Parallelised Diffeomorphic
Sampling-based Motion Planning
Supplementary Material

Anonymous Author(s)
Affiliation
Address
`email`

1 Here we provide additional experimental details and results that are complementary to the main text.
2 When we refer to figures or tables in the main text, we will explicitly annotate them.

A Experimental details

4 All of the numerical experiments are conducted on a Dell laptop with an Intel i7-10750H at 5.0 GHz
5 with 12 cores CPU, 32GB RAM and an NVIDIA GeForce GTX 1650 Ti Mobile GPU.

6 The Robot Operating System (ROS) platform is used for the simulation, in which the *MoveitIt!*
7 framework is used as the simulator for manipulator and the Open Motion Planning Library (OMPL)
8 is used as the basis for our implementation for the sampling-based motion planners.

9 The robotic manipulator used in this experiemnt is the Kinova Jaco arm with a code name of
10 `j2n6s300`. The code name refers to the physical robotic arm settings, which is a jaco v2 6DOF
11 service 3 fingers. The robotic arm has 9 degree of freedom (*dof*) in total, which include 6 *dof*
12 in the arm and 3 *dof* in the gripper. The joint limits for the 1st, 4th, 5th and 6th joint are $[-\pi, \pi]$
13 and can wrap back without hard constrains; for the 2nd and 3rd joints are $[0.820305, 5.46288]$ and
14 $[0.331613, 5.95157]$ respectively; and for the fingers of the gripper the joint limits are $[0, 1.2]$.

B Numerical evaluation on the Diffeomorphic Sampling Distribution

16 Table 1 illustrates two metric on the numerical evaluation on our diffeomoprhic sampling distribution
17 compared to the uniform prior distribution that is commonly used in sampling-based motion planning.

Table 1: Numerical results on (i) the *total number of sampled configurations* (Num. samples) and
(ii) *percentage of feasible samples* (Feasible samples) on the original uniform distribution and the
parallelised diffeomorphic distribution. *Num. samples* denotes the total drawn samples within the
time budget, whereas *Feasible samples* denotes how good those samples are with regards to collisions.
Results are obtained by repeating 30 times each for both variants and across all three environments,
with a time budget of 20 seconds. Results shown are mean $\pm$ one standard deviation ($\mu \pm \sigma$).
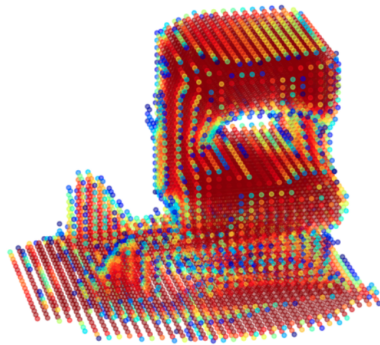
			Environment		
			Divider	Cupboard	Lab-setup
Original	Num. samples	**Total**	**9543 ± 907**	**22606 ± 1375**	**13153 ± 554**
	Feasibile samples	**Total**	**52.19 ± 0.41 %**	**12.77 ± 1.12 %**	**48.38 ± 0.86 %**
PDMP	Num. samples	**Total**	**10554 ± 751**	**23536 ± 1067**	**12985 ± 716**
		(from Prior)	*17 ± 11*	*5322 ± 1278*	*294 ± 120*
		(from Morphed)	*10536 ± 748*	*18214 ± 1190*	*12692 ± 736*
	Feasibile samples	**Total**	**81.52 ± 1.22 %**	**31.96 ± 2.23 %**	**72.83 ± 2.10 %**
		(from Prior)	*51.02 ± 0.85 %*	*12.98 ± 1.92 %*	*49.12 ± 0.51 %*
		(from Morphed)	*85.17 ± 1.74 %*	*37.51 ± 2.09 %*	*73.35 ± 1.82 %*

The results for PDMP in table 1 are broken down into samples that are contributed by the uninformed prior and by the diffeomorphic distribution. The prior is used when the concurrent queue (see Figure 2 in main text) is empty, which might be due to the PDMP is still warming up at the beginning or when sampled configurations are drawn at a rapid pace. This can be observed in the *Cupboard* environment, where there are a substantial amount of prior samples. Note that the percentage of morphed samples is still relatively high at 77% ($\frac{18214}{23536} \times 100\%$). This phenomenon might have occurred due to the low feasibility (highly clutter) nature of the environment with only 12% of samples being feasible in the original distribution. This in-turn renders the algorithm needing to draw more samples and spend less time in the actual tree-building and rewiring procedure of the algorithm.

Overall, by comparing *Num. samples* in table 1 it is clear that PDMP can always keep up with the rate of drawing samples even in the event of rapid samples drawing (in which it will revert back to prior due to the novel design in the architecture). The morphed samples are then more beneficial to the planning problem as they are more likely to be feasible in free space.

C   Continuous Differentiable Occupancy Map details

We learn a continuous occupancy representation using a neural network, mapping from coordinate points to the probability of the coordinate being occupied. The Neural network has the architecture:

| Input |
| Dense(3,150) |
| Tanh() |
| Dense(150,150) |
| Tanh() |
| Dense(150,150) |
| Tanh() |
| Dense(150,1) |
| Softmax() |
| Output |

(a) An example reconstructed set-up for our real-world experiments *Lab-setup*

(b) Architecture of our continuous occupancy map neural network model

where *Dense(n,m)* indicates a fully-connected layer with n inputs and m outputs. We optimise the neural network with respect to a binary cross entropy loss, via Adam with a step-size of 1×10^{-3}, for 625 epochs.

We treat the map as a binary classifier, and evaluate how well the map predict coordinate points in the task space. The performance of the occupancy representation in the environments used in our experiments, as measured by accuracy, area under the receiver operating characteristic curve (ROC-AUC), and precision is given in table 2. We observe that our neural network can accurately capture occupancy information in the environment.

	Divider	Cupboard	Tower
Accuracy	0.99	0.99	0.98
ROC-AUC	0.99	0.93	0.98
Precision	0.97	0.93	0.99

Table 2: The quality of representing environment occupancy via our neural network model

Next we pick the "Divider" environment and perform an ablation studying on the number of layers, and number of units in the hidden layers of our neural network model. To study the effect of the number of layers, we consider alternative models with 1 *Dense(150,150)* layer and 3 *Dense(150,150)* layers, whereas our setup had two of these layers. We also tune the hidden dimension of these layers, considering alternative models with hidden dimensions of 100 and 200, instead of the default 150. The results are tabulated in table 3. We observe that adjusting both the number of layers or the hidden dimension size does not drastically alter performance.

	1-layer	2-layers	3-layers	100 hidden units	200 hidden units
Accuracy	0.99	0.99	0.99	0.98	0.99
ROC-AUC	0.99	0.99	0.99	0.98	0.99
Precision	0.97	0.97	0.98	0.96	0.98

Table 3: Ablation results on the neural network model for occupancy

D Probabilistic Completeness

We shall demonstrate that drawing sample points from the transformed distribution maintains the probabilistic-completeness of the popular RRT sampling-based method.

We shall begin by considering the support of the prior distribution and the transformed distribution. Let P_y be the prior probability measure on space Y, with the diffeomorphism $F : Y \to Z$. Let $P_z(\mathbf{z}) := P_y(F^{-1}(\mathbf{z}))$, with $\mathbf{z} \in Z$

Definition D.1 (Support of probability measure). *Let P_y be a measure on a topological space Y, then the support of P_y is the set, $suppP_y := \{\mathbf{y} \in Y | P_y(\mathbf{y}) > 0\}$,*

Intuitively, the support of a probability distribution is the set of possible values of a random variable having non-zeros probability density. We shall study how the support of the prior and the transformed distribution change depending on F. In particular we consider when the support of the prior and transformed distributions are equal.

Lemma D.1 ([1] Equation 4). *The support of the prior and transformed probability measure are equal, i.e. $suppP_y = suppP_z$, if Y and Z are* homeomorphic *i.e., isomorphic as topological spaces.*

As $F : Y \to Z$ is a *diffeomorphism*, that is a differentiable homeomophism, $suppP_y = suppP_z$.

Theorem 1 (Probabilistic Completeness). *If a RRT-algorithm, drawing samples from random variable $\mathbf{y}$, is probabilistic complete, then the RRT-algorithm drawing samples from $f(\mathbf{y})$, where f is a diffeomorphism, is also probabilistic complete.*

Proof. As random variables $\mathbf{y}$ and $f(\mathbf{y})$ are linked by diffeomorphism f, by lemma D.1, they have the same support. Clearly, as the sampling time $t \to +\infty$, the set of created vertices with the sampling distribution $\mathbf{y}$, $V(\mathbf{y})$, and set of created vertices with the sampling distribution $f(\mathbf{y})$, $V(f(\mathbf{y}))$, are equal. Then by Theorem 23 from [2], the probabilistic completeness follows directly from the probabilistic completeness of RRT with sampling distribution $\mathbf{y}$. $\square$

E Numerical Evaluation of Motion Planners

Table 4 illustrates the numerical results of various motion planners with and without the proposing diffeomorphic sampler (complementary to Figure 4 in the main text). Overall, PDMP allows each motion planners to utilise sampled configurations that are more likely to be feasible (see table 1), which in turn allow PDMP planners to achieve shorter *time-to-solution* when compared to their original counterpart in table 4. Therefore, they are also more likely to successfully obtain a solution trajectory within the allocated time budget.

Table 4: Numerical results on various sampling-based motion planners on each environment. The *time-to-solution* refers to the time it took to obtain a solution trajectory (in seconds); and the *success pct.* refers to the percentage of runs that had successfully found a solution. The *Original* refers to the unmodified planner, whereas *PDMP* refers to the same planner with a replaced diffeomorphic sampler. Results are over 30 runs and with a time budget of 20 seconds. Results shown are mean $\pm$ one standard deviation ($\mu \pm \sigma$).
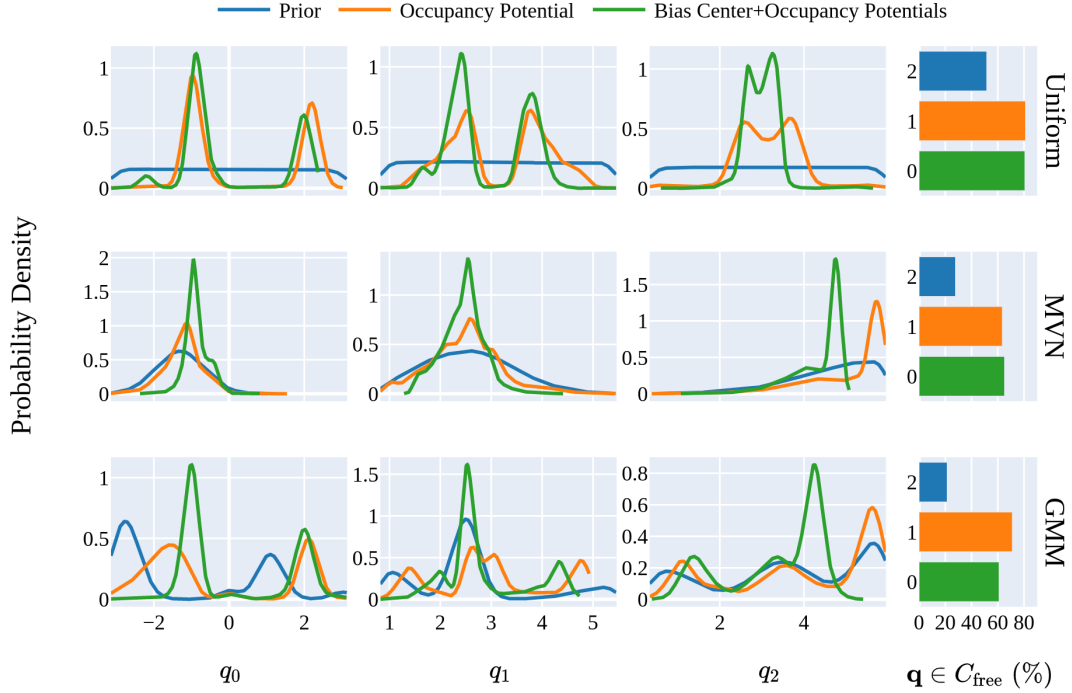
			Environment		
			Divider	Cupboard	Lab-setup
RRT*	Time-to-solution	Original	13.83 ± 7.72	18.79 ± 4.53	12.56 ± 7.78
		PDMP	5.99 ± 3.98	17.01 ± 6.14	9.26 ± 8.37
	Success pct.	Original	43.33 %	6.67 %	53.33 %
		PDMP	96.67 %	20.00 %	70.00 %
RRT-Connect*	Time-to-solution	Original	2.72 ± 0.91	5.51 ± 7.38	1.60 ± 0.63
		PDMP	1.60 ± 1.26	3.64 ± 5.76	1.29 ± 0.44
	Success pct.	Original	100 %	83.33 %	100 %
		PDMP	100 %	93.33 %	100 %
Lazy-PRM*	Time-to-solution	Original	14.98 ± 7.96	N/A	16.57 ± 6.93
		PDMP	13.37 ± 7.44	N/A	14.71 ± 7.07
	Success pct.	Original	33.33 %	0 %	20.00 %
		PDMP	43.33 %	0 %	46.67 %

F   PDMP with Various Prior Distribution
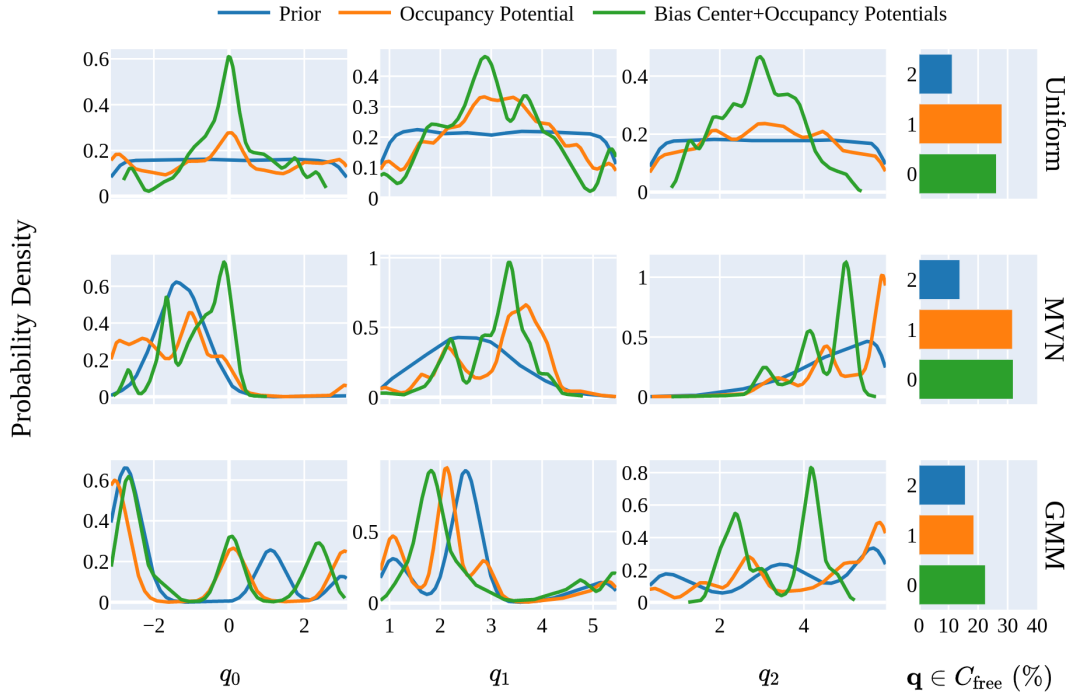
We extend the plots of the angles distribution across different prior distribution and environments in Fig. 2 (complementary to Figure 6 in the main text). The plots illustrate the empirical probability density of the joint angle distribution for the *Divider* (Fig. 3) and the *Cupboard* (Fig. 2b) environment. We used three different prior distribution: (i) a *Uniform* distribution, (ii) a *Multivariate Normal* distribution and (iii) a *Gaussian Mixture Model* to test the versatility of the diffeomorphic transformation. The parameters of (ii) and (iii) are randomly chosen and fixed across the two environments.

The blue *Prior* line plots refers the probability density of the prior distribution; the orange *Occupancy Potential* refers to the morphed samples with only occupancy potentials (i.e. the main results presented by this paper); and the green *Bias Center+Occupancy Potentials* illustrates morphed samples that includes both a potentials to morph towards the center of joints (to inject user bias), and the occupancy potentials to avoid collisions.

All of the green plots should be more concentrated to the center of the joint angles when compared to the other two. Sometimes it exhibits a higher feasibility even when compared to the orange *occupancy potential* plots, likely due to the fact that biasing towards center had avoided violating the joint limits (e.g. the 2nd and 3rd joints which does not wrap around). The orange *occupancy potential* plots tends to exhibit the highest feasibility across environments. Interestingly, the morphed samples formed a Multimodal distribution even when the prior is uniform, of which the multimodality likely resembles the likelihood of free space in the C-Space. The multimodality in Fig. 2b seems to be highly varying when compared to Fig. 3, which agrees with the low feasibilty results of the *Cupboard* environment in table 1 as the environment is extremely cluttered. Lastly, it is clear that the morphed distribution is highly resemblance of the prior distribution, which implies that the diffeomorphic transformation is flexible to be applied to any arbitrary prior distribution to inject user bias or avoid collision from the occupancy map.

Figure 2: Distribution of angles for the first three joint angles, for the uninformed prior, an informed distribution based on occupancy, and an informed distribution based on both user bias and occupancy (biasing towards center of joints, which is the middle of each x-axis). For each sub-figure, the rows from *top to bottom* refers to morphing different prior distributions, which includes a **Uniform**, a **Multivariate Normal (MVN)** and a **Gaussian Mixture Model (GMM)** prior. Parameters for MVN and GMM are randomly picked to be within the range of the joint limits. Each x-axis refers to joint angles (in radian) and y-axis refers to the emperical probability density function obtained with KDE. The right plots on both (a) and (b) refers to the percentage of samples that are feasible.

G Larger plot of the planner performance

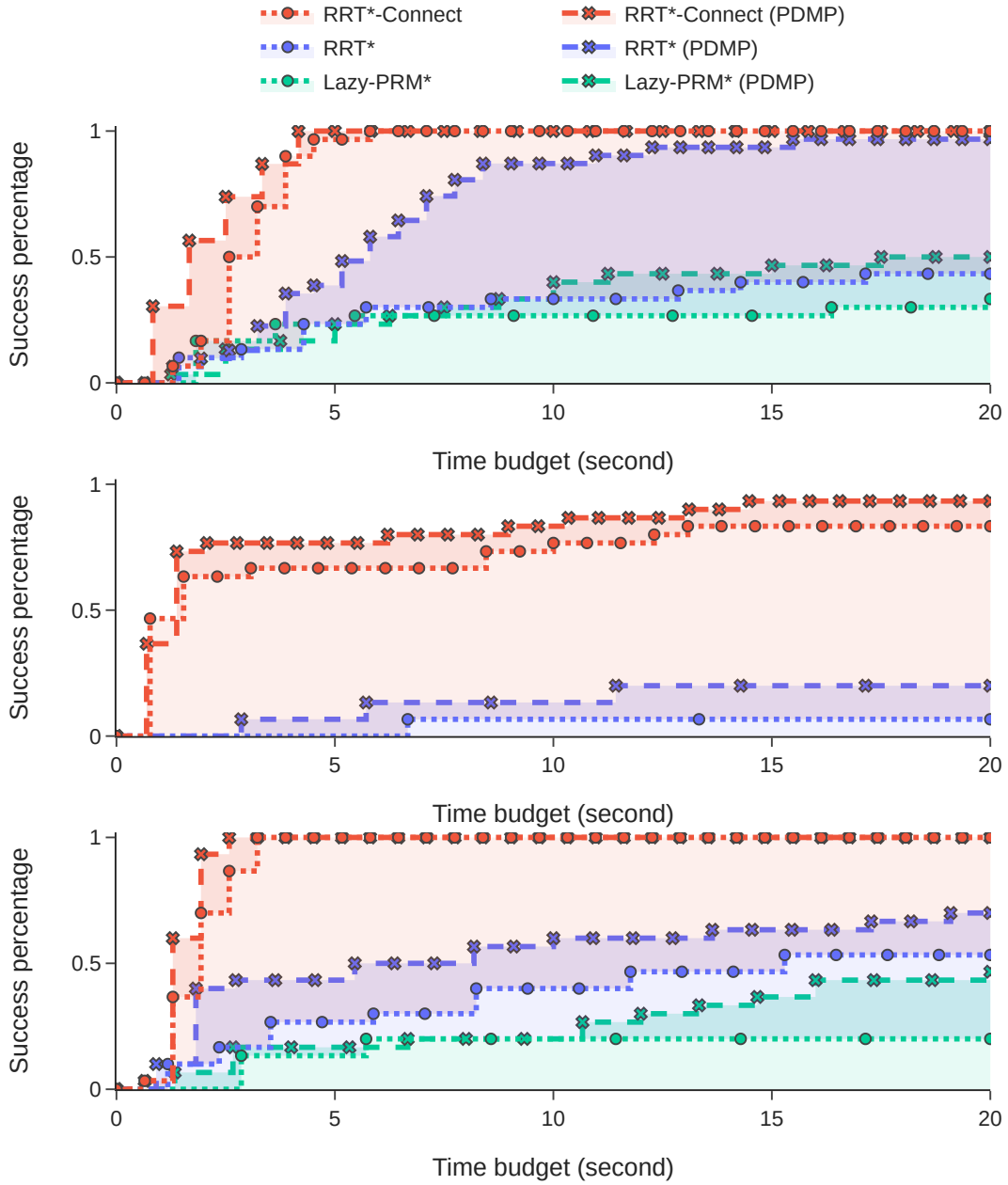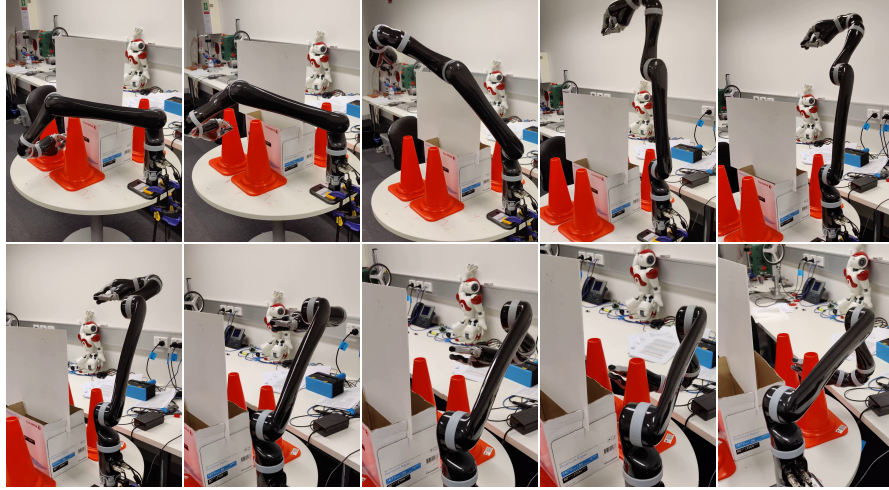Fig. 4 is a larger plot complementary to Figure 4 in the main text
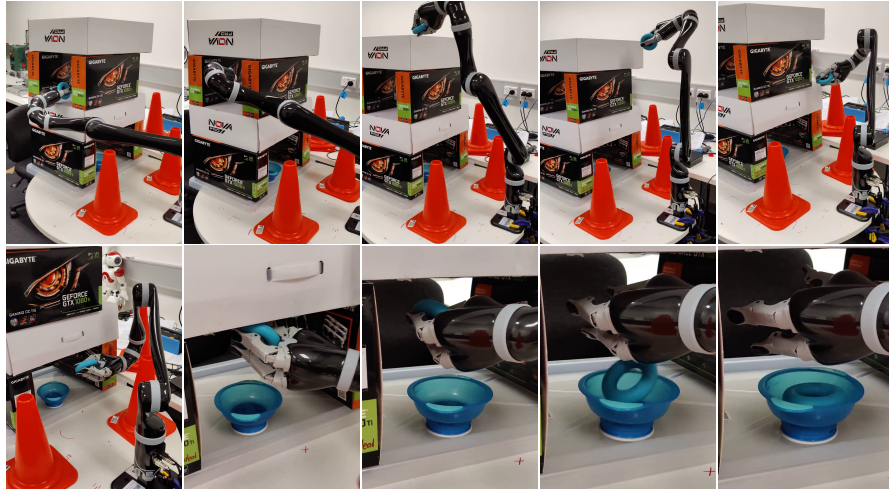

Figure 3: Planner performance

Figure 4: From top to bottom are (i) Divider, (ii) Cupboard and (iii) Lab-setup. The success rate of the planning algorithm variants, over 30 runs. We observe that PDMP enables all flavours of sampling-based motion planning algorithms to have improved success rates, particularly at lower planning times

H Real-world experiment

Fig. 5 visualises the sequence of motion of the planned solution trajectory by PDMP, in the *Divider*
and *Lab-setup* environment. Videos are also included in the supplementary material.

(a) Divider

(b) Lab-setup

Figure 5: Sequence of photos that illustrate the lab experiments with the Jaco arm.

References

[1] R. Cornish, A. L. Caterini, G. Deligiannidis, and A. Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *ICML*, 2020.

[2] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.