$*, i.e.,* $\mathbf{A} \in \mathfrak{L}$*, where* $\mathbf{A} = (a^{[0]}, a^{[1]}, \dots, a^{[h]})$*,* $a^{[k]} \in \mathcal{A}$*,* $s^{[k]} \in \mathrm{pre}(a^{[k]})$*,* $s^{[k+1]} = a^{[k]}(s^{[k]})$*, and* $s^{[h]} \in \mathcal{G}$*.*

- Domain consists of discrete representation of task to be solved.
- Plan consists of discrete actions to transition from start state to goal state.

Background - Task Planning

- We would like to be able to generate alternate task plans, so that motion planner feedback can influence task plan choice.
- We would also like to be able to reuse work from previous planning rounds.
- Dominant approaches in task planning are heuristic search (e.g. A*) and constraint based methods (e.g. SMT).

Background - Task Planning

- Satisfiability Modulo Theories (SMT) is essentially SAT solving with additional rules (“theories”) for defining constraints in non-boolean domains.
- SMT effectively provides a high-level interface for SAT solving.
- Incremental solving with Z3 (pushing/popping constraints).
- Want to encode failed motion plans with constraints.

Background - Task Languages

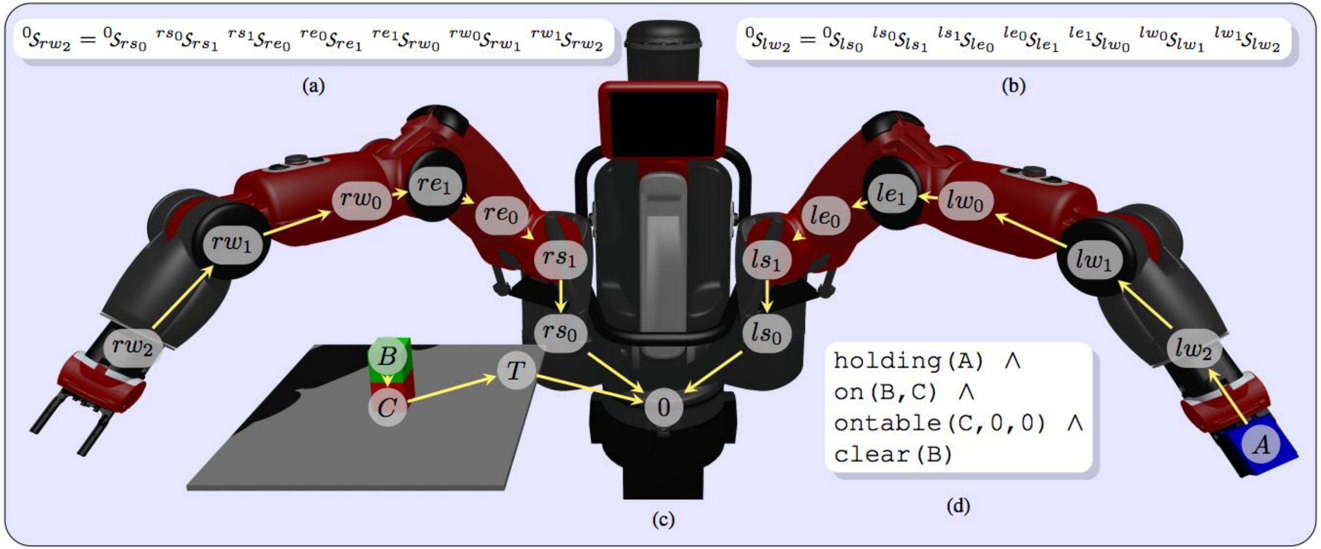
- Many task language notations exist.
- Most popular are PDDL, LTL, or special purpose formal languages.
- All notations roughly equivalent in expressibility.
- Ideal TMP solution is notation agnostic.
- In experiments for this paper, PDDL is used.

```
(define (domain transfer-linear)
  (:types block location - object)
  (:functions (position ?obj - block) - location)
  (:derived (occupied ?loc - location)
            (exists (?obj - block) (= (position ?obj)
                                        ?loc)))
  (:action transfer
           :parameters (?obj - block
                        ?dst - location)
           :precondition (not (occupied ?dst))
           :effect (and (= (position ?obj)
                            ?dst))))
```

Background - Motion Planning

- Robots typically modeled as kinematic trees.
- For task and motion planning, need to be able to modify kinematic tree as objects are manipulated.
- Sampling based planners are the dominant method for high DoF motion planning.
- Probabilistic completeness a consideration for TMP.
- For this work, RRT-Connect is used.

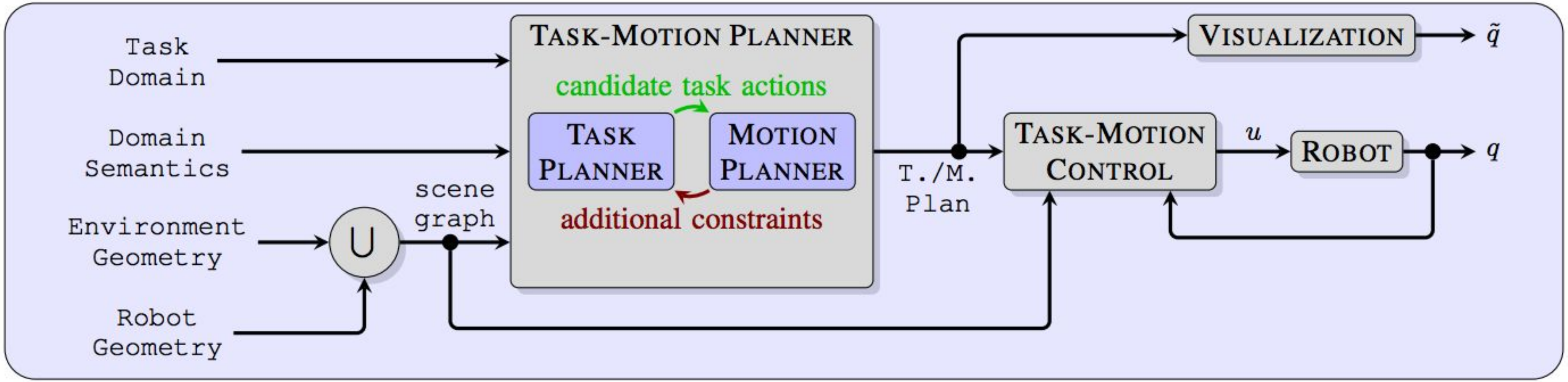
Background - Motion Planning



Background - Task and Motion Planning

- Seek to establish correspondence between task operators and motion planning problems.
- Due to probabilistic nature of sampling planners, cannot prove nonexistence of motion plans for operators.
- Task and motion plan consists of paired task operators and motion plans.
- Task and motion planning steps are typically interleaved, failed motion plans revisited.

Background - Task and Motion Planning



Related Work

- Many existing methods for TMP, mostly for specific domains
- Existing methods typically focus on performance.
- TMP methods include semantic attachments, knowledge bases, Hierarchical Planning in the Now, discretized motion planning.
- aSyMov and Synergistic Framework ensure probabilistic completeness, but use different algorithms for performing task planning, motion planning, and at the TMP interface level than TMKit.

Related Work

- Some related TMP works deal directly with differential dynamics, though TMKit does not.
- Many other TMP works use roadmaps, which then avoid much of the repeated computation necessary in TMKit.
- However, roadmap methods decrease flexibility to changes in configuration space.
- A few TMP approaches require or make use of backtracking in lieu of establishing alternate task plans.

Contributions - Abstractions

Definition 1. Task Language. *The task language is a set of strings of actions, defined by* $\mathfrak{L} = (\mathcal{P}, \mathcal{A}, \mathcal{E}, s^{[0]}, \mathcal{G})$, *where,*

- $\mathcal{P}$ *is the state space ranging over variables* $p_0, \ldots, p_n$,
- $\mathcal{A}$ *is the set of task operators, i.e., terminal symbols,*
- $\mathcal{E} \subseteq (\mathbb{P}(\mathcal{P}) \times \mathcal{A} \times \mathbb{P}(\mathcal{P}))$ *is the set of* symbolic transitions. *Each* $e_i \in \mathcal{E}$ *denotes transitions* $\mathrm{pre}(a_i) \xrightarrow{a_i} \mathrm{eff}(a_i)$, *where* $\mathrm{pre}(a_i) \subseteq \mathcal{P}$ *is the precondition set,* $a_i \in \mathcal{A}$ *is the operator, and* $\mathrm{eff}(a_i) \subseteq \mathcal{P}$ *is the effect set. We represent a concrete transition on* a_i *at step* k *from state* $s^{[k]}$ *to* $s^{[k+1]}$ *as* $s^{[k+1]} = a_i(s^{[k]})$, *where* $s^{[k]} \in \mathrm{pre}(a_i)$, $s^{[k+1]} \in \mathrm{eff}(a_i)$, *and for all state variables* p_j *independent of (i.e., free in)* $\mathrm{eff}(a_i)$, ${p_j}^{[k+1]} = {p_j}^{[k]}$,
- $s^{[0]} \in \mathcal{P}$ *is the start state,*
- $\mathcal{G} \subseteq \mathcal{P}$ *is the set of accept states, i.e., the task goal.*

Definition 2. Task Plan. *A* task plan $\mathbf{A}$ *is a string in the task language* $\mathfrak{L}$, *i.e.,* $\mathbf{A} \in \mathfrak{L}$, *where* $\mathbf{A} = (a^{[0]}, a^{[1]}, \ldots, a^{[h]})$, $a^{[k]} \in \mathcal{A}$, $s^{[k]} \in \mathrm{pre}(a^{[k]})$, $s^{[k+1]} = a^{[k]}(s^{[k]})$, *and* $s^{[h]} \in \mathcal{G}$.

Contributions - Abstractions

Definition 3. Scene Graph. $\gamma = (\mathcal{Q}, \mathcal{L}, \mathcal{F})$, *where*

- $\mathcal{Q} \subseteq \mathbb{R}^n$ *is a space of configurations,*
- $\mathcal{L}$ *is a finite set of unique frame labels,*
- $\mathcal{F}$ *is a finite set of kinematic frames (graph nodes), such that each frame* $f_\ell = (\ell, \varrho_\ell, \varsigma_\ell, \mu_\ell)$, *where,*
  - $\ell \in \mathcal{L}$ *is the unique label of frame of* f_ℓ
  - $\varrho_\ell \in \mathcal{L}$ *is the label of the parent of frame of* f_ℓ*, indicating graph edge connections*
  - $\varsigma_\ell : \mathcal{Q} \mapsto \mathcal{SE}(3)$*, maps from the configuration space to the workspace pose of* f_ℓ *relative to its parent* ϱ_ℓ*, indicating graph edge values*
  - μ_ℓ *is a rigid body mesh representing geometry attached to* f_ℓ.

Definition 4. Motion Plan. *A* motion plan *is a sequence of neighboring configurations* $\mathbf{Q} = \left(q^{[0]}, q^{[1]}, \ldots, q^{[n]}\right)$ *such that each* $q^{[k]} \in \mathcal{Q}$ *and* $\left\|q^{[i+1]} - q^{[i]}\right\| < \epsilon_q$*, for some small* ϵ_q*. The initial configuration is* $\mathrm{first}(\mathbf{Q}) = q^{[0]}$*, and the final configuration is* $\mathrm{last}(\mathbf{Q}) = q^{[n]}$.

Contributions - Abstractions

Definition 5. Task-Motion Domain.
$\mathfrak{D} = (\mathfrak{L}, \sigma^{[0]}, \lambda_\alpha, \lambda_\rho, \Omega)$

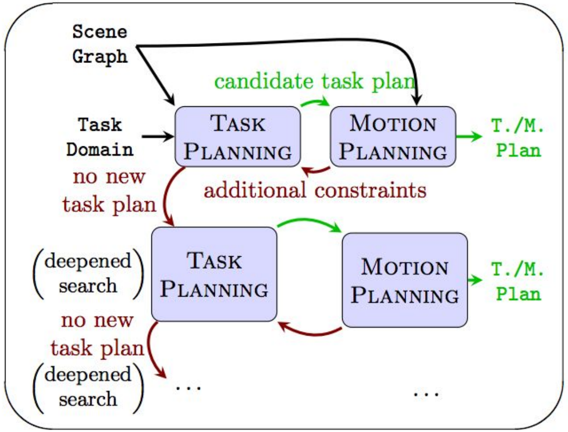
- $\mathfrak{L}$ *is the task language, where* $\mathcal{P} = \mathcal{P}_m \times \mathcal{P}_t$ *represents the motion component* $\mathcal{P}_m$ *and non-motion component* $\mathcal{P}_t$ *of task state,*
- $\sigma^{[0]} = (s^{[0]}, \gamma^{[0]}, q^{[0]})$ *is initial the task-motion state: task state* $s^{[0]}$*, scene graph* $\gamma^{[0]}$ *and configuration* $q^{[0]}$*,*
- $\lambda_\alpha : \Gamma \mapsto \mathcal{P}_m$ *domain semantics to abstract the scene graph* $\gamma \in \Gamma$ *to the motion component task state* $s_m \in \mathcal{P}_m$*,*
- $\lambda_\rho : \Gamma \times \mathcal{A} \mapsto \mathbb{P}(\mathcal{Q}) \times \Gamma$ *domain semantics to refine the initial scene graph* $\gamma^{[k]} \in \Gamma$ *and task operator* $a^{[k]} \in \mathcal{A}$ *to a motion planning goal (a set of configurations)* $\Theta^{[k]} \subseteq \mathcal{Q}$ *for the action and a final scene graph* $\gamma^{[k+1]} \in \Gamma$ *via reparenting frames,*
- $\Omega \subseteq \Gamma \times \mathcal{P}$ *is the goal condition.*

Definition 6. Task and Motion Plan. *A task and motion plan is a sequence of task operators and motion plans,* $\mathbf{T} = \left(\left(a^{[0]}, \mathbf{Q}^{[0]}\right), \ldots, \left(a^{[h]}, \mathbf{Q}^{[h]}\right)\right)$ *where* $\left(a^{[0]}, \ldots, a^{[h]}\right) \in \mathfrak{L}$ *and for each step* k*,* $\left(\Theta^{[k]}, \gamma^{[k+1]}\right) = \lambda_\rho(a^{[k]}, \gamma^{[k]})$ *such that* $\mathrm{last}\left(\mathbf{Q}^{[k]}\right) \in \Theta^{[k]} \wedge \mathrm{first}\left(\mathbf{Q}^{[k+1]}\right) = \mathrm{last}\left(\mathbf{Q}^{[k]}\right)$.

Contributions - Abstractions

- To unify task planning and motion planning layers, introduce notion of “domain semantics.”
- Domain semantics define how task operators map to motion planning problems.
- In TMKit, specified by Lisp or Python functions.
- TMKit also supports a meta-operator that “reparents” by modifying the underlying scene graph.

Contributions - IDTMP

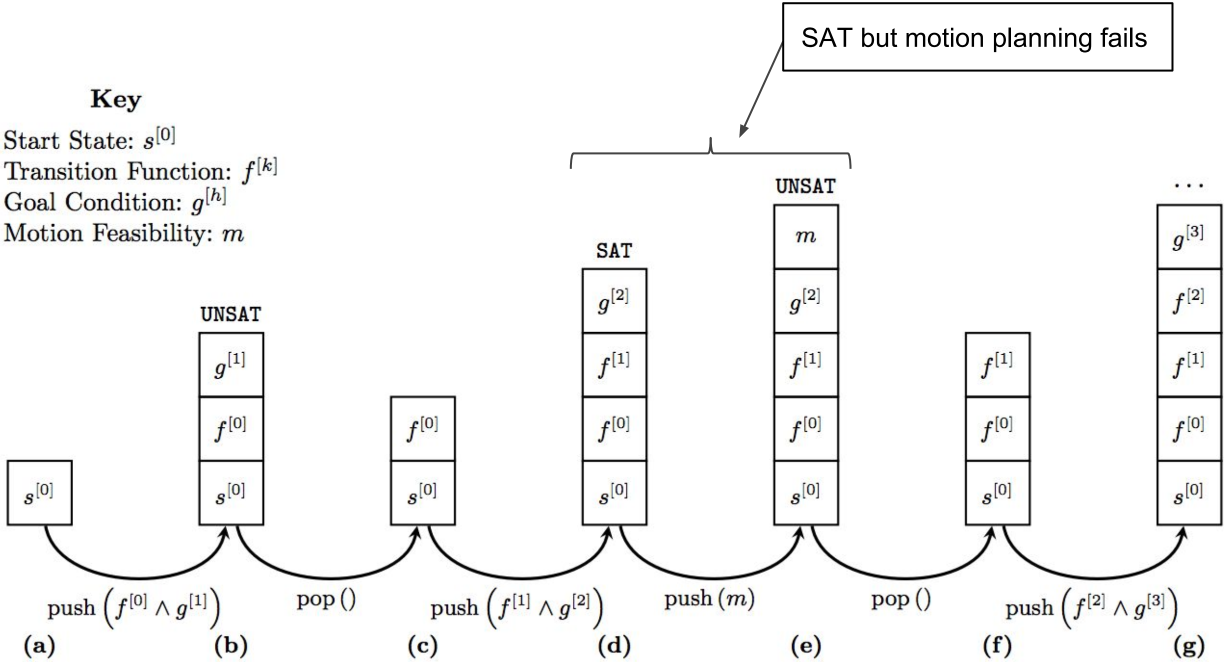


- Iteratively Deepened Task and Motion Planning.
- “Secret sauce” of TMKit.
- Essentially:
  - Generate candidate task plans with set “step horizon”
  - Evaluate task plan feasibility using motion planner with set “sampling horizon”
  - If satisfying plan found, return
  - Else record found motion constraint or increment step horizon and sampling horizon, and loop.

Contributions - IDTMP

- Incremental solving with SMT solver using a stack of “scopes” containing constraints.
- Constraints can be derived from geometric information returned by failed refinement of a task operator’s corresponding motion planning problem.
- The naive solution is to simply enumerate failed task plans as constraints.
- Better constraints developed in extensions.

Contributions - IDTMP



Contributions - Completeness

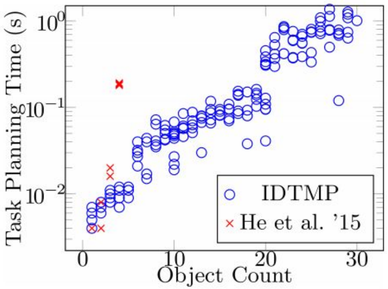
- An SMT solver will enumerate all candidate task plans at all step horizons, so if a feasible task plan exists in **n** steps, the solver is guaranteed to find it.
- RRT-Connect is probabilistically complete.
- Thus, since we retry failed motion plans by pushing and popping constraints, IDTMP is guaranteed to find a satisfying task and motion plan in the limit if one exists.

Contributions - Extensions to IDTMP

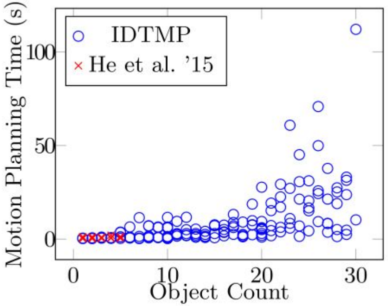
- By introducing post-plan and post-state connectivity assumptions, we can make use of more general constraints derived from failed motion plans.
- **Motion Plan Caching** - requires post-plan connectivity.
- **Failure Generalization Constraints** - requires post-state connectivity.
- **Collision Generalization Constraints** - requires post-state connectivity.
- All of these extensions preserve completeness since we pop the generated constraints eventually to deal with spurious motion planning failure.

Experiments - Scalability

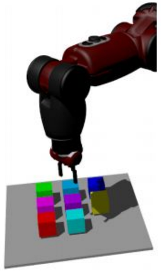
(a)



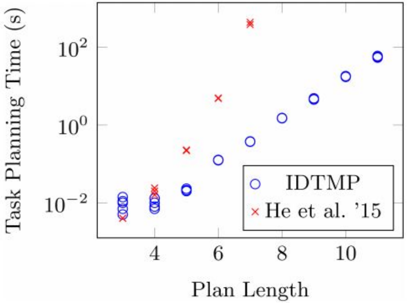
(b)



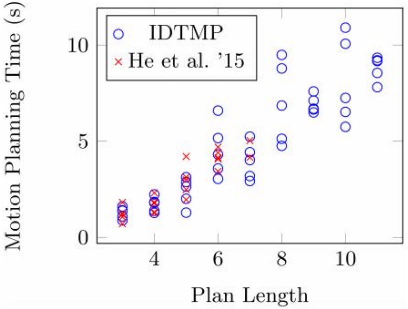
(c)



(a)

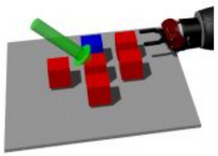


(b)

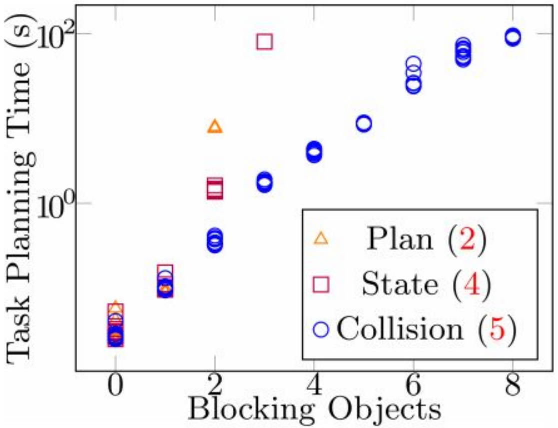


(c)

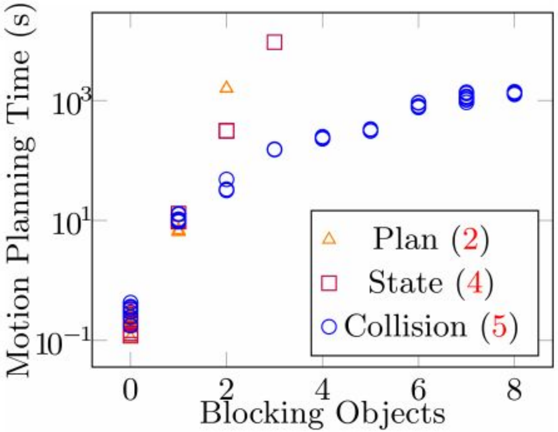
Experiments - Extensions



(a)



(b)



(c)

Dent

- Largely a technical accomplishment
  - Establishment of TMKit framework
  - Development of IDTMP algorithm
- Primary conceptual novelty in formalizations of task and motion planning concepts.
- Also exposes the inherent modularity of TMP in TMKit to facilitate future research.

Analysis - Literature Review

- Both papers make good use of interleaved references throughout.
- IJRR/RSS paper especially makes use of ~3 pages of references.
- Though task planning and combined task and motion planning are well covered by references, motion planning is less contextualized.
- Explanation of similar TMP approaches such as aSyMov and the Synergistic Framework is somewhat lacking in detail.

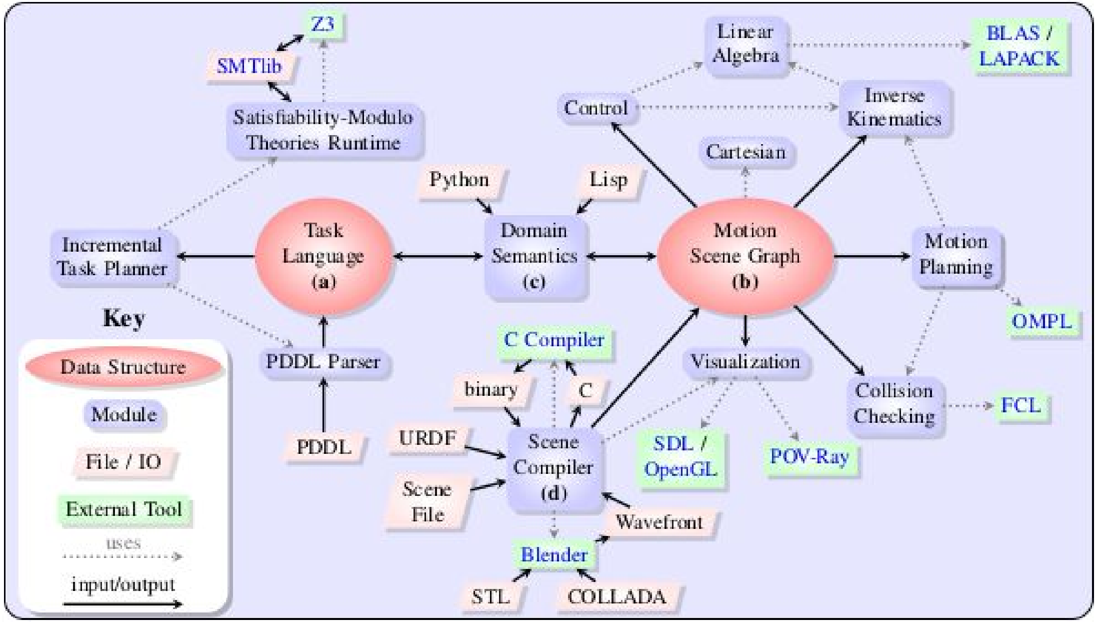
Analysis - Methodology

- One of the weaker points of the paper.
- Only three experiments performed, all in simulation, against only one alternate approach.
- Environments in which experiments are conducted pose no significant motion planning challenge.
- However, theoretical ideas are well justified, modularity seems reasonable for TMP, and IDTMP seems eminently extensible.
- Limitations to geometric motion planning are also acknowledged.

Analysis - Presentation

- Overall papers are well written, figures are informative.
- Occasional spelling errors and sentence fragments during related work discussion are distracting.
- Implementation-specific details and more general framework capabilities are not always clearly separated, especially in the RAM paper.

Questions?


Z3
SMTlib
Satisfiability-Modulo Theories Runtime
Linear Algebra
BLAS / LAPACK
Control
Inverse Kinematics
Cartesian
Python
Lisp
Incremental Task Planner
Task Language (a)
Domain Semantics (c)
Motion Scene Graph (b)
Motion Planning
OMPL
Key
Data Structure
Module
File / IO
External Tool
uses
input/output
PDDL Parser
PDDL
C Compiler
binary
C
URDF
Scene Compiler (d)
Scene File
Wavefront
Blender
STL
COLLADA
Visualization
SDL / OpenGL
POV-Ray
Collision Checking
FCL