Supplementary Material for the paper *Kinodynamic Planning in the Configuration Space via Velocity Interval Propagation*

Quang-Cuong Pham, Stéphane Caron, Yoshihiko Nakamura
Department of Mechano-Informatics, University of Tokyo, Japan

June 1, 2013

A Useful lemmata

Lemma 1 Assume that a forward β-profile hits the MVC at $s = s_1$ and a backward α-profile hits the MVC at $s = s_2$, with $s_1 < s_2$, then there exists at least one $\alpha \rightarrow \beta$ switch point on the MVC at some position $s_3 \in [s_1, s_2]$.

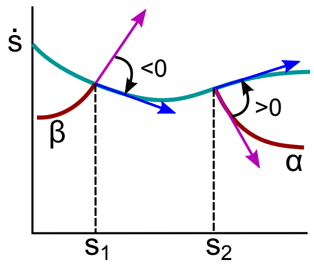

Figure 1: Illustration for the Switch Point Lemma.

Proof: At $(s_1, \mathrm{MVC}(s_1))$, the angle from the vector β to the tangent to the MVC is negative (see Fig. 1). In addition, since we are on the MVC, we have $\alpha = \beta$, thus the angle from α to the tangent is negative too. Next, at $(s_2, \mathrm{MVC}(s_2))$, the angle of α to the tangent to the MVC is positive (see Fig. 1). Thus, by the continuity of the vector field α, we have that, between s_1 and s_2, either there exists a point where the angle between α and the tangent to the MVC is 0 (in which case we have a *tangent* switch point), or there exists a point where the MVC is discontinuous (in which case we have a *discontinuous* switch point) or is non differentiable (in which case we have a *zero-inertia* switch point). For more details, the reader is referred to [7, 6] □

Lemma 2 Either one of the LCs reaches $\dot{s} = 0$, or the CLC is *continuous*

Proof Assume by contradiction that no LC reaches $\dot{s} = 0$ and that there exist "holes" in the CLC. Consider the smallest "hole". The left border s_1 of the hole must then be defined by the intersection of the MVC with a forward β-LC (coming from the previous $\alpha \to \beta$ switch point), and the right border s_2 of the hole must be defined by the intersection of the MVC with a backward α-LC (coming from the next $\alpha \to \beta$ switch point). By the Lemma 1, there must then exist a switch point between s_1 and s_2, which contradicts the fact that the hole was chosen to be the smallest □

B Comparison of VIP-RRT with KNN-RRT on the double pendulum example

B.1 KNN-RRT

B.1.1 Overall algorithm

Our implementation of RRT in the state-space [2] is detailed in Algorithms 1 and 2.

Box 1 KNN_RRT$(\mathbf{x}_{\mathrm{init}}, \mathbf{x}_{\mathrm{goal}})$

```
1: $\mathcal{T}$.INITIALIZE($\mathbf{x}_{\mathrm{init}}$)
2: for $\mathrm{rep} = 1$ to $N_{\mathrm{max\_rep}}$ do
3:    $\mathbf{x}_{\mathrm{rand}} \leftarrow$ RANDOM_STATE()
4:    $\mathbf{x}_{\mathrm{new}} \leftarrow$ EXTEND($\mathcal{T}, \mathbf{x}_{\mathrm{rand}}$)
5:    $\mathcal{T}$.ADD_VERTEX($\mathbf{x}_{\mathrm{new}}$)
6:    $\mathbf{x}_{\mathrm{new2}} \leftarrow$ EXTEND($\mathbf{x}_{\mathrm{new}}, \mathbf{x}_{\mathrm{goal}}$)
7:    if $d(\mathbf{x}_{\mathrm{new}}, \mathbf{x}_{\mathrm{goal}}) \leq \epsilon$ or $d(\mathbf{x}_{\mathrm{new2}}, \mathbf{x}_{\mathrm{goal}}) \leq \epsilon$ then
8:       return Success
9:    end if
10: end for
11: return Failure
```

Steer-to-goal frequency: Note that every 5 extension attempts, the algorithm tries to steer directly to $\mathbf{x}_{\mathrm{goal}}$ (by setting $\mathbf{x}_{\mathrm{rand}} = \mathbf{x}_{\mathrm{goal}}$ on line 3 of Algorithm 1). See also the discussion in [2], p. 387, about the use of unidirectional and bidirectional RRTs. We have also noted that the frequency of steer-to-goal did not significantly affect the performance of the algorithm, except when it is too large, *e.g.*, once every two extension attempts.

Metric: the metric for the neighbors search in EXTEND (Algorithm 2) and to assess whether the goal has been reached (line 7 of Algorithm 1) is defined as:

$$\begin{aligned} d(\mathbf{x}_a, \mathbf{x}_b) &= d\left((\mathbf{q}_a, \mathbf{v}_a), (\mathbf{q}_b, \mathbf{v}_b)\right) \\ &= \frac{\sum_{j=1,2} \sqrt{1 - \cos(\mathbf{q}_{aj} - \mathbf{q}_{bj})}}{4} + \frac{\sum_{j=1,2} |\mathbf{v}_{aj} - \mathbf{v}_{bj}|}{4V_{\max}}, \end{aligned} \tag{1}$$

where $V_{\max}$ denotes the maximum velocity bound set in the random sampler (function RANDOM_STATE() in Algorithm 1). This simple metric is similar to an Euclidean metric but takes into account the periodicity of the joint values.

Termination condition: we defined the goal area as a ball of radius $\epsilon = 10^{-2}$ for the metric (1) around the goal state $\mathbf{x}_{\text{goal}}$. As an example, $d(\mathbf{x}_a, \mathbf{x}_a) = \epsilon$ corresponds to a maximum angular difference of $\Delta q_1 \approx 0.057$ rad $\approx$ 3.24 degrees in the first joint.

This choice is connected to that of the integration time step (used *e.g.*, in Forward Dynamics computations in section B.1.2), which we set to $\delta t = 0.01$ s. Indeed, the average angular velocities we observed in our benchmark was around $\bar{V} = 5$ rad.s^{-1} for the first joint, which corresponds to an average instantaneous displacement $\bar{V} \cdot \delta t \approx 5.10^{-2}$ rad, which is of the same order as Δq_1 above.

Nearest-neighbor heuristic: instead of considering only extensions from the nearest neighbor as in the standard algorithm, we considered the best extension from the K nearest neighbors (line 5 in Algorithm 2). Here "best" is defined as the closest to $\mathbf{x}_{\text{rand}}$ in the metric d defined in Equation (1).

Box 2 EXTEND($\mathcal{T}, \mathbf{x}_{\text{rand}}$)

```
1: for $k = 1$ to $K$ do
2:     $\mathbf{x}^k_{\text{near}} \leftarrow$ KTH_NEAREST_NEIGHBOR($\mathcal{T}, \mathbf{x}_{\text{rand}}, k$)
3:     $\mathbf{x}^k_{\text{new}} \leftarrow$ STEER($\mathbf{x}^k_{\text{near}}, \mathbf{x}_{\text{rand}}$)
4: end for
5: return $\arg\min_k d(\mathbf{x}^k_{\text{new}}, \mathbf{x}_{\text{rand}})$
```

B.1.2 Local steering

Regarding the local steering scheme (STEER on line 3 of Algorithm 2), there are two main approaches : "controller-based" steering of "sampling-based" steering (cf. [2]).

"Controller-based" steering In the "controller-based" approach, one would try to design a controller that would bring the system from a given state to another given state. However, as remarked in [2], for a nonlinear system this is as a very difficult problem. To allow meaningful comparisons with VIP-RRT, we considered the following simple "controller-based" scheme, which finds a path in state-space before checking its feasibility through inverse dynamics – in the same spirit as VIP-RRT, which finds a path in configuration-space and then checks for a feasible time-parametrization. Trying to design the best possible nonlinear controller for the double pendulum would be out of the scope of this work, as it would imply either problem-specific tunings or substantial modifications to the core RRT algorithm (as done e.g. in [3]).

To connect the states $\mathbf{x}_a = (\mathbf{q}_a, \mathbf{v}_a)$ and $\mathbf{x}_b = (\mathbf{q}_b, \mathbf{v}_b)$, we interpolated, for each joint i, a third-order polynomial $P_i(t)$ between the $\mathbf{x}_{ai}$ and $\mathbf{x}_{bi}$. Since a third-order polynomial has four parameters and there are four constraints for each joint ($P_i(0) = \mathbf{q}_{ai},\ P_i'(0) = \mathbf{v}_{ai},\ P_i(T) = \mathbf{q}_{bi},\ P_i(T) = \mathbf{q}_{bi}$), the only free parameter is the time duration T of the local trajectory. Our local planner tries 10 different values of T between 0.01 s and 2 s.

For each value of T, we thus have a trajectory connecting $\mathbf{x}_a$ and $\mathbf{x}_b$. By inverse dynamics, we compute the torques necessary to execute this trajectory: if they fall within the torque limits, then the trajectory is deemed feasible, and the local steering returns $\mathbf{x}_b$ as $\mathbf{x}_{\text{new}}$. Otherwise, we cut the trajectory at the last state before violation of the torque bounds. If no value of T allows reaching $\mathbf{x}_b$, we pick the one whose last state is closest to $\mathbf{x}_b$ and return it as $\mathbf{x}_{\text{new}}$.

"Sampling-based" steering This is the standard method when no efficient "controller-based" method exists [2, 1]. Our implementation closely follows [2, 1] and is described in Algorithm 3 below.

Box 3 STEER($\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{rand}}$)

```
1: for $p = 1$ to $N_{\text{local\_trajs}}$ do
2:     $\mathbf{u} \leftarrow$ RANDOM_CONTROL($\tau_1^{\max}, \tau_2^{\max}$)
3:     $\Delta t \leftarrow$ RANDOM_DURATION($\Delta t_{\max}$)
4:     $\mathbf{x}^p \leftarrow$ FORWARD_DYNAMICS($\mathbf{x}_{\text{near}}, \mathbf{u}, \Delta t$)
5: end for
6: return $\operatorname{argmin}_p d(\mathbf{x}^p, \mathbf{x}_{\text{rand}})$
```

The random control is a stationary (τ_1, τ_2) sampled as:

$$(\tau_1, \tau_2) \sim \mathcal{U}([-\tau_1^{\max}, \tau_1^{\max}] \times [-\tau_2^{\max}, \tau_2^{\max}]).$$

The random time duration Δt is sampled uniformly in $[\delta t, \Delta t_{\max}]$ where $\Delta t_{\max}$ is the maximum time duration of local trajectories (parameter to be tuned), and δt is the time step for the forward dynamics integration, set to $\delta t = 0.01$ s as discussed in Section B.1.1. The number of local trajectories to be tested, $N_{\text{local\_trajs}}$, is also a parameter to be tuned.

Comparing the two steering methods The "controller-based" steering yielded RRTs with much slower exploration speeds (and, as a consequence, much higher search time) compared with the "sampling-based" steering, as illustrated in Figure 2.

The slow exploration of the "controller-based" steering method may be explained by the fact that, since the velocities are sampled uniformly in a wide range $[-V_{\max}, +V_{\max}]$, most attempted interpolations are rejected because of insufficient torques.

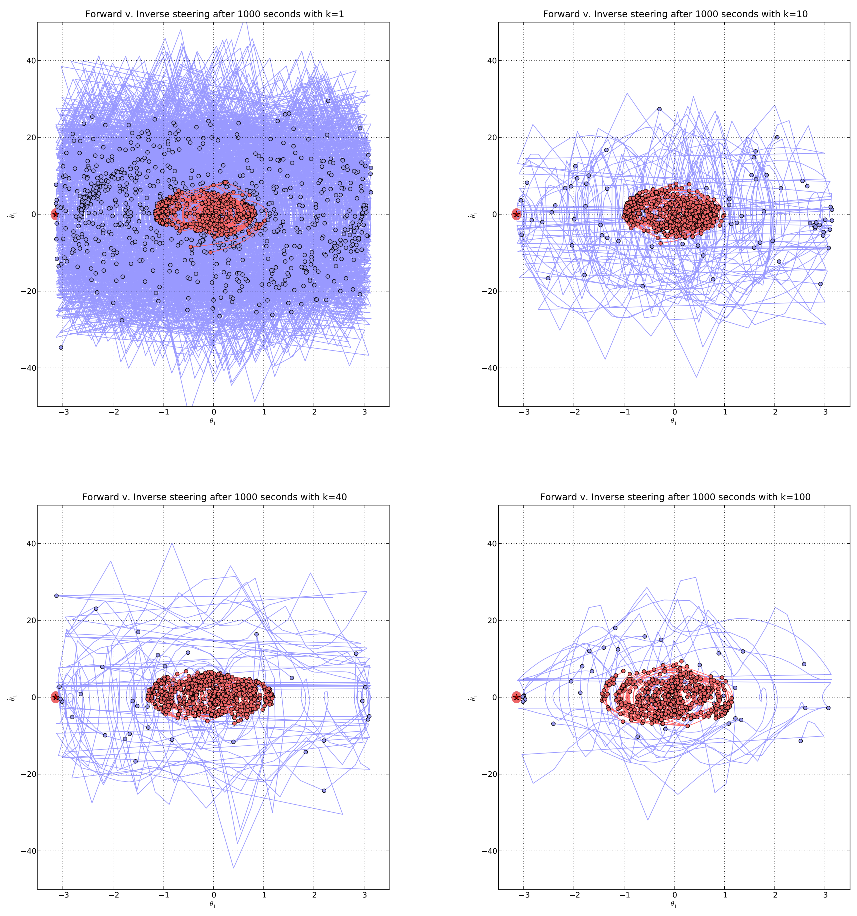

Figure 2: Comparison of "controller-based" and "sampling-based" steering methods for $K \in \{1, 10, 40, 100\}$. The X-axis represents the angle of the first joint and the Y-axis its velocity. The trees grown by the "controller-based" and "sampling-based" methods are in red and blue, respectively. The goal area is depicted by the red ellipse on the left side.

Let us remark here that, although VIP-RRT follows the “controller-based” paradigm (it indeed interpolates paths in configuration space and then computes feasible velocities along the path using Bobrow-like approach, which includes inverse dynamics computations), it is much more successful. The main reason is that the Velocity Interval Propagation calculates the whole interval of all reachable velocities, and then iteratively propagates it – instead of being bound to exploring just one velocity (as in the case of state-space RRTs) which often may not be reachable, and when it is, may not allow reaching a wide range of velocities in a subsequent step.

Note however that VIP-RRT only saves and propagates the *norm* of the velocity vectors, not their directions, which may make the algorithm probabilistically incomplete (cf. discussion in the main paper).

B.1.3 Fine-tuning the parameters of KNN-RRT with “sampling-based” steering

Based on the above results, we shall focus on KNN-RRTs with “sampling-based” steering in the remainder of this section B.

The parameters to be tuned are thus :

- $N_{\text{local\_trajs}}$: number of local trajectories tested in each call to STEER;
- $\Delta t_{\max}$: maximum duration of each local trajectory.

The values tested for these two parameters are summed up in table 1.

Number of trials	$N_{\text{local\_trajs}}$	$\Delta t_{\max}$
10	1	0.2
10	30	0.2
10	80	0.2
20	20	0.5
20	20	1.0
20	20	2.0

Table 1: Parameter sets for each test.

The parameters we do not tune are :

- Maximum velocity $V_{\max}$ for sampling velocities. We set $V_{\max} = 50\,\text{rad.s}^{-1}$, which is about twice the maximum velocity observed in the successful trials of VIP-RRT in [5];
- Number of neighbors K. In this tuning phase, we set $K = 10$. Other values of K will be tested in the final comparison with VIP in section B.2;

- Space-time precision $(\epsilon, \delta t)$: as discussed in Section B.1.1, we chose $\epsilon = 0.01$ and $\delta t = 0.01$ s.

Finally, in this tuning phase, we set the torque limit as $(\tau_1^{\max}, \tau_2^{\max}) = (13, 7)$ N.m, which are relatively "easy" values, in order to obtain faster termination times for RRT. More difficult values such as $(\tau_1^{\max}, \tau_2^{\max}) = (11, 5)$ N.m will be tested in our final comparison with VIP-RRT in section B.2.

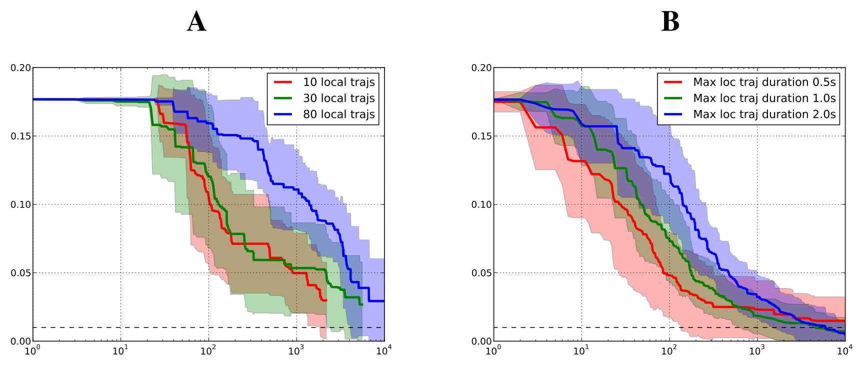

Figure 3: Minimum distance to the goal as a function of time for different values of $N_{\text{local\_trajs}}$ and $\Delta t_{\max}$. At each instant, the minimum distance of the tree to the goal is computed. The average of this value across the 10 trials of each set is drawn in bold, while shaded areas indicate standard deviations. **A**: tuning of $N_{\text{local\_trajs}}$. **B**: tuning of $\Delta t_{\max}$.

Fig. 3A shows the result of simulations for different values of $N_{\text{local\_trajs}}$. One can note that the performance of RRT is similar for values 10 and 30, but gets worse for 80. Based on this observation, we chose $N_{\text{local\_trajs}} = 20$ for the final comparison in section B.2;.

Fig. 3B shows the simulation results for various values of $\Delta t_{\max}$. One can note that the performance of RRT is similar for the three tested values, with smaller values (*e.g.*, 0.5 s) performing better earlier in the trial and larger values (*e.g.*, 2.0 s) performing better later on. We also noted that smaller values of $\Delta t_{\max}$ such as 0.1 s or 0.2 s tended to yield poorer results (not shown here). Our choice for the final comparison was thus $\Delta t_{\max} = 1.0$ s.

B.2 Comparing KNN-RRT and VIP-RRT

In this section, we compare the performance of KNN-RRT (for $K \in \{1, 10, 40, 100\}$, the other parameters being set to the values discussed in the previous section) against VIP-RRT with 10 neighbors (the one presented in [5]). For practical reasons, we further limited the execution time of every trial to 10^4 s, which had no impact in most cases or otherwise induced a slight bias in favor of RRT (since we took 10^4 s as our estimate of the "search time" when RRT does not terminate within this time limit).

We ran the simulations for two instances of the problem, namely

- $(\tau_1^{\max}, \tau_2^{\max}) = (11, 7)$ N.m and
- $(\tau_1^{\max}, \tau_2^{\max}) = (11, 5)$ N.m, the one presented in in [5].

For each problem instance, we ran 40 trials for each planner VIP-RRT, state-space RRT with 1 nearest neighor (RRT-1), RRT-10, RRT-40 and RRT-100. Note that for each trial i, all the planners received the same sequence of random states

$$\mathbf{X}_i = \left\{ \mathbf{x}_{\text{rand}}^{(i)}(t) \in \mathbf{R}^4 \;\middle|\; t \in \mathbf{N} \right\} \sim \mathcal{U}\left((] - \pi, \pi]^2 \times [-V_{\max}, +V_{\max}]^2)^{\mathbf{N}} \right),$$

although VIP-RRT only used the first two coordinates of each sample since it plans in the configuration space.

Table 2 and Fig. 4 present the results for the problem instance $(\tau_1^{\max}, \tau_2^{\max}) = (11, 7)$ N.m. All trials of VIP successfully terminated within the time limit. The average search time was 3.3 min. Among the KNN-RRT, RRT-40 performed best with a success rate of 92.5% and an average computation time ca. 45 min, which is however 13.4 times slower than VIP-RRT.

Planner	Success rate	Search time (min)
VIP-RRT	100%	3.3±2.6
RRT-1	40%	70.0±34.1
RRT-10	82.5%	53.1±59.5
RRT-40	92.5%	44.6±42.6
RRT-100	82.5%	88.4±54.0

Table 2: Comparison of VIP-RRT and KNN-RRT for torque limits $(\tau_1^{\max}, \tau_2^{\max}) = (11, 7)$. Note that in the calculation of average search times, we set the search times of unsuccessful trials to the time-out limit 10^4 s.

Table 3 and Fig. 5 present the results for the second instance $(\tau_1^{\max}, \tau_2^{\max}) = (11, 5)$ N.m. All trials of VIP successfully terminated within this time limit, with an average search time ca. 9.8 min. Among the KNN-RRT, again RRT-40 performed best in terms of search time (54.6 min on average, which was 5.6 times slower than VIP-RRT), but RRT-100 performed best in terms of success rate within the 10^4s time limit (92.5%).

We noted that the search time of VIP increased significantly from instance $(\tau_1^{\max}, \tau_2^{\max}) = (11, 5)$ to instance $(\tau_1^{\max}, \tau_2^{\max}) = (11, 7)$, while that of RRT only marginally increased. This may be related to the remark of reviewer 1 about the "superposition" of states: as torque constraints become tighter, more "pumping" swings are necessary to reach upright configurations. However, since we plan in configuration space, configurations with different speeds (corresponding to different pumping cycles) can become indistinguishable. While this problem could easily be addressed by including in the distance computation the reachable velocity interval associated with each vertex, we chose not to do so here to preserve

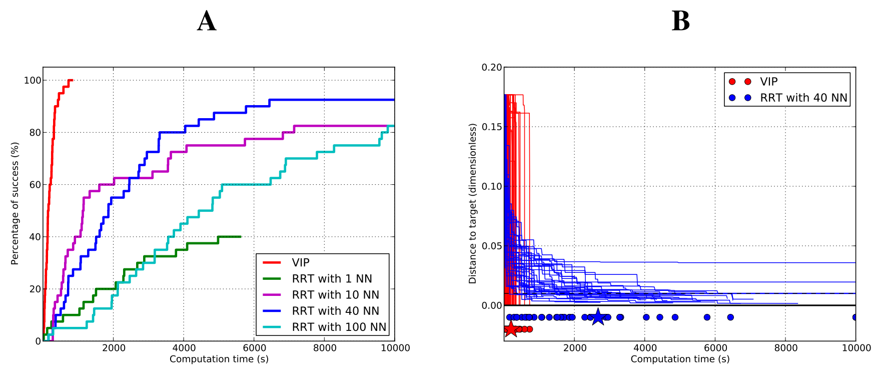

Figure 4: Results for torque limits (11,7). **A**: Percentage of trials that have reached the goal area at given time instants. **B**: Individual plots for each trial. Each curve shows the distance to the goal as a function of time for a given instance (red: VIP-RRT, blue: RRT-40). Dots indicate the time instants when a trial successfully terminated. Stars show the mean values of termination times.

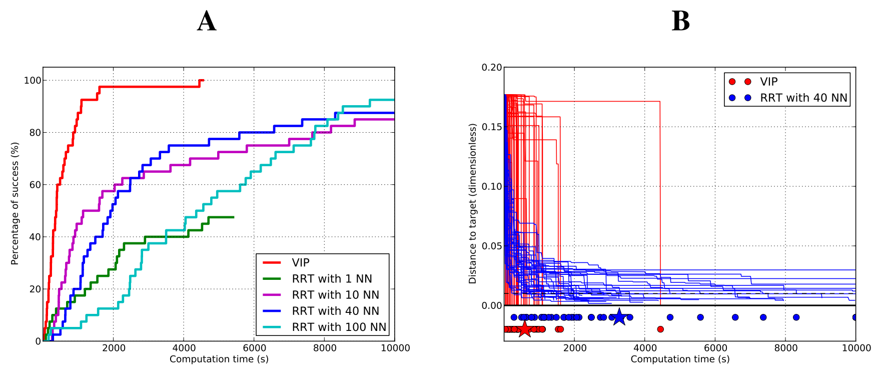

Figure 5: Results for torque limits (11,5). Same legends as in Fig. 4.

Planner	Success rate	Search time (min)
VIP-RRT	100%	9.8±12.1
RRT-1	47.5%	63.8±36.6
RRT-10	85%	56.3±60.1
RRT-40	87.5%	54.6±52.2
RRT-100	92.5%	81.2±46.7

Table 3: Comparison of VIP-RRT and KNN-RRT for torque limits $(\tau_1^{\max}, \tau_2^{\max}) = (11, 5)$.

generality. But even without this enhancement, VIP-RRT still significantly overperformed KNN-RRT.

More generally, for asymptotically small torque limits, VIP-RRT might be eventually overtaken by a state space KNN-RRT. Yet, for reasonably tight torque limits as tested here (which still require several pumping cycles), VIP-RRT still provides a significant improvement in search time and success rate over KNN-RRT. In terms of the number of free parameters to be tuned, VIP-RRT is also more parsimonious than KNN-RRT (for instance VIP-RRT does not require tuning such parameters as $N_{\text{local\_trajs}}$, $\Delta t_{\max}$, $V_{\max}$).

B.3 Conclusion

In this section, we have conducted a comparison of the VIP-RRT algorithm (with 10 neighbors), proposed in [5], with fine-tuned KNN-RRTs, on the example of a double pendulum with torque limits. In the two problem instances, VIP-RRT was respectively 13.4 and 5.6 times faster than the best KNN-RRT in terms of search time.

As the potential gain from planning in configuration space versus planning in state space is expected to increase with the dimension of the considered system, we believe that the improvements shown on the current simple example will scale up and make possible the resolution of systems of higher dimensions (*e.g.*, triple, quadruple pendulums, etc., or humanoid robots – which are inaccessible to today's kinodynamic planners). Another advantage of planning in configuration space lies in the space/time decoupling effect, which is more relevant in the bottle manipulation example where configuration-space obstacles play a significant role, but which we could not explore here because of time constraints.

The two above directions are the subject of our current research efforts.

C Reduction of the bottle + tray system to Bobrow form

Let $\mathbf{p} = (x, z)$ denote the coordinates of the center of gravity of the bottle in the laboratory reference frame, $m\mathbf{g}$ the gravity force, $\mathbf{f}_N$ the normal *reaction* force that the tray exerts on the bottle and $\mathbf{f}_F$ the *friction* force that the tray exerts on the bottle (Fig. 6). Newton's second law then gives

$$m\ddot{\mathbf{p}} = m\mathbf{g} + \mathbf{f}_N + \mathbf{f}_F.$$

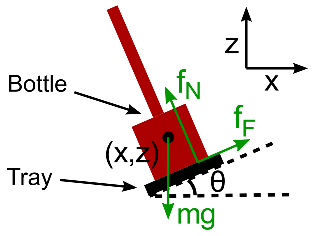

Figure 6: A bottle on a tray. The vectors $\mathbf{f}_N$ and $\mathbf{f}_F$ represent respectively the normal reaction force and the friction force. For the bottle not to move with respect to the tray, one must ensure that $\|\mathbf{f}_F\| \leq \mu\|\mathbf{f}_N\|$, where μ is the coefficient of static friction.

Projecting the above equation on the x- and z- axis gives

$$\begin{aligned} m\ddot{x} &= -N\sin\theta + F\cos\theta, \\ m\ddot{z} &= -mg + N\cos\theta + F\sin\theta, \end{aligned}$$

where $N = \|\mathbf{f}_N\|$ and $F = \|\mathbf{f}_F\|$.

This leads to, after some simple algebraic manipulations,

$$N = m\cos^2\theta(g + \ddot{z} - \ddot{x}\tan\theta), \tag{2}$$

$$F = \frac{m\ddot{x} + m\cos^2\theta(g + \ddot{z} - \ddot{x}\tan\theta)\sin\theta}{\cos\theta}. \tag{3}$$

To ensure that the bottle does not move with respect to the tray, the following two constraints must be satisfied[1]:

- the normal reaction force is non-negative, i.e. $N \geq 0$;
- the friction force is limited by $-\mu N \leq F \leq \mu N$, where μ is the coefficient of static friction (Coulomb's law for static friction).

Using the expressions obtained in (2) and (3), these conditions can expressed as

$$g + \ddot{z} - \ddot{x}\tan\theta \geq 0, \tag{4}$$

$$\ddot{x} + \cos^2\theta(g + \ddot{z} - \ddot{x}\tan\theta)(\sin\theta + \mu\cos\theta) \geq 0, \tag{5}$$

$$\ddot{x} + \cos^2\theta(g + \ddot{z} - \ddot{x}\tan\theta)(\sin\theta - \mu\cos\theta) \leq 0. \tag{6}$$

[1]Actually, there exists a third contact condition, related to the ZMP. However, we chose here to neglect this condition for the sake of exposition clarity. This is equivalent to assuming that the mass of the bottle is concentrated at its bottom. Note that the ZMP condition can also be reduced to the Bobrow form, cf. [4].

Assume that we are given a D^2 trajectory $(x(u), z(u), \theta(u))_{u\in[0,T]}$. Following Bobrow's method, we differentiate $x(s(t))$ and $z(s(t))$ twice with respect to t to obtain

$$\begin{aligned} \ddot{x} &= \ddot{s}x_s + \dot{s}^2 x_{ss}, \\ \ddot{z} &= \ddot{s}z_s + \dot{s}^2 z_{ss}. \end{aligned}$$

Substituting the above equations into (4) yields

$$\ddot{s}(z_s - x_s \tan\theta) \geq -(z_{ss} - x_{ss}\tan\theta)\dot{s}^2 - g. \tag{7}$$

There are three cases:

- if $z_s - x_s \tan\theta = 0$, then we have a zero-inertia point, which must be dealt with specifically;
- if $z_s - x_s \tan\theta > 0$, then we have a *lower* bound on the acceleration

$$\ddot{s} \geq \alpha(\dot{s}, s) = \frac{-(z_{ss} - x_{ss}\tan\theta)\dot{s}^2 - g}{z_s - x_s\tan\theta};$$

- if $z_s - x_s \tan\theta < 0$, then we have an *upper* bound on the acceleration

$$\ddot{s} \leq \beta(\dot{s}, s) = \frac{-(z_{ss} - x_{ss}\tan\theta)\dot{s}^2 - g}{z_s - x_s\tan\theta}.$$

The same manipulations can be applied to inequalities (5) and (6), which provide further bounds on the accelerations. Note that this may also give rise to other zero-inertia points.

References

[1] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[2] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[3] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez. LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *IEEE International Conference on Robotics and Automation*, 2012.

[4] Q.-C. Pham and Y. Nakamura. Time-optimal path parameterization for critically dynamic motions of humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[5] Q.-C. Pham, S. Caron, and Y. Nakamura. Kinodynamic planning in the configuration space via velocity interval propagation. In *Robotics: Science and System*, 2013.

[6] Z. Shiller and H.H. Lu. Computation of path constrained time optimal motions with dynamic singularities. *Journal of dynamic systems, measurement, and control*, 114:34, 1992.

[7] J.J.E. Slotine and H.S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, 5(1):118–124, 1989.