Motion planning in observations space with learned diffeomorphism models

Andrea Censi Adam Nilsson Richard M. Murray

Abstract— **We consider the problem of planning motions in observations space, based on learned models of the dynamics that associate to each action a diffeomorphism of the observations domain. For an arbitrary set of diffeomorphisms, this problem must be formulated as a generic search problem. We adapt established algorithms of the graph search family. In this scenario, node expansion is very costly, as each node in the graph is associated to an uncertain diffeomorphism and corresponding predicted observations. We describe several improvements that ameliorate performance: the introduction of better image similarities to use as heuristics; a method to reduce the number of expanded nodes by preliminarily identifying redundant plans; and a method to pre-compute composite actions that make the search efficient in all directions.**

I. Introduction

The "verification principle" [1] applied to robotics says that we will not be able to create robust robotic systems, unless we make them "aware" of their goals and assumptions, by providing them with the ability to learn, falsify, and relearn models of themselves and the world. Clearly, there is a wide spectrum of initial knowledge levels that is interesting to explore: the less an agent assume, the more it is robust, but the more arduous the learning problem becomes. In the bootstrapping scenario, we want to design agents that have no prior assumptions, except that they are embodied in a robot.

Previous work [2–4] has shown that it is possible to learn very low-level model of robotic sensorimotor cascades that are flexible enough to represent completely different sensors, such as range-finders and cameras, though we are far from the goal of capturing the entire set of all possible robots. Models are only as good as the task that they enable. So far, we have shown that using these models it is possible to solve tasks, such as fault detection, that only require "instantaneous" models. In this paper, we show that these models can be used for prediction over long-time horizons, and thus can be used for solving planning problems in observations space.

By planning in "observations space", we mean the problem of planning the motion based on models that act directly in the observations space (in this case, the set of images), as opposed to methods that work in the state space. This implies that there is a 1-to-1 map between state space and observation space, which means that there must not be hidden states, or that those hidden states can be neglected for the purpose of planning. In particular, we use models that associate to each action a diffeomorphism of the observations domain. These models capture the dominant dynamics of sensors such as cameras and range-finders [4].

A. Censi and R. M. Murray are with the Control & Dynamical Systems department, California Institute of Technology, Pasadena, CA. E-mail: {andrea, murray}@cds.caltech.edu. A. Nilsson is with the department of Automatic Control at Lund University, Sweden. E-mail: adam.nilsson.49@student.lth.se

Related work: Our goal is to formulate and solve the problem in full generality, with no assumptions about the set of diffeomorphisms, or the statistics of the observations. If one has more prior information about the system, then in the robotics literature and related fields there is a massive body of work addressing special cases of the problem. Visual servoing [5] techniques work with point/line features extracted from the images. This assumes that it is possible to define stable features that can be tracked from an image to the next. *Visual odometry* [6] can be seen as a formally very similar problem, though the use-cases are different enough to warrant different approaches. Most recently, there have been techniques that can work directly with pixel intensities [7–10]. These still assume that the transformation from world to image space is known.

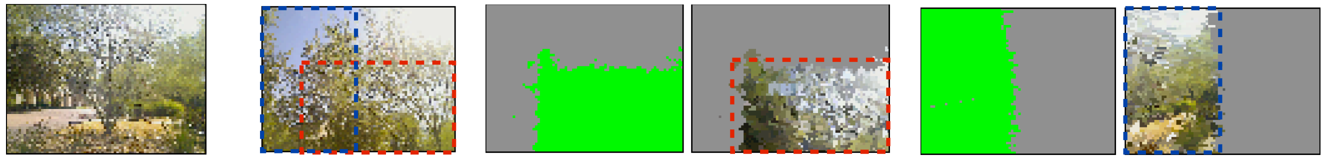
If the diffeomorphisms are small, then techniques from the medical image processing literature might be applicable (e.g., [11]). These techniques assumes that the deformation is small enough to be parametrized by infinitesimal vector fields, and do not cope well with uncertainty.

Paper outline: Section II recalls from [4] the techniques to represent and learn generic diffeomorphism models of robotic sensorimotor cascades. Section III formulates the planning problem, recalls established graph search techniques, and comments on the importance of the choice of the heuristics function. Section IV shows how planning efficiency is improved if explored plans are first reduced through an equivalence relation that can be inferred from the data. Section V shows the improvements following a principled way to choose a "basis" of composite actions that guarantee that all directions of the state spaces are uniformly explored. This dramatically improves the performance for nonholonomic systems. In the spirit of *reproducible research*, the source code, raw data, processed logs, learned models, and more complete statistics of the benchmarks can be found at the url http://purl.org/censi/2012/dptr1.

II. Diffeomorphism dynamical systems

Uncertain images and diffeomorphisms: Let $\mathcal{S}$ be a differentiable manifold. An *image* $\boldsymbol{y}$ is a function from $\mathcal{S}$ to some output space O. For an RGB camera, $O = [0,1]^3$. Call $\mathsf{Im}(\mathcal{S})$ the set of all images. To have a minimal representation of uncertainty, we define a set of "uncertain images" $\mathsf{UIm}(\mathcal{S})$, which are tuples $(\boldsymbol{y}, \boldsymbol{z})$, with $\boldsymbol{y} \in \mathsf{Im}(\mathcal{S})$ being a normal image, and $\boldsymbol{z} : \mathcal{S} \to \{0,1\}$ a binary "certainty" value. The function $\mathsf{vis} : \mathsf{UIm}(\mathcal{S}) \to [0,1]$ returns the percentage of the image that is certain.

Let $\mathsf{Diff}(\mathcal{S})$ be the set of diffeomorphism of $\mathcal{S}$. As we did with images, define an enlarged space of "uncertain" diffeomorphisms $\mathsf{UDiff}(\mathcal{S})$, which contains tuples (φ, γ), with $\varphi \in \mathsf{Diff}(\mathcal{S})$ and $\gamma : \mathcal{S} \to \{0,1\}$. It is possible to learn more



(a) $\boldsymbol{y}_{\text{start}}$ (b) $\boldsymbol{y}_{\text{goal}}$ (c) $\mathsf{p\_to\_d}(p_\circ) \cdot \boldsymbol{y}_{\text{start}}$ (d) $\mathsf{p\_to\_d}(p^\star) \cdot \boldsymbol{y}_{\text{start}}$

Fig. 1. Visibility constraints, uncertainty due to learned models, and ambiguities are the main phenomena that we have to deal with in this planning problem. Due to the limited field of view, our models must be able to represent the uncertainty in the predictions. The uncertainty is quantized into a binary quantity (green: parts of the image that it is possible to predict). A problem instance is a pair of images $\boldsymbol{y}_{\text{start}}$ and $\boldsymbol{y}_{\text{goal}}$. In this benchmark we have available the ground truth trajectory that the camera underwent, which we call $p_\circ \in \mathsf{Plans}$. Due to ambiguities of the image and uncertainty in the learned models, sometimes the algorithms find solutions that are better than the ground truth. In this case, the ground truth $p_\circ$ corresponds to moving left, then down, while the solution $p^\star$ found corresponds to moving right, and gives a smaller error in image space.

detailed models of uncertainty for diffeomorphisms, but for the purposes of this paper, it is sufficient to use binary uncertainty. This is the simplest representation that allows reasoning on constraints given by the field of view, and represent what is possible to predict and what is not (Fig. 1c).

The sets $\mathsf{UIm}(\mathcal{S})$ and $\mathsf{UDiff}(\mathcal{S})$ are meant to be tractable slices of $\mathsf{ProbMeasures}(\mathsf{Im}(\mathcal{S}))$ and $\mathsf{ProbMeasures}(\mathsf{Diff}(\mathcal{S}))$.

Diffeomorphism dynamics: Let $\mathcal{U}$ be a set of discrete "commands" or "actions". A *DDS* is a discrete-time dynamical system with input in $\mathcal{U}$ and state space $\mathsf{Im}(\mathcal{S})$. For a given command $\boldsymbol{u} \in \mathcal{U}$, let $\mathsf{p\_to\_d}(\boldsymbol{u}) \in \mathsf{UDiff}(\mathcal{S})$ be a function that associates a diffeomorphism to each action (mnemonics: *p-to-d* as in *plan to diffeomorphism).* The state $\boldsymbol{x}_k \in \mathsf{Im}(\mathcal{S})$ evolves according to the dynamics

$$\begin{aligned} x_{k+1}(s) &= x_k(\varphi_k(s)), \\ \varphi_k &= \mathsf{p\_to\_d}(\boldsymbol{u}_k). \end{aligned}$$

This can be written in a more compact way as

$$\boldsymbol{x}_{k+1} = \mathsf{p\_to\_d}(\boldsymbol{u}_k) \circ \boldsymbol{x}_k.$$

The observations are a function $\boldsymbol{y} : V \to O$, where $V \subset \mathcal{S}$ is the "viewport", or the portion of the space $\mathcal{S}$ that is actually observable. For compactness of notation, we assume that the observations $\boldsymbol{y}$ are actually a function on the whole space $\mathcal{S}$ by assuming that they are set to a null value in $\mathcal{S} \backslash V$.

Learning diffeomorphisms: The method in [4] learns diffeomorphisms by estimating their point-wise values. The viewport V is sampled in a grid at points $\{s^i\}_{i=1}^N \subset \mathcal{S}$ (i.e., pixels). To represent a discretized diffeomorphism on $\mathcal{S}$, for each point $s^i \in S$, the learning method estimates the coordinates of the closest point to $\varphi(s^i)$. The estimation is independent for each point. While we know that the underlying dynamics is composed by diffeomorphisms, we do not represent continuity/smoothness constraints in the discretized representation.

In this paper, we focus explicitly on the planning problem and that we are given the values $\mathsf{p\_to\_d}(\boldsymbol{u}) \in \mathsf{UDiff}(\mathcal{S})$, for $\boldsymbol{u} \in \mathcal{U}$. However, many considerations are explicitly meant to deal with the fact that these diffeomorphisms are learned from data. In particular, we have to deal with two main problems: 1) these are noisy models; 2) they are described nonparametrically, not symbolically.

III. Planning in observations space

This section formally states the planning problem and gives baseline results for graph-search methods.

A. Plans and induced diffeomorphisms

Define the set of plans as the set of all sequences of elements of $\mathcal{U}$, including the empty sequence:

$$\mathsf{Plans} = \emptyset \cup \mathcal{U} \cup \mathcal{U}^2 \cup \mathcal{U}^3 \cup \dots.$$

Let $p_2 \circ p_1$ be the composition of two plans, where p_1 is executed *before* p_2. (This right-to-left definition matches the usual notation of function composition).

For a plan $p \in \mathsf{Plans}$ and a given DDS, define $\mathsf{p\_to\_d}(p) \in \mathsf{UDiff}(\mathcal{S})$ to be the uncertain diffeomorphism that is associated to the plan. We define and compute $\mathsf{p\_to\_d}$ recursively as follows. For the empty plan, let $\mathsf{p\_to\_d}(\emptyset) = \mathsf{Id}_\mathcal{S}$, where $\mathsf{Id}_\mathcal{S}$ is the identity on $\mathcal{S}$. For plans of length 1, we have assumed to have been given $\mathsf{p\_to\_d}(\boldsymbol{u})$ for all $\boldsymbol{u} \in \mathcal{U}$. For plans of length 2 and above, $\mathsf{p\_to\_d}$ can be defined recursively using the fact that it is a homomorphism from Plans to $\mathsf{UDiff}(\mathcal{S})$:

$$\mathsf{p\_to\_d}(p_2 \circ p_1) = \mathsf{p\_to\_d}(p_2) \circ \mathsf{p\_to\_d}(p_1).$$

In this expression, the "$\circ$" on the left is plan concatenation, while the "$\circ$" on the right is the usual function composition, in the case of certain diffeomorphisms, and must be interpreted as probability density convolution on Lie groups [12], in the case of uncertain diffeomorphisms. Given a plan p and an image $\boldsymbol{y} \in \mathsf{UIm}(\mathcal{S})$, define

$$p \cdot \boldsymbol{y} = \mathsf{p\_to\_d}(p) \cdot \boldsymbol{y} \in \mathsf{UIm}(\mathcal{S})$$

as the predicted observations after executing p.

B. Formal statement of the planning problem

Problem 1 (Planning in observations space): Given:

1) *A discrete set of commands $\mathcal{U}$;*
2) *The values of $\mathsf{p\_to\_d}(\boldsymbol{u})$ for all $\boldsymbol{u} \in \mathcal{U}$;*
3) *A distance d on $\mathsf{UIm}(\mathcal{S})$;*
4) *Two images $\boldsymbol{y}_{start} \in \mathsf{Im}(\mathcal{S})$ and $\boldsymbol{y}_{goal} \in \mathsf{Im}(\mathcal{S})$;*
5) *A minimum visibility threshold v_0;*
6) *A maximum distance threshold d_g;*

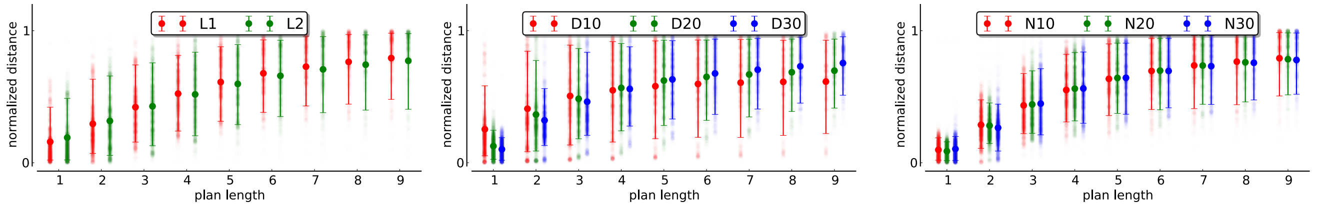
find a plan $p^\star \in \mathsf{Plans}$ such that the predicted image $p^\star \cdot \boldsymbol{y}$ according to $p^\star$ has at least visibility v_0:

$$\mathsf{vis}(p^\star \cdot \boldsymbol{y}_{start}) \geq v_0, \tag{1}$$

and it is no farther than d_g from the goal image:

$$d(p^\star \cdot \boldsymbol{y}_{start}, \boldsymbol{y}_{goal}) \leq d_g.$$

We do not include in this formulation a concept of "obstacles" nor we optimize on the length of the plan.



(a) Statistics for $d^{\mathsf{Im}}_{L_1}$ and $d^{\mathsf{Im}}_{L_2}$ (b) Statistics for $d^{\mathsf{Im}}_{D_\alpha}$, with $\alpha = 10\%, 20\%, 30\%$. (c) Statistics for $d^{\mathsf{Im}}_{N_\alpha}$, with $\alpha = 10\%, 20\%, 30\%$.

Fig. 2. Comparison heuristics for the planning problem. Each subfigure shows the distribution of $d(\boldsymbol{y}_1, \boldsymbol{y}_2)$ for a certain distance d, as a function of the length of the plan that takes from $\boldsymbol{y}_1$ to $\boldsymbol{y}_2$, over the whole image sequence that we use for learning (approximately 22 minutes of data, at 1000 images, with snapshots taken at 1 second intervals). Note that these distances have different units: $d^{\mathsf{Im}}_{D_\alpha}$ is measured in units of $\mathcal{S}$, while the others are measured in units of O. To overcome this problem, we normalize the data using the order statistics. More in details, for each distance d, we first compute $h(d, \delta) = \{d(\boldsymbol{y}_1, \boldsymbol{y}_2) \mid \boldsymbol{y}_1, \boldsymbol{y}_2 \in P(\delta)\}$, where $P(\delta)$ is the set of all pairs of images at plan distance δ. Then we compute $H(d) = \cup_\delta h(d, \delta)$, the set of all distances. Finally, we normalize using the order statistics, by computing and plotting $h'(d, \delta) = \mathrm{order}(h'(d, \delta), H(d)) \in [0, 1]^{|P(\delta)|}$, where $\mathrm{order}(x, \boldsymbol{v})$ returns the order of a number x in the vector $\boldsymbol{v}$. This procedure makes the comparison invariant to any monotonic transformations of the distances: $d'(\boldsymbol{y}_1, \boldsymbol{y}_2) = f(d(\boldsymbol{y}_1, \boldsymbol{y}_2))$ with $f \in \mathsf{Diff}_+(\mathbb{R}^+_\circ)$. Note that the $d^{\mathsf{Im}}_{N_\alpha}$ distances are much better predictors of the distance to the goal than the L_1 and L_2 distances. For example, it is possible to discriminate exactly between 1 and 4 steps.

Fig. 1 shows an example instance of a problem for the system that we use for most of the experiments. Our "robot" is a Logitech *Orbit* camera which can be controlled with pan/tilt commands[1]. We learn a DDS model from the logs recorded while the camera underwent random motions, using 4 actions, corresponding to pan left, pan right, tilt up, and tilt down. In image space, these actions correspond, as a first approximation, to horizontal/vertical translation, though the estimated diffeomorphisms are precise enough to capture the spherical lens distortion.

A problem instance consists of an initial image $\boldsymbol{y}_{\text{start}}$ (Fig. 1a) and a goal image (Fig. 1b). Because we sample these images from the logs, we know the ground truth plan $p_\circ$ that was executed. Note, however, that the planning objective is not to recover the ground truth plan $p_\circ$, but rather to find *any* plan $p^\star$ that makes the prediction $p^\star \cdot \boldsymbol{y}_{\text{start}}$ match with $\boldsymbol{y}_{\text{goal}}$. Due to the limited field of view and the uncertainty of the learned models, it is common to have ambiguous solutions (Fig. 1). The constraint on the visibility (1) is essential to maintain the problem well-posed. For each instance, the thresholds v_0 and d_g are computed using the prediction given the ground truth $p_\circ$, to assure that a solution exists for each instance[2].

C. Measuring similarities

In this problem there are several objects for which we need a way to measure similarity: the space of plans Plans, the space of images $\mathsf{Im}(\mathcal{S})$, the space of diffeomorphism $\mathsf{Diff}(\mathcal{S})$, and their uncertain counterparts $\mathsf{UDiff}(\mathcal{S})$ and $\mathsf{UIm}(\mathcal{S})$. To simplify the discussion, we use the word "distance" in an informal way. Most of the similarities are actually distances (they satisfy the triangle inequality) or could be made so with only slight modification of the definition.

[1]One reason this sensor was chosen is that it is the cheapest ($100) off-the-shelf actuated sensor that we know. It is controlled through a ROS interface. In addition to the data that we used, on the website we give one-click* scripts to collect raw data logs from the sensor, preprocess the data to a given resolution, extract tuples of the kind $(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{y}_{k+1})$ that can be used for learning, run the diffeomorphism learning code, and prepare benchmarks for planning. * Disclaimer: it is "one-click" only after many clicks for installation.

[2]Strictly speaking, the visibility threshold v_0 contains information about the solution $p_\circ$, as it is a function of the plan length. However, none of the algorithms we consider uses this side information to "cheat".

1) Measuring on $\mathsf{Diff}(\mathcal{S})$: The domain $\mathcal{S}$ is a Riemannian manifold, so it comes equipped with a metric $d^{\mathcal{S}}$. Given this metric, we can define a metric on $\mathsf{Diff}(\mathcal{S})$ as:

$$d^{\mathsf{Diff}(\mathcal{S})}_{L_1}(\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2) = \int_{\mathcal{S}} d^{\mathcal{S}}(\varphi_1(s), \varphi_2(s)\,\mathrm{d}s.$$

2) Measuring on $\mathsf{Im}(\mathcal{S})$: Distances on $\mathsf{Im}(\mathcal{S})$ and $\mathsf{UIm}(\mathcal{S})$ are used both for defining thresholds for success, and as heuristics during planning. The usual norm-induced distances on the function space $\mathsf{Im}(\mathcal{S})$ are

$$d^{\mathsf{Im}}_{L_p}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \tfrac{1}{|\mathcal{S}|}\Big(\int_{\mathcal{S}} |y_1(s) - y_2(s)|^p \;\mathrm{d}s\Big)^{\frac{1}{p}}.$$

How can we judge the quality of a heuristics? The ideal heuristics would be a function of the distance in terms of plans: $d(\boldsymbol{y}, \boldsymbol{y}_{\text{goal}}) = f(L)$, where L is the length of the shortest plan to go from $\boldsymbol{y}$ to $\boldsymbol{y}_{\text{goal}}$ and f is some monotonic function. We cannot require this to hold exactly (if this held exactly, we would not need to solve the planning problem, as we could just follow the gradient of the distance), but it is reasonable to ask for this to hold at least for small L. According to this criterion, $d^{\mathsf{Im}}_{L_1}$ and $d^{\mathsf{Im}}_{L_2}$ are not good heuristics for this problem (Fig. 2a—see the caption for some important details on how to compare these distances).

This motivates the search for other distances. Our intuition was that the distances must be robust to some residual diffeomorphism, in the sense that $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ should be considered close if $\boldsymbol{y}_2 = \varphi \circ \boldsymbol{y}_1$ for a "small" diffeomorphism φ. The family of distances $d^{\mathsf{Im}}_{D_\alpha}$ looks at pairwise distances between similar pixels in images. For each pixel $s \in \mathcal{S}$ in the first image, we find the most similar pixel in the second image, looking in the area $A_\alpha(s)$ around s, which is a fraction $\alpha \ll 1$ of the size of the whole domain:

$$A_\alpha(s) = \{s' \in \mathcal{S} \mid d^{\mathcal{S}}(s, s') \leq \alpha|\mathcal{S}|\}.$$

We then average these distances over the domain:

$$d^{\mathsf{Im}}_{D_\alpha}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \tfrac{1}{|\mathcal{S}|}\int_{\mathcal{S}} d^{\mathcal{S}}(s, \operatorname*{arg\,min}_{v \in A_\alpha(s)} |y_1(s) - y_2(v)|)\,\mathrm{d}s.$$

It is easy to see that $d^{\mathsf{Im}}_{D_\alpha}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ is 0 if the two images are related by a diffeomorphism smaller than α.

We observed that a simpler variation works better. Instead of averaging the distances between similar pixels, the distance $d^{\mathsf{Im}}_{N_\alpha}$ averages the minimum dissimilarity in $A_\alpha(s)$:

$$d^{\mathsf{Im}}_{N_\alpha}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \frac{1}{|\mathcal{S}|} \int_{\mathcal{S}} \min_{v \in A_\alpha(s)} |y_1(s) - y_2(v)| \, \mathrm{d}s.$$

This distance is more stable with respect to the choice of the parameter α, and that it is the most predictive of the length of the plan between the observations (Fig. 2c).

3) Measuring on $\mathsf{UIm}(\mathcal{S})$ *and* $\mathsf{UDiff}(\mathcal{S})$*:* Finally, because we have to deal with a scalar uncertainty, we can lift a distance on $\mathsf{Im}(\mathcal{S})$ to $\mathsf{UIm}(\mathcal{S})$ by simply weighting by the scalar uncertainty. For a given distance that can be expressed as the integral of an error field $\int_{\mathcal{S}} e_d(s) \, \mathrm{d}\mathcal{S}$, we define the corresponding weighted distance $\overline{d}$ as

$$\overline{d}\left(\langle \boldsymbol{y}_1, \boldsymbol{z}_1 \rangle, \langle \boldsymbol{y}_2, \boldsymbol{z}_2 \rangle\right) = \frac{\int z_1(s) z_2(s) e_d(s) \, \mathrm{d}s}{\int z_1(s) z_2(s) \, \mathrm{d}s}.$$

Likewise, distances on $\mathsf{Diff}(\mathcal{S})$ are lifted to $\mathsf{UDiff}(\mathcal{S})$.

D. Planning with graph search

We shall adapt existing planning techniques to our problem. We consider several variations of bidirectional search [13, Chapter 2], which are summarized in Table I. It is not immediate to adapt sampling-based planning, such as RRT-based methods, because we do not have the ability to sample a "random" state (in our case, a random image).

The bidirectional search algorithm constructs two trees in the state space (which is, in our case, $\mathsf{UIm}(\mathcal{S})$), one with root in the start state (in our case, $\boldsymbol{y}_{\text{start}}$), and one with root in the goal state (in our case, $\boldsymbol{y}_{\text{goal}}$). The two trees are grown at each iteration, until they "touch" according a given metric.

There are many variations according to the policy for node expansion. In *breadth-first* search, trees are expanded in all directions uniformly. A *greedy* approach expands the node which is closest to the root of the other tree. The *"greedy-tree"* approach expands the node on one tree which is closest to the other tree. This implies comparing the node with all nodes in the other tree. This makes sense only if, as in our case, computing distances is much cheaper than expanding nodes.

Computation details: The software is written in Python. The dominant cost consists in applying diffeomorphisms to images. Judiciously using vectorized operations on flattened arrays, the cost of applying a generic diffeomorphism (i.e., a generic permutation of the pixels), is in the order of 0.2 seconds for 128x128 images. There is probably a $10\times$ speed-up available if coded in C, by being more careful about memory allocation, and writing the operations in a more cache-friendly manner; and probably another $10\times$ speed-up by using a GPU implementation, which has been shown to substantially help in analogous problems [14].

Results: Table II shows the algorithms performance in problem instances such as those shown in Fig. 1, randomly sampled from our logs. As expected, uninformed unidirectional search (algorithm GNB) is the least efficient, and, given our constraints on resources (5 minutes of execution time) succeeds in only 40% of the cases, corresponding to the "easy" instances. Breadth-first bidirectional search (BNB) does slightly better at 80% success. As expected, given our considerations in Section III-C.2, the informed search algorithms (BNG, BNT) do better, succeeding in 100% of the cases.

TABLE I

PLANNING ALGORITHMS COMPARED IN THIS PAPER

	directions	*use* PlanReduce*?*	*node priority*	*composite actions*
GNB	1		breadth-first	
BNB	2		breadth-first	
GEB	1	✓	breadth-first	
BEB	2	✓	breadth-first	
BEG	2	✓	greedy	
BET	2	✓	greedy-tree	
BETc	2	✓	greedy-tree	✓

Each algorithm is named with an acronym of the form {G,B}{N,E}{B,G,T}. {G,B}: either one-directional ("G") or bidirectional ("B") search; {N,E}: "E" when using the equivalence relation induced by PlanReduce (Section IV), "N" otherwise. {B,G,T} indicate the priority for node expansion: "B" stands for breadth-first (uninformed); "G" stands for greedy (on either tree, the node with the image closest to the root of the other tree is expanded first); and "T" for greedy tree (the node which is closest to the any node on the other tree is expanded first). BETc is a variant of BET using the composite actions (Section V).

TABLE II

PERFORMANCE OF ALL ALGORITHMS FOR THE CASE $|p_\circ| = 7$

	success	mean $|p^\star|$	mean time (s)	mean # nodes
GNB	40%	1.8 †	87.1	208
BNB	80%	4.4	50.2	168
BNG	100%	5.4	2.8	24
BNT	100%	5.4	2.3	22
GEB	100%	4.6	4.7	57
BEB	100%	4.9	3.3	39
BEG	100%	5.1	2.0	19
BET	100%	5.4	1.8†	18†

†: Note that sometimes there exists a very short alternative solution to the ground truth. The very inefficient GNB gets 40% of success due to those instances, which give a mean length of 1.8 steps. ‡: As expected in this scenario, the BET algorithm is the most efficient for time and resources.

IV. Increasing the planning performance using equivalent plan reduction

It is possible to increase the performance of the planning algorithms by exploiting some of the structure of the dynamics that can be inferred a posteriori from the learned (or given) diffeomorphisms. Section IV-A and IV-B recall recall the set of action predicates previously introduced [4] and how they can be estimated from data. Section IV-C describes how this knowledge can be used to design an algorithm that collapses equivalent plans to a canonical form. Section IV-D shows that introducing this variation in the node expansion greatly improves planning performance.

A. Actions predicates

Suppose that the (unobservable) underlying state evolves according to $x_{k+1} = f(x_k, \boldsymbol{u}_k)$. We identify several predicates on the set of commands:

1) Actions that have no effect:

$$\mathsf{void}(\boldsymbol{u}) \quad \Leftrightarrow \quad f(x, \boldsymbol{u}) = x.$$

These can be safely excluded from planning.

TABLE III

PERFORMANCE OF BET FOR SAMPLES OF DIFFERENT LENGTH

	success	mean $|p^\star|$	mean time (s)	mean # nodes
$|p_\circ| = 1$	100%	1.0	0.3	8
$|p_\circ| = 2$	100%	1.8†	0.5	8
$|p_\circ| = 3$	100%	2.6†	0.7	11
$|p_\circ| = 4$	100%	5.1†	12.2‡	21
$|p_\circ| = 5$	100%	4.8	1.5	18
$|p_\circ| = 6$	100%	6.0	2.5	22
$|p_\circ| = 7$	100%	5.4	1.8	18
$|p_\circ| = 8$	100%	7.4	15.1	30
$|p_\circ| = 9$	100%	7.8	8.7	29
$|p_\circ| = 10$	100%	8.8	8.6	35
$|p_\circ| = 11$	§80%	10.6	30.5	50
$|p_\circ| = 12$	§90%	7.3	16.3	32

†: Note that the length of the solution is possibly smaller or longer than the plan of the ground truth $p_\circ$ (Fig. 1).
‡: In a few cases, the algorithms get trapped in a large basin containing a local minimum; eventually they exit, but this drives up the average times.
§: For 11–12 steps, depending on the trajectory, there is only minimal overlap between goal and start image, making the problem not well posed.

Algorithm 1 planreduce

```
function PlanReduce(p ∈ Plans):
    return pr(p, ∅)

function pr(p2 ∈ Plans, p1 ∈ Plans)
  write p1 as a ∘ A
  write p2 as B ∘ b
  if inverse(b, a): # If the two actions are inverse...
    return pr(B, A) # ...they both disappear.
  if commute(b, a)
    # Now a comes after b. Would the plan be shorter
    # if we put b before a?
    aA ← pr(a, A)
    bA ← pr(b, A)
    if len(aA) < len(bA): # current order is fine
      return pr( B, b ∘ aA)
    else if len(aA) > len(bA): # use alternative
      return pr( B, a ∘ bA)
    else: # len(aA) ==len(bA), choose according to ordering
      if b≺ a:
        return pr(B, b ∘ aA)
      else:
        return pr(B, a ∘ bA)
```

2) Pairs of actions that have the same effect:

$$\mathsf{same}(\boldsymbol{u}_1, \boldsymbol{u}_2) \quad \Leftrightarrow \quad f(x, \boldsymbol{u}_1) = f(x, \boldsymbol{u}_2).$$

If two actions have the same effect then it is not necessary to include both in planning.

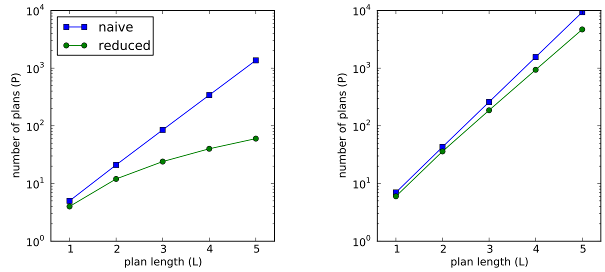
3) Pairs of actions that are left/right inverses:

$$\mathsf{inverse}(\boldsymbol{u}_1, \boldsymbol{u}_2) \quad \Leftrightarrow \quad f(f(x, \boldsymbol{u}_1), \boldsymbol{u}_2) = x.$$

If two actions u_1 and u_2 have the inverse effect, then we can exclude plans which contain the subplan $u_2 \circ u_1$, as there is always shorter plan achieving the same effect. Note that, in general, right inverse does not imply left inverse ($\mathsf{inverse}(\boldsymbol{u}_1, \boldsymbol{u}_2) \nLeftarrow \mathsf{inverse}(\boldsymbol{u}_2, \boldsymbol{u}_1)$).

4) Pairs of actions that commute:

$$\mathsf{commute}(\boldsymbol{u}_1, \boldsymbol{u}_2) \quad \Leftrightarrow \quad f(f(x, \boldsymbol{u}_1), \boldsymbol{u}_2) = f(f(x, \boldsymbol{u}_2), \boldsymbol{u}_1).$$



(a) Orbit camera (b) Dubin's car

Fig. 3. The PlanReduce algorithm allows to exclude redundant plans, as implied by the predicates described in Section IV-A. For a system with N actions, the size of the set Plans_L which contains all plans up to length L is N^L (blue curve). (a) In the case of the Orbit camera, for which all actions commute, each action has an inverse, and each pair of actions explores a different direction in the state space, the number of reduced plans $\mathsf{RedPlans}_L = \mathsf{PlanReduce}(\mathsf{Plans}_L)$ grows with the volume of the state space explored. If the state space has dimension K, then the volume is proportional to L^K (green curve). (b) In the case of the Dubin's car, where not all actions commute, the size of $\mathsf{RedPlans}_L$ still grows exponentially, though with a smaller constant factor.

If two actions u_1 and u_2 commute, then it is not necessary to consider plans of the pattern $p_2 \circ u_2 \circ u_1 \circ p_1$, if the plan $p_2 \circ u_1 \circ u_2 \circ p_1$ has already been considered.

B. Inferring pairwise actions relations

We can infer these predicates from the learned diffeomorphisms, though there is some tuning constant to be adjusted. Let d be a chosen distance on $\mathsf{UDiff}(\mathcal{S})$.

1) An action $\boldsymbol{u}$ is void if $\mathsf{p\_to\_d}(\boldsymbol{u})$ is close to the identity:

$$\mathsf{void}(\boldsymbol{u}) \quad \Leftrightarrow \quad d(\mathsf{p\_to\_d}(\boldsymbol{u}), \mathsf{Id}_{\mathcal{S}}) \leq c\, d_0. \tag{2}$$

2) Two actions $\boldsymbol{u}_1, \boldsymbol{u}_2$ are the same if $\mathsf{p\_to\_d}(\boldsymbol{u}_1)$ and $\mathsf{p\_to\_d}(\boldsymbol{u}_2)$ are close:

$$\mathsf{same}(\boldsymbol{u}_1, \boldsymbol{u}_2) \quad \Leftrightarrow \quad d(\mathsf{p\_to\_d}(\boldsymbol{u}_1), \mathsf{p\_to\_d}(\boldsymbol{u}_2)) \leq c\, d_0. \tag{3}$$

3) Two actions are inverse of each other if their diffeomorphisms are inverse of each other:

$$\mathsf{inverse}(\boldsymbol{u}_1, \boldsymbol{u}_2) \quad \Leftrightarrow \quad d(\mathsf{p\_to\_d}(\boldsymbol{u}_1), \mathsf{p\_to\_d}(\boldsymbol{u}_2)^{-1}) \leq c\, d_0. \tag{4}$$

4) Two actions commute if the diffeomorphism associated to $\boldsymbol{u}_1 \circ \boldsymbol{u}_2$ is similar to the one associated to $\boldsymbol{u}_2 \circ \boldsymbol{u}_1$:

$$\mathsf{commute}(\boldsymbol{u}_1, \boldsymbol{u}_2) \quad \Leftrightarrow$$

$$d(\mathsf{p\_to\_d}(\boldsymbol{u}_1 \circ \boldsymbol{u}_2), \mathsf{p\_to\_d}(\boldsymbol{u}_2 \circ \boldsymbol{u}_1)) \leq c\, d_0. \tag{5}$$

In these expressions, the scaling constant $d_0 > 0$ is computed as the maximum size of the diffeomorphisms:

$$d_0 = \max_{\boldsymbol{u}} d(\mathsf{p\_to\_d}(\boldsymbol{u}), \mathsf{Id}_{\mathcal{S}})). \tag{6}$$

The dimensionless tuning constant $c \ll 1$ accounts for the noise and the approximation that we can accept.

C. Reducing plans to their canonical form

To incorporate this knowledge information into the planning algorithms, we devise an algorithm $\mathsf{PlanReduce} : \mathsf{Plans} \to \mathsf{Plans}$ that takes a given plan, and, based on the knowledge of these predicates, returns a possibly shorter plan which has the same associated diffeomorphism.

The listing given for PlanReduce (Algorithm 1) only shows how to deal with the commute and inverse relations. (It is trivial what to deal with void commands: just do not include them in the planning. Likewise, if there are commands that are equivalent, then just include only one of them in the set of available commands.) The algorithm is written here in tail-recursive form with two parameters p_1 and p_2, which makes it easy to analyze, though an implementation (and the one available on the website) can be written as a loop where p_1 and p_2 are two stacks. At all times, the invariant that is preserved is that $\mathsf{p\_to\_d}(p_2 \circ p_1) = \mathsf{p\_to\_d}(p)$.

Dealing with inverses (line 7) is easy: if $p_2 = B \circ b$ and $p_1 = a \circ A$ and $\mathsf{inverse}(a, b)$, then a and b cancel each other and we can continue with $p_2 = B$ and $p_1 = A$.

Dealing with commands that commute is slightly more involved. To obtain a well-defined behavior, it is necessary to assume that there is a total ordering "$\prec$" on the space $\mathcal{U}$, because, if $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ commute, then there needs to be a way to choose between $\boldsymbol{u}_2 \circ \boldsymbol{u}_1$ and $\boldsymbol{u}_1 \circ \boldsymbol{u}_2$ as the canonical plan to return. At line 7, given a plan of the form $B \circ b \circ a \circ A$, where a and b commute, we can choose to replace $b \circ a \circ A$ with $a \circ b \circ A$. If one choice gives a shorter reduced subplan (found through a recursive call in lines 12–13), then we choose the shorter, otherwise the total order $\prec$ is used to decide.

The algorithm's complexity is $\mathcal{O}(|p|)$ if memoization [15] is used to remember the reductions already computed. (Because of the recursive calls at lines 12–13, if memoization is not used it might have exponential cost in $|p|$.) In our scenario, the cost of this algorithm, which deals only with sequences of integers, is negligible with respect to the cost of applying a diffeomorphism to an image.

Proposition 2 (Properties of PlanReduce*):* PlanReduce always terminates. The reduced plan is equivalent to the original plan:

$$\mathsf{p\_to\_d}(\mathsf{PlanReduce}(p)) = \mathsf{p\_to\_d}(p), \qquad (7)$$

but it is possibly shorter: $|\mathsf{PlanReduce}(p)| \leq |p|$.

Proof: Writing the algorithm in tail-recursive form makes this analysis easy. To see that the algorithm terminates, note that at each step either the total length $|p_1| + |p_2|$ decreases, or, when it does not, the quantity that decreases is $\mathsf{noutoforder}(p_2 \circ p_1)$, defined as $\mathsf{noutoforder}(p) = \sum_{i=1}^{|p|} \sum_{j=1}^{i-1} (p^i \prec p^j)$, which is the number of pairs that do not appear in the string in the order according to $\prec$. To see that (7) holds, notice that $\mathsf{p\_to\_d}(p_2 \circ p_1) = \mathsf{p\_to\_d}(p)$ is conserved through each recursive call. ∎

From (7) it follows that

$$\mathsf{p\_to\_d}(\mathsf{Plans}) = \mathsf{p\_to\_d}(\mathsf{PlanReduce}(\mathsf{Plans})),$$

which means that we cover all plans, assuming that the predicates were correctly estimated from data. Note, however, that the stronger condition $\mathsf{p\_to\_d}(p_1) = \mathsf{p\_to\_d}(p_2) \Leftrightarrow \mathsf{PlanReduce}(p_1) = \mathsf{PlanReduce}(p_2)$ does not hold, because we only use first-order information between commands. For example, it might be that $\boldsymbol{u}_2 \circ \boldsymbol{u}_1$ commutes with $\boldsymbol{u}_3$, but not $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ separately.

D. Effects of reduction on planning performance

Fig. 3 compares, for each horizon L, the number of plans that would be generated without reduction:

$$\mathsf{Plans}_L = \{p \in \mathsf{Plans} \mid |p| \leq L\},$$

with the number of the reduced plans:

$$\mathsf{RedPlans}_L = \{\mathsf{PlanReduce}(p) \mid p \in \mathsf{Plans}_L\}.$$

In the case of a system where all actions commute, like our pan-tilt camera, the number of plans as a function of the length L changes from exponential to polynomial. If actions do not commute (as in the Dubin's car example shown in the following), then the number of plans is still exponential but with slightly smaller constant factor.

This translates to faster algorithms that need fewer node evaluations. We just need to modify the node expansion routine. Suppose that a node to be expanded is labeled by a plan p. Instead of returning the set of successors as $\{\boldsymbol{u} \circ p \mid \boldsymbol{u} \in \mathcal{U}\}$, the successors are computed as the set $\{\mathsf{PlanReduce}(\boldsymbol{u} \circ p) \mid \boldsymbol{u} \in \mathcal{U}\}$. This allows to not expand nodes that would be redundant.

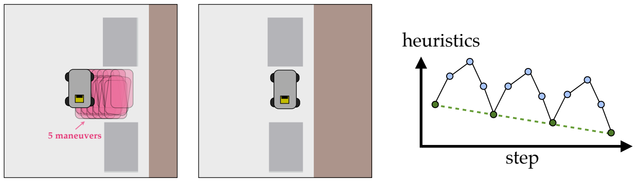
We modify the BNB, BNG, BNT algorithms obtaining the variants BEB, BEG and BET (see Table I for a summary). As expected, these perform much better in terms of memory and speed (Table II). Table III shows the performance of BET broken down for different lengths of the ground truth plan.

V. Combining actions for more efficient search

The performance of all methods considered up to now is dependent on how well the distance between images works as a heuristics for the planning problem. What we wish to happen is that the heuristics is mostly always decreasing along the plan from the start to the goal image. If this is not true, then the graph search algorithms try to fill basins containing local minima before continuing the search in a useful direction. (There are some possible mitigation measures that we did not discuss [13], such as mixing a breadth-first and a greedy strategy, as well as biasing expansion towards less explored areas.) The worst scenario possible is one in which the heuristics has large local variations along the feasible plan from start to goal image. The perfect example for this is the problem of parking with a car-like dynamics.

Consider a Dubin's car with 6 actions $\{F, L, R, f, l, r\}$, corresponding to driving forward (F), forward-right (R), forward-left (L), and their relative inverses, indicated with lower case. The observations are taken to be a "local map" of the environment, as it could be obtained by an omnidirectional range-finder (Fig. 4). The task is encoded by the start and goal images, in (a) and (b), respectively, of how the local map would look like in the initial and goal position. The actions induce Euclidean motions of the local map.

We generated several variation of this benchmark requiring an increasing number of maneuvers, from 1 to 5. None of the algorithms considered so far allow to solve the benchmark for more than 2 maneuvers (Table IV), given the resources limits we have established.



(a) $\boldsymbol{y}_{\text{start}}$ (b) $\boldsymbol{y}_{\text{goal}}$ (c) Qualitative behavior

Fig. 4. Parking for a Dubin's car formulated as planning in the image space. Here the "image" is a local map of the environment, as could possibly be obtained by a range-finder mounted on the car, in the starting position, in (a), and in the goal position, in (b). The car's cartoonish silhouette is only for visualizing the desired motion and is not part of the algorithms' input. (c) Motions of the car induce Euclidean motions of the image. Along the feasible plan that goes from start to goal image, the distance between the current image and the goal image has several local minima that make this a very challenging benchmark for greedy algorithms.

TABLE IV
PERFORMANCE IN THE PARKING BENCHMARKS (1–5 MANEUVERS)

	BET			BETc		
$p_\circ$	$p^\star$	*time (s)*	*# nodes*	$p^\star$	*time (s)*	*# nodes*
$RLrl$	$RLrl$†	5.52	52	$rlRL$	0.65	24
$(RLrl)^2$	$(rlRL)^2$	6.97	62	$(rlRL)^2$	1.26	24
$(RLrl)^3$	fail‡	316.19	227	$(rlRL)^3$	3.72§	48
$(RLrl)^4$	fail‡	317.76	157	$(rlRL)^4$	2.71§	48
$(RLrl)^5$	fail‡	368.50	167	$(rlRL)^5$	9.98§	72

†: $RLrl$ and $rlRL$ are two equivalent subplans. The algorithms use one or the other, according to some incidental order of expansions for the actions. ‡: BET fails for more than 2 maneuvers. §: BETc completes all benchmarks.

A. Taking advantage of diffeomorphism composition

The heart of the problem is that for some classes of systems, including nonholonomic systems, only considering "primitive" commands in $\mathcal{U}$ does not explore all possible directions of the state space (Fig. 5a). Using words from linear algebra in a metaphorical way, the "span" of the primitive actions $\mathsf{p\_to\_d}(\mathcal{U})$ has smaller "dimension" than the state space, which is the "span" of $\mathsf{p\_to\_d}(\mathsf{Plans})$.

Note that, for affine systems, where the (infinitesimal) actions are vector fields, then one can generate "new" actions, corresponding to infinitesimal "maneuvers", by computing the closure of the corresponding Lie algebra (see Isidori [16] for the general theory and Siciliano *et. al.* [17] for the application to mobile robots dynamics). Here, we cannot assume that diffeomorphisms are small/infinitesimal, and do not have a symbolic description of them. Nevertheless, our algorithmic approach shares the same spirit.

In our approach, to find a subset of Plans that explores all interesting directions, we construct a subset of plans P_{near} by enumerating the set $\mathsf{RedPlans}_{|\mathcal{U}|}$ and retaining the ones whose diffeomorphism lies close to the identity:

$$P_{\text{near}} = \{p \in \mathsf{RedPlans}_{|\mathcal{U}|} \mid d(\mathsf{p\_to\_d}(p), \mathsf{Id}_{\mathcal{S}}) \leq d_0\},$$

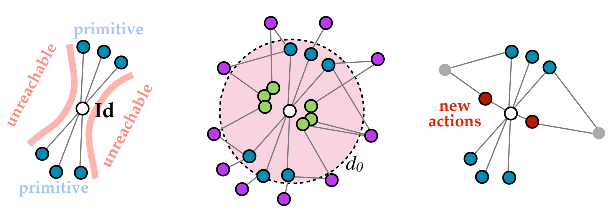
where "close" is defined by the scaling constant d_0 in (6), as illustrated in Fig. 5b. This set can be quite dense. We choose a subset of P_{near} by eliminating plans too close to each other; we have thus found a new set of commands, which allows more efficient search along all directions of the state space simultaneously (Fig. 5c). In the case of the Dubin's car, this procedure generates all "parking" maneuvers, which generates a net lateral motion, as well as some near-rotations in place.

We call "BETc" the variant of BET that computes and uses these composite actions. For this algorithm the parking benchmark is very easy, as it can move exactly sideways, thereby avoiding the local minima of the heuristics (Table IV).

B. Towards exploiting the global structure

It is interesting to see what happens if we carry over this exploration of the plan space beyond a neighborhood of the origin. In general, this cannot be done by a simple enumeration of all plans, but rather must be done by a proper graph search. Like in the planning scenario, the nodes of the graph are labeled with plans, but now there is no image associated; nodes are collapsed based on a threshold on the distance of the induced diffeomorphisms.

The resulting search graphs (Fig. 6), once embedded in Euclidean space, allow to recover the topology of the underlying state space, modulo the visibility constraints. For example, for an omnidirectional camera that can rotate on itself, we recover the topology of $\mathrm{SO}(2)$ (Fig. 6b). If the camera has a limited field of view, then the topology is that of a line (Fig. 6a). The Orbit camera dynamics can be embedded in 2D along two principal directions that correspond to pan and tilt (Fig. 6c), though a more metrically faithful representation in 3D wraps the nodes around the surface of a torus (Fig. 6d). The Dubin's car state space is $\mathrm{SE}(3)$, which needs at least $\mathbb{R}^4$ for a topologically correct embedding (as $\mathrm{SO}(2) \times \mathbb{R}^2$, with $\mathrm{SO}(2)$ embedded in $\mathbb{R}^2$) though locally can be embedded in $\mathbb{R}^3$ (Fig. 6e).



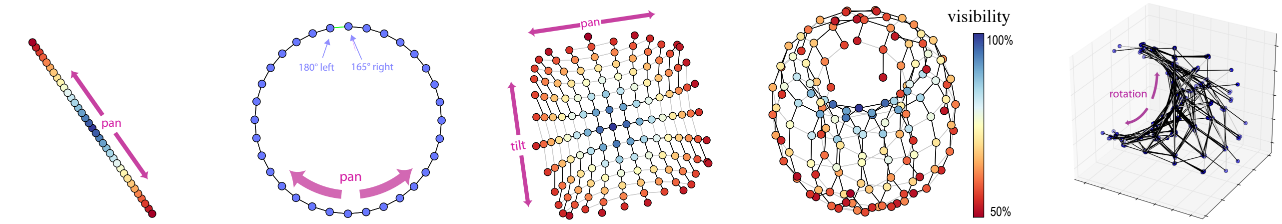
(a) Primitive actions (b) Constructing P_{near} (c) New actions

Fig. 5. (a) In nonholonomic systems, the primitive actions do not explore all degrees of freedom of the underlying state space. (b) We construct the set P_{near} of all plans that lead to a diffeomorphism near the identity. (c) By choosing a representative covering of P_{near}, we can select a set of plans that explore all degrees of freedom.

VI. CONCLUSIONS

Diffeomorphism models approximate the dominant dynamics of robotic sensorimotor cascades and can be learned from raw data. This paper has shown that these learned models have enough predictive power to be used for long-horizon open-loop image-based motion planning.

Several established approaches of the graph planning family have been adapted to this problem. The fact that each node in the graph, labeled by a plan, has also associated a diffeomorphism and a predicted uncertain images motivates the introduction of mechanisms that minimize the number of nodes that must be fully expanded and evaluated. The mechanism of plan reduction, described in Section IV, allows not to consider plans that can be said to be redundant just



(a) Panning camera (b) Omni. camera (c) Orbit camera (2D) (d) Orbit camera (3D) (e) Dubin's car

Fig. 6. *Embedding of* Plans *in Euclidean space according to the distance between induced diffeomorphisms, colored by visibility.* To each plan p, we associate an "uncertain diffeomorphism" $\mathsf{p\_to\_d}(p) \in \mathsf{UDiff}(\mathcal{S})$ and a "visibility" value, which is the portion of the image that we can actually predict, due to limited field of view. The empty plan, corresponding to the identity diffeomorphism, has 100% visibility. To obtain these images, we first compute a distance matrix between plans using the distance between their diffeomorphisms. We then use MDS [18] to find an embedding. In this way, we can recover the topology of the underlying state space from the learned incremental ations. Figure (a) is obtained using the (synthetic) dynamics of a camera with only one degree of freedom and limited field of view. As the camera pans left or right, the visibility decreases. Figure (b) corresponds to the case of an omnidirectional camera that rotates on itself. In this case, the visibility does not decrease. We recover the fact that rotating 180° on the left is the same as rotating 180° on the right. Figure (c) and (d) corresponds to the experimental data from the Orbit camera. Figure (e) shows the case of the Dubin's car, for 100 plans in the neighborhood of the identity. The underlying state space is SE(2), which cannot be embedded in $\mathbb{R}^2$ or $\mathbb{R}^3$ due to the different topology, though locally it organizes itself into the x, y, θ directions.

by looking at the pairwise relations between actions that we can infer from the learned diffeomorphism. The fact that actions are uncertain diffeomorphisms makes it possible to compress multiple actions into one. This allows to reason offline about the geometry of the system, and choose a "basis" of actions that explore efficiently in all directions, as explained in Section V. Using these improvements, the graph search algorithms in image space behave in a way which is qualitatively similar to the corresponding planning problem in the state space. For example, for systems where all actions commute, such as the camera of our experiments, planning with diffeomorphisms behaves very much like a search problem on a lattice with an imprecise heuristics.

In this paper we focused on the planning problem and we glossed over possible improvements of learning, such as using more complex representations of uncertainty, such as a covariance for every point-wise value of the diffeomorphisms. With a more thorough probabilistic treatment, we can give a clear statistical interpretation to some of the parameters that now appear as tuning constants, such as the value c in (2)–(5). Those thresholds on distances can be replaced with likelihood ratio tests, and c with a probability of making a mistake in inferring those predicates.

The fact that it is possible to recover the global topological/-metric structure of the underlying dynamical systems (Fig. 6) only from the incremental learned actions alone makes it possible to think of different approaches to solve the planning problem. Given the statistics of a set of image similarities to be used as heuristics (such as those shown in Fig. 2), it might be possible to derive optimal, zero-tuning branch-and-bound methods that explore systematically and optimally the space of plans. Or, given a set of arbitrary diffeomorphisms, it might be possible to derive image invariants/covariants, in the spirit of image moments.

Finally, we remark that hidden dynamics could make these methods fail spectacularly. Such is the case of sensors mounted on articulated bodies. However, in some cases, the hidden state only gives a perturbation of the diffeomorphism, such as in the case of the environment nearness when a camera undergoes translational motion. In those cases, it might be possible that, in closed loop, feedback allows to ignore some of those hidden states, thus making the problem simpler than open-loop planning.

References

[1] A Stoytchev. "Some Basic Principles of Developmental Robotics". In: *IEEE Trans. on Autonomous Mental Development* 1.2 (2009) DOI:10.1109/TAMD.2009.2029989.

[2] A. Censi and R. M. Murray. "Bootstrapping bilinear models of robotic sensorimotor cascades". In: *Int. Conf. on Robotics and Automation*. Shanghai, China, 2011 DOI:10.1109/ICRA.2011.5979844.

[3] A. Censi and R. M. Murray. "Bootstrapping sensorimotor cascades: a group-theoretic perspective". In: *Int. Conf. on Intelligent Robots and Systems*. San Francisco, CA, 2011 DOI:10.1109/IROS.2011.6095151.

[4] A. Censi and R. M. Murray. "Learning diffeomorphism models of robotic sensorimotor cascades". In: *Int. Conf. on Robotics and Automation*. Saint Paul, MN, 2012 (url).

[5] F. Chaumette and S. Hutchinson. "Visual servo control, Part I". In: *IEEE Robotics and Automation Magazine* 13.4 (2006).

[6] D. Scaramuzza and F. Fraundorfer. "Visual Odometry [Tutorial]". In: *IEEE Robot. Automat. Mag.* 18.4 (2011).

[7] V. Kallem, M. Dewan, J. P. Swensen, G. D. Hager, and N. J. Cowan. "Kernel-based visual servoing". In: *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2007.

[8] C. Collewet, E. Marchand, and F. Chaumette. "Visual servoing set free from image processing". English. In: *Int. Conf. on Robotics and Automation*. IEEE. Pasadena, United States, 2008 (link) (url).

[9] S. Han, A. Censi, A. D. Straw, and R. M. Murray. *A bio-plausible design for visual pose stabilization*. Tech. rep. CaltechCDSTR:2010.001. California Institute of Technology, 2010.

[10] G. Silveira and E. Malis. "Direct Visual Servoing: Vision-Based Estimation and Control Using Only Nonmetric Information". In: *IEEE Trans. on Robotics* 28.4 (2012).

[11] A. Sweet and X. Pennec. "Log-domain diffeomorphic registration of diffusion tensor images". In: *Proceedings of the 4th international conference on Biomedical image registration*. Berlin, Heidelberg: Springer-Verlag, 2010 (url).

[12] G. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Applied and Numerical Harmonic Analysis. Birkhäuser, 2011. ISBN: 9780817649432.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[14] E Olson. "Real-time correlative scan matching". In: *Int. Conf. on Robotics and Automation*. 2009 DOI:10.1109/ROBOT.2009.5152375.

[15] J. Mayfield, T. Finin, and M. Hall. "Using automatic memoization as a software engineering tool in real-world AI systems". In: *Proc. of the 11th Conf. on Artificial Intelligence for Applications*. IEEE Computer Society, 1995. ISBN: 0-8186-7070-3 (url).

[16] A. Isidori. *Nonlinear Control Systems*. London, UK, UK: Springer-Verlag, 2000. ISBN: 1852331887.

[17] B. Siciliano, L. Villani, L. Sciavicco, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2008.

[18] T. Cox and M. Cox. *Multidimensional Scaling*. Boca Raton, FL: Chapman & Hall / CRC, 2001. ISBN: 1-58488-094-5.