Probabilistically Complete Kinodynamic Planning for Robot Manipulators with Acceleration Limits

Alexander Holston

Tobias Kunz and Mike Stilman ~ IROS2014

KAIST

Contents

- Objective
- Background / Related Work
- Method
- Results
- Conclusion

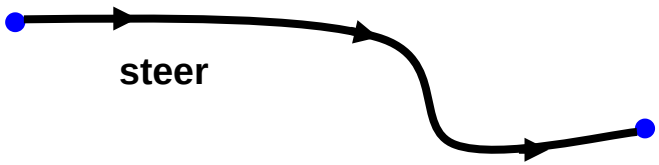
Objectives

- **Create an efficient kinodynamic planner**
  - **Uses joint acceleration limits**
- **Shows how dynamically optimal paths can be calculated**
  - **Further investigates path feasibility**
- **Considers paths with non-zero start or end starts**
  - **i.e. hitting a nail with a hammer**

Background – Geometric Planner

- First intuition would be to planning using RRT
  - RRT is a successful geometric planner
  - Works quickly for high DoF

- RRT relies on a good ‘steer’ function
  - Joins two states together
  - Typically, for differential constraints efficient methods do not exist



KAIST

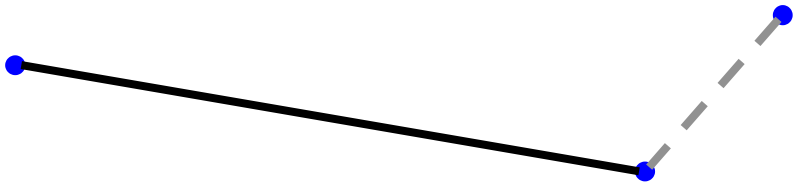
Background – KinoDynamic RRT

- Tree growth simulates a small time-step forward with a control input

- Two methods
  - Select control input randomly
    - Does not grow efficiently (Probabilistically Complete)
  - Test multiple control inputs
    - In general not probabilistically complete (Distance does not match dynamics)

Background – KinoDynamic RRT

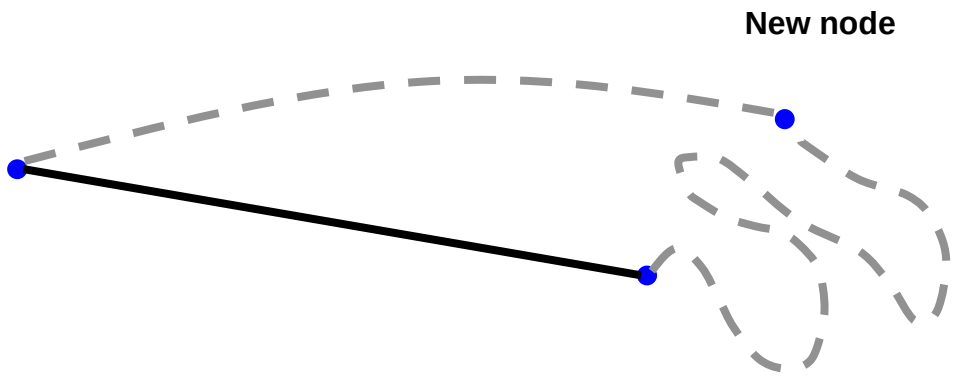
- **To overcome RRT problem KinoDynamic RRT was created**
  - **Combined incremental simulator with RRT**
  - **Uses distance function to connect**



KAIST

Background – KinoDynamic RRT

- To overcome RRT problem KinoDynamic RRT was created
  - Combined incremental simulator with RRT
  - Uses distance function to connect
    - Distance is uninformed of dynamics
    - Cannot explore the space efficiently



KAIST

Method - Introduction

- **This paper adds acceleration limits to actuators**
  - **Gives ability to apply Bounded Value Problem**
    - **Solving simplified ODE between two points**
  - **This steer function gives ability to drive towards optimal next state**

- **We can find dynamical distance between two states**
  - **Non-zero end points are now possible**
  - **Simplified ODE makes general path planning faster**

Method – Steer Function

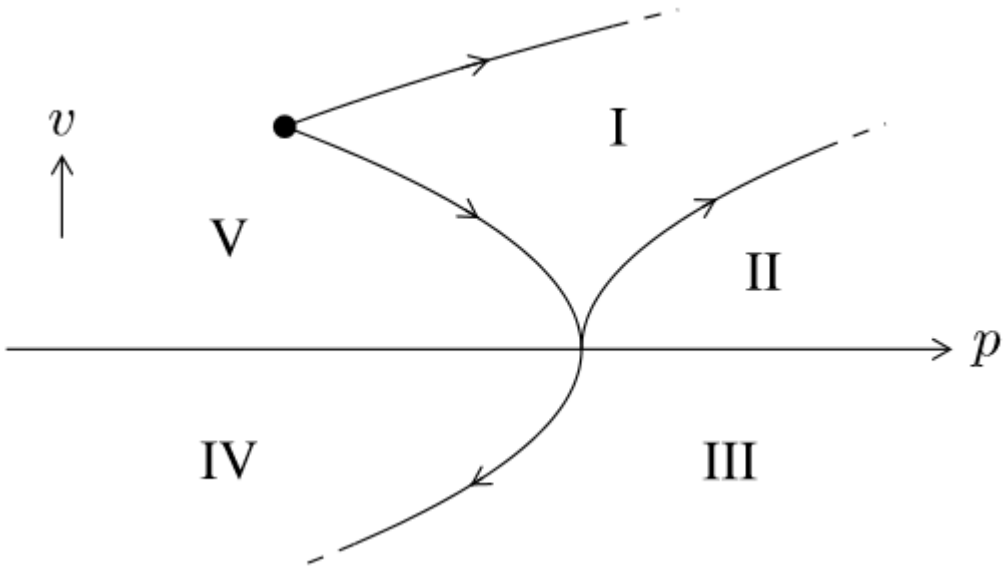
- **Break up each DoF to compute**
  1. **Find minimum time**
  2. **Check for infeasibility in solution**
     - **A particular velocity or position may not connect with another**

- **We consider these limits**

$$p_{\min} \leq p \leq p_{\max}$$
$$-v_{\max} \leq v \leq v_{\max}$$
$$-a_{\max} \leq a \leq a_{\max}$$

Method – Minimum Time

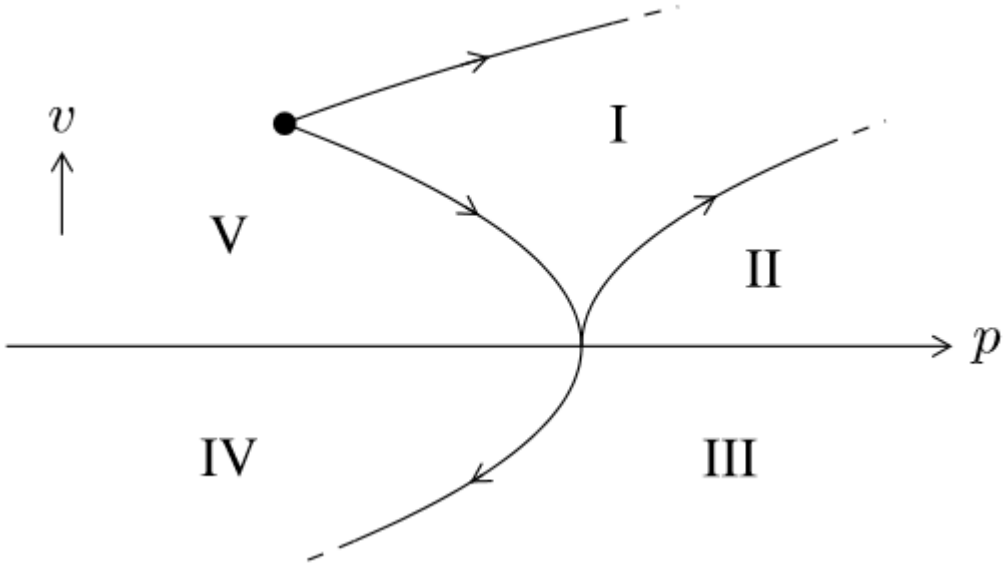
- **Move at acceleration limit** $-a_{\max} \leq a \leq a_{\max}$
  - **Connection between two points made of only 2 or 3 segments**
    - **3rd segment is velocity is exceeded**



KAIST

Method – Minimum Time

- **Determine sign of first acceleration a_1**
  - **Regions: I, II, III**
    - **Same sign as v**
  - **Regions: IV, V**
    - **Opposite sign as v**



$$\Delta p_{\text{acc}} = \frac{1}{2}(v_1 + v_2)\frac{|v_2 - v_1|}{a_{\max}}$$
$$\sigma = \text{sgn}(p_2 - p_1 - \Delta p_{\text{acc}})$$
$$a_1 = -a_2 = \sigma a_{\max}$$
$$v_{\text{limit}} = \sigma v_{\max}$$

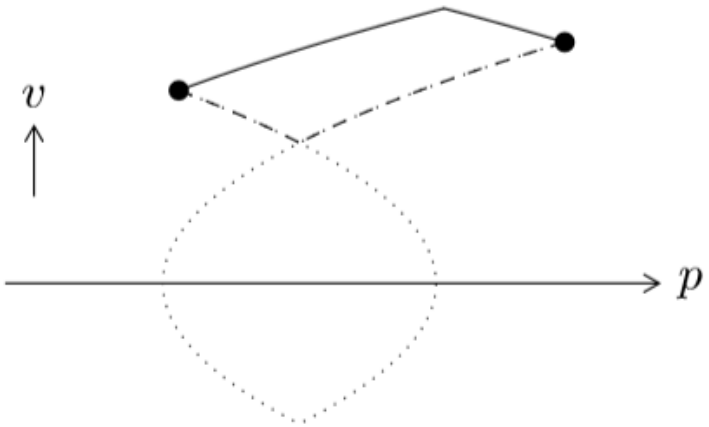
KAIST

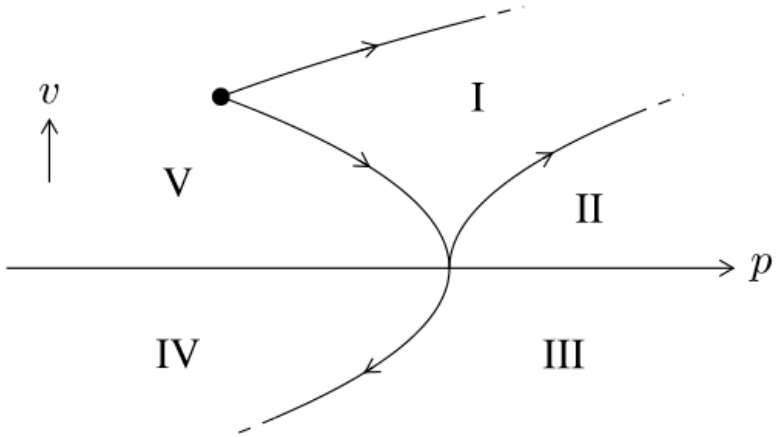
Method – Minimum Time

- Given we only have two accelerations
  - Accelerate fully, and decelerate fully
  - Solve the minimum time
- If velocity limit is reached
  - Third segment is dictated by velocity instead of acceleration

Method – Infeasibility

- **Consider high DoF**
  - **Each is calculated individually**
  - **Given obstacles etc. each DoF may need to wait**
    - **This can be done by slowing acceleration**
  - **Need to be sure it is possible**
- **Infeasibility only possible in Region: I**





Method – Solving Connection

- **Solve for minimum acceleration between two points**
  - **Remember there are infinite possible solutions between two points**
- **Solve for a_1**

$$T^2 a_1^2 + (2T(v_1+v_2) - 4(p_2-p_1))\, a_1 - (v_2-v_1)^2 = 0$$

- **Find trajectory times**

$$t_{a1} = \frac{1}{2}\left(\frac{v_2 - v_1}{a_1} + T\right) \qquad t_{a2} = T - t_{a1}$$

- **Check if max velocity is violated – Set function for acceleration accordingly**

$$a_1 = -a_2 = \frac{(v_{\text{limit}} - v_1)^2 + (v_{\text{limit}} - v_2)^2}{2(v_{\text{limit}} T - (p_2 - p_1))} \qquad v_{\text{limit}} = \text{sgn}(a_1)\, v_{\text{max}}$$

KAIST

Method – RRT Implementation

- Now that the steer function is configured it is implemented into an RRT algorithm
  - Bidirectional RRT Connect
    - Works from both ends
    - From goal with reversed dynamics
  - Rejects infeasible samples
    - 92% rejection

Method – RRT Implementation

- **Get sample**
  - **Use steer to check if feasible**
  - **Add intermediate states to V and E**
  - **If both connect to same point – Success**
- **NearestNeighbour uses Steer to search**

Algorithm 1: DIMT-RRT($x_{\text{init}}, X_{\text{goal}}, \Delta t$)

$V_1 \leftarrow \{x_{\text{init}}\}$; $E_1 \leftarrow \emptyset$;
$V_2 \leftarrow X_{\text{goal}}$; $E_2 \leftarrow \emptyset$;
$d =$ **true**;
while true do
  $x_{\text{rand}} \leftarrow$ SampleReachableState();
  if Connect$(V_1, E_1, x_{\text{rand}}, d, \Delta t)$ **then**
    if Connect$(V_2, E_2, x_{\text{rand}}, \neg d, \Delta t)$ **then**
      return ExtractTrajectory(V_1, E_1, V_2, E_2, d);
  Swap$((V_1, E_1), (V_2, E_2))$;
  $d = \neg d$;

Algorithm 2: Connect($V, E, x_{\text{rand}}, d, \Delta t$)

$x_{\text{near}} \leftarrow$ NearestNeighbor(V, x_{rand}, d);
$(T, \sigma) \leftarrow$ Steer$(x_{\text{near}}, x_{\text{rand}}, d)$;
if CollisionFree(T, σ) **then**
  $X_{\text{int}} \leftarrow$ IntermediateStates$(T, \sigma, \Delta t)$;
  $V \leftarrow V \cup X_{\text{int}} \cup \{x_{\text{rand}}\}$;
  $E \leftarrow E \cup \{x_{\text{near}}\} \times X_{\text{int}} \cup \{(x_{\text{near}}, x_{\text{rand}})\}$;
  return true;
else
  return false;

KAIST

Results

- **Tested on problem of hitting nail on the head**
  - **Acceleration is important to give force to the nail**
  - **Velocity needs to be parallel**

	RRT only	after 100 shortcuts	after 200 shortcuts
# samples	39.5 ± 41.3	-	-
# nodes	567.1 ± 407.0	-	-
Computation time	56 ms ± 54 ms	113 ms ± 63 ms	157 ms ± 65 ms
Trajectory length	12.4 s ± 4.0 s	6.4 s ± 1.1 s	6.1 s ± 1.1 s

	DIMT-RRT [this paper]		Kinodynamic RRT [6]	
	extend	connect	extend	connect
# samples	9,019 ± 8,415	14.6 ± 10.8	> 1,000,000	
# nodes	9,633 ± 8,096	434.1 ± 188.4	> 900,000	> 2,500,000
Computation time	19.8 s ± 37.1 s	37 ms ± 22 ms	> 8 hours	> 23 hours

KAIST

Results

- **Geometric Planner reaches the nail but ignores the required dynamics (reaches the sides)**

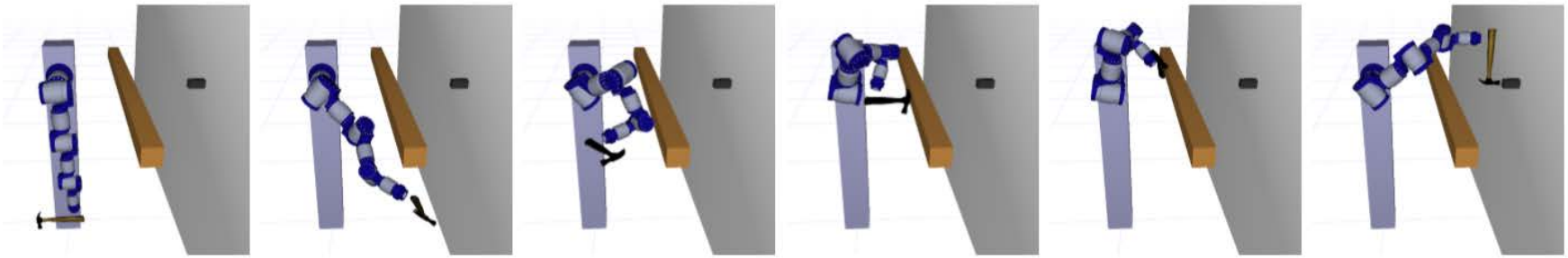


Fig. 6. The DIMT-RRT planner hits the nail at the desired velocity

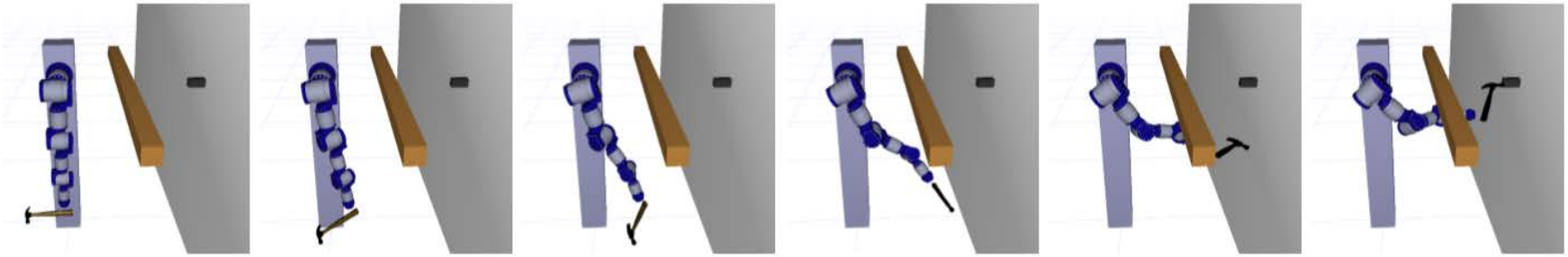


Fig. 7. The geometric RRT planner reaches the nail but not at the desired velocity

Conclusion

- **Acceleration limits were applied to allow a steer function to replace distance**
  - **Allows efficient calculations that consider real dynamics**
- **Gives a probabilistically complete outcome**
  - **Solution can adequately explore feasible solutions while extending towards the desired outcome**
    - **Due to ability to computer BVP**
- **Good balance between geometric planner and full dynamic simulations**

KAIST

Q&A

?

KAIST