Identification of Effective Motion Primitives for Ground Vehicles

Tobias Löw[1,2], Tirthankar Bandyopadhyay[1], Paulo V. K. Borges[1]

Abstract— **Understanding the kinematics of a ground robot is essential for efficient navigation. Based on the kinematic model of a robot, its full motion capabilities can be represented by theoretical motion primitives. However, depending on the environment and/or human preferences, not all of those theoretical motion primitives are desirable and/or achievable. This work presents a method to identify effective motion primitives (eMP) from continuous trajectories for autonomous ground robots. The pipeline efficiently performs segmentation, representation and reconstruction of the motion primitives, using initial human-driving behaviour as a guide to create a motion primitive library. Hence, this strategy incorporates how the environment affects the robot operation regarding accelerations, speed, braking, and steering behaviours.**

The method is thoroughly tested on an autonomous car-like electric vehicle, and the results show excellent generalisation of the theoretical motion primitive distribution to real vehicle. The experiments are carried out on large site with very diverse characteristics, illustrating the applicability of the method.

I. Introduction

Motion planning for autonomous field robots is a daunting problem, not only because of the complexity of planning under the traditional kinematic, dynamic constraints, the uncertainty of observations and actuation, but also the effect of often unobservable environmental factors. Expert human operators have an uncanny ability to infer the operational parameters of varying environmental and task conditions for many field vehicles and successfully execute the task at hand. For instance, human drivers of all-terrain vehicles constantly respond to the terrain feedback, ground conditions like soft soil or hard surface, hilly or planar surfaces, by modulating their speed, turning radius and angular speeds in near real-time. The complexity of the interaction of all these parameters preclude the use of explicit modelling for motion planning and control for autonomous navigation. Traditional grid based [1] or sampling based [2], [3], approaches have achieved remarkable success in motion planning for ground robots primarily on planar surfaces like road networks. Such approaches often tend to treat environmental characteristics as a disturbance to either be handled at the controller level or incorporate some kind of motion limiting constraints at the kinematic or system dynamics level. They do not model the environment itself nor examine how the robot and the environment interact. As an example, they consider the full spectrum of the theoretical kinematics and only introduce some heuristic bias towards the goal, inspite of the fact that, depending on the environment, not all kinematics are feasible and/or necessary.

It is important to note that modelling the environment in enough detail for the necessary subset of the kinematics to be identified is a computationally expensive task and it would require extensive modelling that would need to be repeated for every slight change in the environment. Since an explicit model of the environment is therefore intractable, another approach should be investigated. This is particularly relevant in field robotics, since it deals with highly dynamic environments with significant uncertainty.

The search space for motion planning with the full kinematics of a robot is generally quite large. In most cases, searching all of it is often infeasible. For efficiency, an alternative is to bias the search space with some heuristic rule. Still, this representation only includes the inherent kinematics of the robot, but not the effects of the environment on those. This paper solves the problem of identifying this robot↔environment relationship and thereby the effective motion primitives of the robot in that environment. Such characterization finds application in a number of scenarios where the environment is initially mapped and autonomous vehicles operate consistently over that environment based on that specific map. Typical examples include manufacturing areas, industrial environments, oil and gas plants, agricultural fields, among others.

The problem has two main aspects: one is (i) the segmentation of the trajectory and the other is (ii) the compact representation of those segments as motion primitives. The requirement on the segmentation procedure is that it can process the data sequentially in an online fashion, hence needing real time capability. The representation needs to be memory and speed efficient when it comes to reproduction. In addition, the representation needs to allow for a comparison between the theoretical forward kinematics and the real data and ideally be able to use both information spaces simultaneously.

The proposed concept, coined effective Motion Primitives (eMP), approaches this problem by automatically segmenting the trajectories into unit motions that are represented by motion primitives. eMP consists of a complete pipeline that performs the segmentation, representation and reconstruction of the motion primitives. The method is extremely efficient computationally and provides a practical suite of motion strategies can be be effectively applied for ground vehicle navigation. Experiments with a car-like robot show the applicability of the approach in industrial areas, roads, and unstructured/off-road terrains. It is important to note that the method does the reconstruction of the motion primitives

[1]Robotics and Autonomous Systems Group, CSIRO, Australia. {tirtha.bandy@csiro.au, paulo.borges@csiro.au}

[2]Multi-Scale Robotics Lab, ETH Zurich, Zurich, Switzerland, {tobias.loew@alumni.ethz.ch}

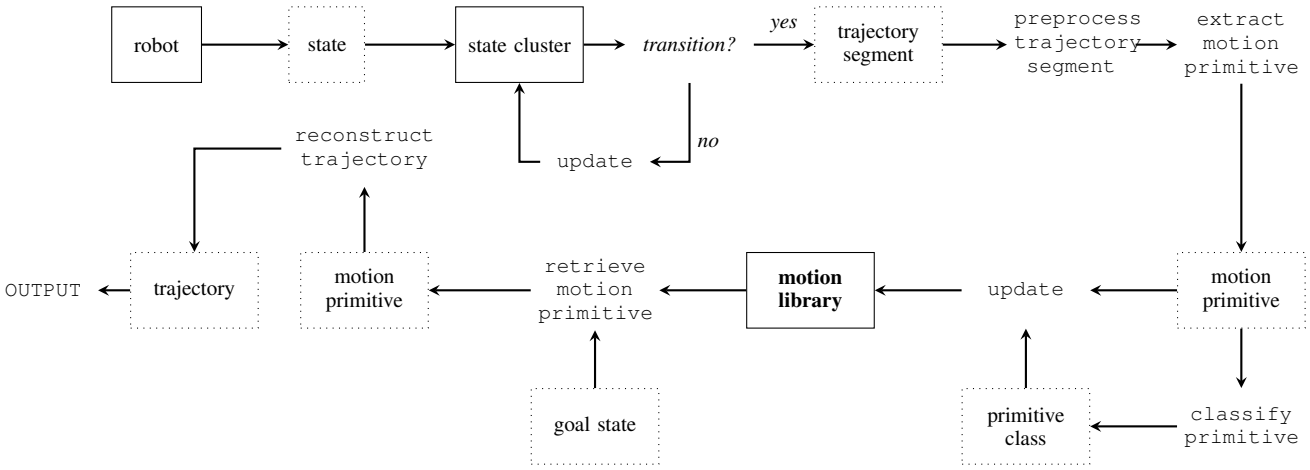


Fig. 1. Flow diagram for the algorithm. The process starts at the 'robot' block. Blocks with solid line borders do not change over time once their parameters have been defined. Blocks shown in dashed lines are constantly being updated. The key output is a trajectory with steering angle and a speed for the robot to move.

in a *global* sense (a map of the environment is assumed). Hence, a baseline algorithm for comparison is, for example, the A* [1], for which results are also presented.

This paper is organised as follows. Section II presents related work on motion primitives, trajectory segmentation and programming by demonstration. Section III describes eMP the algorithm in detail. Sections IV and V demonstrate the application of the algorithm in theoretical and real-world scenarios, respectively.

II. Related Work

Motion primitives are a well-known concept in robotics and a number of frameworks have been developed [4], [5], [6], [7]. Most of these approaches require manual segmentation of the robot trajectories [8]. Automatic segmentation and clustering are usually performed using stochastic methods [9], [10] and/or expectation maximization [11], [12] strategies. The scalable online sequence clustering (SOSC) algorithm [13] works by relying on small-variance asymptotics for the online updating of large scale Bayesian nonparametric mixture models. It is a useful tool for clustering an online trajectory when the number of clusters is initially unknown.

Finding lower dimensional representations of trajectories that can be reconstructed to the original movements has also been done through 'programming by demonstration' (PbD) [14], which aims at teaching skills to a robot without having to explicitly program them using a computer language [15], [16]. A key goal of programming by demonstration is to generalize previously acquired knowledge to operate in new situations, as for example, by generating a trajectory database using Gaussian process regression [17] with dynamic movement primitives [18] as the representation. Extracting invariant patterns from demonstrations is also required for the generalization of skills to new situations [19]. For instance, the robot needs to understand the demonstrations at a higher level while being invariant to the appearance and geometric aspects of objects (such as position, size, orientation and the viewpoint of the observer in the demonstrations), extract invariant segments (also termed as sub-goals or options), and smoothly follow the generated sequence of states (i.e. with a linear quadratic tracking controller). Alternatively, the demonstrations can directly be incorporated into the trajectory optimization [20].

Research has also investigated moving the learning from demonstrations into an online setting since real time modeling of complex nonlinear dynamic processes is important in various areas such as humanoid robotics [21], [22], Learning online requires incremental learning of motion primitives from observations. Stochastic segmentation is then used to partition the continuous observation sequence into motion segments. The identified primitives can then be incrementally clustered and organized into a hierarchical tree structure representing the known motion primitives.

In contrast to the works mentioned above, eMP features deterministic segmenting of a trajectory according to hyperparameters. In addition, eMP can be used online and is therefore capable of immediately adapting to changes in the environment. Our approach is also distinguished by the fusion of theoretical motion primitives with knowledge acquired from real world trajectories. This will be discussed in detail in the following section.

III. Algorithm

This section presents the algorithm that addresses the problem described in Section I. Figure 1 shows a block diagram of the pipeline, which can be summarised by the following points:

- Consider a robot traversing through an environment.
- Initially, as the robot moves, the state of the robot is grouped into clusters according to its position, orientation and speed.
- As the robot continues to move, a transition condition continuously checks whether the state still belongs to

that cluster.

- If not, on transitioning the trajectory is segmented and the resulting segment is processed in order for the motion primitive to be extracted.
- After extraction, the class of the primitive is determined and the motion library is updated.

Based on the pipeline above, when the robot is given a goal in space, a motion primitive can be retrieved from the motion library and the trajectory can be reconstructed for the goal to be reached.

A. State Definition

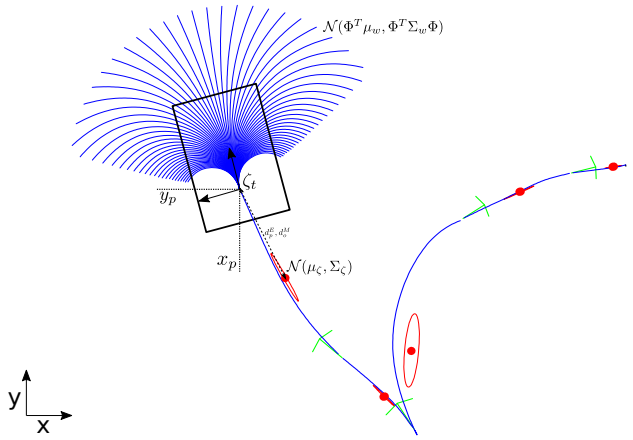


Fig. 2. Overview of the symbols used in this paper. The robot is marked by the black rectangle. Blue lines correspond to motion primitives. Red dots and ellipses mark the cluster mean and covariance respectively. The green arrows indicate the transition points of the segmentation algorithm.

The state of the robot is defined as by

$$\zeta_t = \left[\langle x_p, y_p, z_p\rangle, \langle w_o, x_o, y_o, z_o\rangle, \langle \dot{x}_p, \dot{y}_p, \varphi\rangle\right]^T \quad (1)$$

where $\langle x_p, y_p, z_p\rangle$ denote the current position in world coordinates, $\langle w_o, x_o, y_o, z_o\rangle$ the current orientation and $\langle \dot{x}_p, \dot{y}_p, \varphi\rangle$ the current velocity in the robot frame, with φ being the steering angle for car-like robots. The velocity state consists of controllable inputs for ground robots. For the case of non-holonomic robots, $\dot{y}_p = 0$.

B. Automatic Trajectory Segmentation

The trajectories are essentially a stream of states ζ_t at discrete time intervals Δt. This allows for the use standard techniques to calculate the mean and covariance of the incoming states in an online fashion.

1) State Clustering: The clustering is initialized with the initial mean being the first state ζ_0 and covariance matrix being the identity matrix scaled by an initial variance σ^2.

$$\mu_{\zeta,0} = \zeta_0 \qquad \Sigma_{\zeta,0} = \sigma^2 I \quad (2)$$

As more state data come in, this distribution is updated according to

$$\mu_{\zeta,t+1} = \frac{1}{w_{\zeta,t}+1}\left(w_{\zeta,t}\mu_{\zeta,t} + \zeta_{t+1}\right) \quad (3)$$

$$\begin{aligned}\Sigma_{\zeta,t+1} =& \frac{w_{\zeta,t}}{w_{\zeta,t}+1}\Sigma_{\zeta,t} \\ &+ \frac{w_{\zeta,t}}{(w_{\zeta,t}+1)^2}\left(\zeta_{t+1} - \mu_{\zeta,t+1}\right)\left(\zeta_{t+1} - \mu_{\zeta,t+1}\right)^T\end{aligned} \quad (4)$$

In this case $w_{\zeta,t}$ is the weight factor which is simply the sample count.

$$w_{\zeta,t+1} = w_{\zeta,t} + 1 \qquad w_{\zeta,0} = 1 \quad (5)$$

The state clustering is therefore effectively an implementation of Welford's algorithm [23].

2) Transition Condition: In order to achieve trajectory segmentation each new sample needs to be checked against a transition condition. In case the sample still belongs to the current cluster it is merged into that cluster, otherwise the trajectory is segmented and a new cluster initialized.

To detect a transition, we exploit the fact that a cluster can be well represented by a multivariate Gaussian distribution, as illustrated in Figure 2. This means that standard techniques like conditioning [24] are applicable. The mean and covariance can be divided into their position, orientation and velocity components:

$$\mu_\zeta = \begin{bmatrix}\mu_p \\ \mu_o \\ \mu_v\end{bmatrix} \qquad \Sigma_\zeta = \begin{bmatrix}\Sigma_{pp} & \Sigma_{po} & \Sigma_{pv} \\ \Sigma_{op} & \Sigma_{oo} & \Sigma_{ov} \\ \Sigma_{vp} & \Sigma_{vo} & \Sigma_{vv}\end{bmatrix} \quad (6)$$

Conditioning the position distribution on the current orientation and velocity is therefore done by

$$\overline{\mu}_p = \mu_p + \begin{bmatrix}\Sigma_{po} & \Sigma_{pv}\end{bmatrix} \cdot \begin{bmatrix}\Sigma_{oo} & \Sigma_{ov} \\ \Sigma_{vo} & \Sigma_{vv}\end{bmatrix}^{-1} \left(\begin{bmatrix}\zeta_o \\ \zeta_v\end{bmatrix} - \begin{bmatrix}\mu_o \\ \mu_v\end{bmatrix}\right) \quad (7)$$

$$\overline{\Sigma}_p = \Sigma_p - \begin{bmatrix}\Sigma_{po} & \Sigma_{pv}\end{bmatrix} \cdot \begin{bmatrix}\Sigma_{oo} & \Sigma_{ov} \\ \Sigma_{vo} & \Sigma_{vv}\end{bmatrix}^{-1} \begin{bmatrix}\Sigma_{op} \\ \Sigma_{vp}\end{bmatrix} \quad (8)$$

This yields the conditional distribution $\mathcal{N}\left(\overline{\mu}_p, \overline{\Sigma}_p\right)$ for the position. The conditional distributions for the orientation $\mathcal{N}\left(\overline{\mu}_o, \overline{\Sigma}_o\right)$ and the velocity $\mathcal{N}\left(\overline{\mu}_v, \overline{\Sigma}_v\right)$ can be found in the same way.

For the position the Euclidean distance is an intuitive measure, therefore the distance between the position sample and the conditioned mean is thresholded using $d^E_{p,max}$.

$$d^E_p = \sqrt{(x_p - \overline{\mu}_p)^T (x_p - \overline{\mu}_p)} < d^E_{p,max} \quad (9)$$

For quaternion distributions the Mahalanobis distance is a better choice as a distance measure because it is unitless and scale-invariant, which suits well for unit quaternions. The intuition behind the Mahalanobis distance is that it grows as the sample point moves away from the mean along each principal component axis. The threshold $d^M_{o,max}$ to be checked is therefore with regards to the squared Mahalanobis of the orientation sample to the conditioned distribution.

$$d^M_o = (x_o - \overline{\mu}_o)^T \overline{\Sigma}_o^{-1} (x_o - \overline{\mu}_o) < d^M_{o,max} \quad (10)$$

The choice of $d^E_{p,max}$ and $d^M_{o,max}$ is done empirically. Typical values for a car-like vehicle are $d^E_{p,max} = 8.0\,\text{m}$ and $d^M_{o,max} = \pi$. These limits provide a suitable set of motion primitives for the size of the test vehicle presented in Section V-A.

3) Algorithm for Automatic Trajectory Segmentation: The full procedure for automatic trajectory segmentation is shown in Algorithm 1.

Algorithm 1: Automatic Trajectory Segmentation

```
Data: robot state ζ
Result: transition, trajectory segment
if first robot state then
    initialize μ_ζ = ζ, Σ_ζ = σ²I
else
    check if state still belongs to the cluster
    if state belongs to cluster then
        updateCluster();
    else
        return transition + trajectory segment;
return no transition;
```

C. Motion Primitives

The resulting trajectory segment is re-sampled to T discrete time-steps before being converted to the motion primitive representation. In addition, instead of using the time t as index variable, the phase variable $z(t)$ is introduced.

$$z(t) = \frac{t - t_0}{t_{max}} \tag{11}$$

This ensures that the trajectory segments correspond to T discrete time-steps that are uniformly distributed between 0 and 1. The actual re-sampling process is done using linear interpolation for each state dimension separately.

The representation chosen for the motion primitives are probabilistic movement primitives (ProMPs) [25]. ProMPs offer useful properties that can be exploited later on when building the motion library and reconstructing trajectories, such as conditioning, product of Gaussians as well as a mean and covariance.

The probabilistic movement primitives rely on a basis function, for which a Gaussian radial basis function [26] was chosen in this paper, represented as

$$\phi_k(z(t)) = \frac{\exp\left(-\frac{1}{2}(z(t) - \mu_k)\sigma^{-1}(z(t) - \mu_k)\right)}{\sqrt{(2\pi)^D \sigma}}, \tag{12}$$

where μ_k are the means of the K components that are uniformly spread between 0 and 1. σ is the variance and can be chosen freely. The basis function is used to compute a matrix Φ that is later used for regression. Since all parameters can be set without prior knowledge, this matrix can be precomputed. For T time-steps and D dimensions, this yields a matrix $\Phi \in \mathbb{R}^{DT \times DK}$.

$$\Phi = \begin{bmatrix} I\phi_1(z_1) & \dots & I\phi_K(z_1) \\ \vdots & \ddots & \vdots \\ I\phi_1(z_T) & \dots & I\phi_K(z_T) \end{bmatrix} \tag{13}$$

Given a single trajectory segment the weights can be calculated by solving equation (14) which is a standard Ridge regression [27] with the regularization factor λ being chosen very small, mainly for avoiding computational underflow.

$$w_w = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T \cdot \begin{bmatrix} \zeta_0^T & \dots & \zeta_T^T \end{bmatrix}^T \tag{14}$$

The construction of the motion library is the problem of efficiently storing and merging the identified motion primitives while making the retrieval fast. Keeping all primitives in memory is not feasible from a storage point of view and make searching for a specific primitive computationally expensive.

For car-like robots two distinct and evident classes of motion primitives have been chosen: forwards and backwards driving. This choice reduces the problem to a simple binary classifier, with the y-z-plane as a boundary.

In order to integrate identified motion primitives into the motion library the fact that the motion primitives are represented by high-dimensional multivariate Gaussians can be exploited. For the merging this means another application of Welford's algorithm, i.e. equations (3) and (4), to the mean μ_w and covariance Σ_w of the weight vector from Equation (14) that is identified for each trajectory segment. This makes the motion library a collection of multivariate Gaussian distributions of weights over the basis function, one for each class of motion primitives, as shown in Equation (15).

$$p(w_w | class) = \mathcal{N}(\mu_w, \Sigma_w | class) \tag{15}$$

Reconstruction of a motion can be done via conditioning the distribution using the desired end state ζ_T^* of a trajectory [8]. The same can be done for any state ζ_t^* along the trajectory. The conditioned distribution $\mathcal{N}\left(\overline{\mu}_w, \overline{\Sigma}_w | \zeta_t^*\right)$ can be determined using (16) and (17):

$$\overline{\mu}_w = \mu_w + \Sigma_w \Phi (\Phi^T \Sigma_w \Phi)^{-1} (\zeta_t^* - \Phi^T \mu_w) \tag{16}$$

$$\overline{\Sigma}_w = \Sigma_w - \Sigma_w \Phi (\Phi^T \Sigma_w \Phi)^{-1} \Phi^T \Sigma_w \tag{17}$$

The full algorithm for building the motion library is outlined in Algorithm 2. It requires a trajectory segment as an input that can either be precomputed using the robot's kinematics model or identified from driving data.

Algorithm 2: Building the Motion Library

```
Data: trajectory segment
Result: updated motion library
re-sample trajectory segment
    transform t into z(t), interpolate to get T steps
extract motion primitive
    use equation (14)
determine primitive class
    x_T < 0 or x_T > 0
update motion library
    use equations (3) and (4)
```

Since the output of the reconstruction step is a trajectory, any trajectory following controller [28] can be used to execute it.

IV. Theoretical Motion Primitives

This section provides insights into application of the algorithm to the forward kinematics of a car-like robot, illustrating the theoretical distribution of motion primitives.

The movements of a robot can be described using a simplified kinematics model. This model considers the inputs (i.e. linear velocity) and calculates the current state of the robot. The considered robot is a car-like platform (i.e., non-holonomic), which can be described by the kinematics model shown in Equation (18) [29]. The inputs are the linear velocity v and the steering angle φ. L is the wheelbase of the car-like robot.

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} v\cos(\theta(t)) \\ v\sin(\theta(t)) \\ \frac{v}{L}\tan(\varphi) \end{bmatrix} \qquad u = \begin{bmatrix} v \\ \varphi \end{bmatrix} \tag{18}$$

By recursively calculating these equations and multiplying with Δt theoretical trajectories can be found, i.e. $x(t+1) = x(t) + \dot{x}(t)\Delta t$. Feeding those trajectories to Algorithm 1 and using the limits set in Section III-B.1 resulted in the theoretical trajectory segments shown in Figure 3 (in this figure, $\Delta t = 0.1$ s)

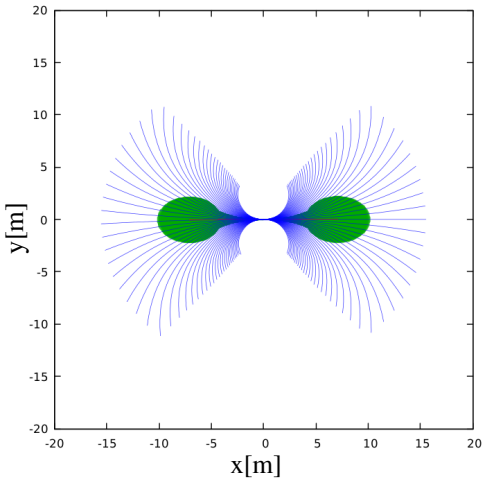


Fig. 3. Theoretical trajectory segments calculated using a car-like kinematics model, a linear velocity input of $v = 1.0$ and a steering angle of $\varphi \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ with 100 discrete steps was used. The mean and covariance of the resulting distribution are marked in red and green, respectively.

When merging the theoretical primitives for car-like robots into their respective classes to create the distributions as shown in Figure 3 the two distinct classes forward and backward driving can clearly be seen.

In this section a theoretical distribution of movement primitives has been computed. The experiments in the next section will show the applicability and generalization capabilities of this distribution to real world trajectories.

V. Experiments

This section presents experiments showing the trajectories generated and subsequent robotic operation using the proposed method. The results show the generalisation of the theoretical movement primitive distribution to real world applications.

A. Test Setup

For our experiments, the test area is a large industrial site (partially shown in the satellite image in Figure 5a). The area includes a road network with variable speed limits (some roads are designated to normal cars and some to industrial machinery), open and narrow spaces, flat and steep/hilly surfaces (both on-road and on off-road areas surrounded by large buildings or bush land). An artificial tunnel environment is also part of the site. Therefore, the site offers great variations, making for an ideal testing environment for the method proposed.

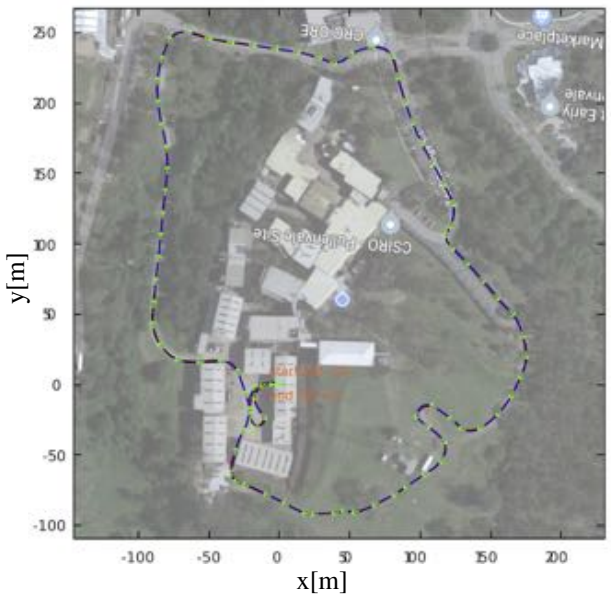
Fig. 4. Robotic platform used for testing is an automated John Deere Gator at the CSIRO's QCAT research facility.

The test platform is a car-like robotic vehicle (Figure 4), a John Deere TE Gator that has been automated at the CSIRO. It is 3.5 m long and 2 m wide. For the identification of the movement primitives, the vehicle is modeled as an Ackermann steering platform. Hence, the control inputs are linear velocity and steering angle. The maximum linear velocity $\dot{x}_{max} \approx 7\,\mathrm{m\,s^{-1}}$ with a total mass of about 800 kg. The steering angle φ is in the range $\varphi \in [-\pi/4, \pi/4]$.

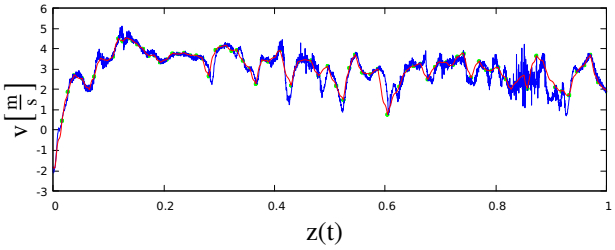
The eMP code has been implemented using C++14 and the Armadillo C++ library for linear algebra & scientific computing [30] was used for the matrix calculations. The robot used for experiments was running ROS Melodic [31] and Ubuntu 18.04 on a standard computer. Equipped with a 3D Velodyne VLP-16 lidar, the robot is localised against the environment using Posemap [32] and a continuous-time SLAM implementation [33], [34].

B. Test Results

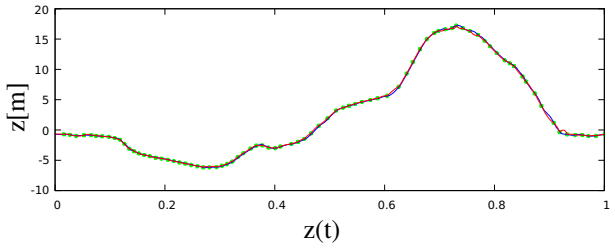
Figure 5a shows an overview of the whole test site including a human-driven trajectory. This trajectory is overlayed with the reconstructed trajectory using the identified transition points. The motion library used for reconstruction



(a) Human-driven and reconstructed trajectory overlayed on the test site.

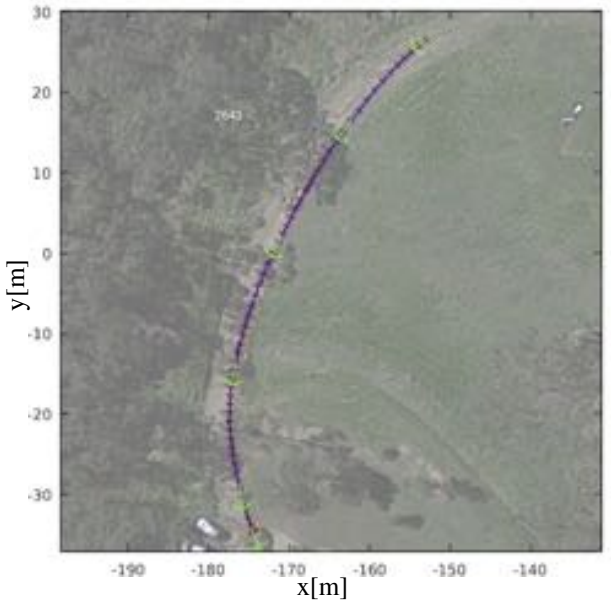


(b) Velocity profile of the driven trajectory over the phase variable $z(t)$.

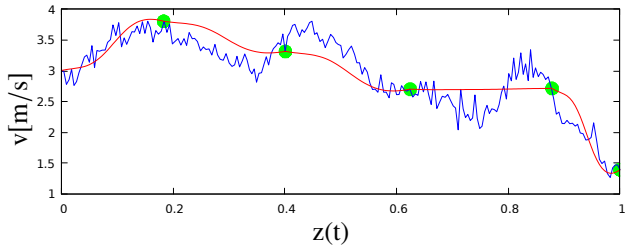


(c) Height profile of the driven trajectory over the phase variable $z(t)$.

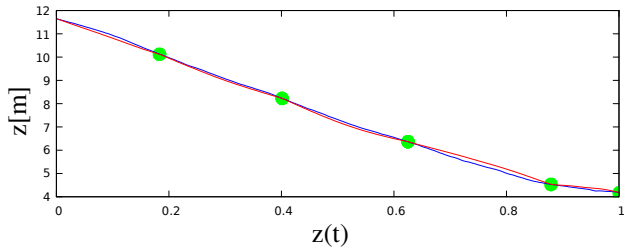
Fig. 5. Test site with trajectory and corresponding velocity and height profiles. The green arrows and circles mark the transition points. The red and blue ones correspond to the human-driven and reconstructed trajectories, respectively.



(a) Human-driven trajectory on the test site overlayed with the reconstructed trajectory.



(b) Velocity profile of the driven trajectory over the phase variable $z(t)$.



(c) Height profile of the driven trajectory over the phase variable $z(t)$.

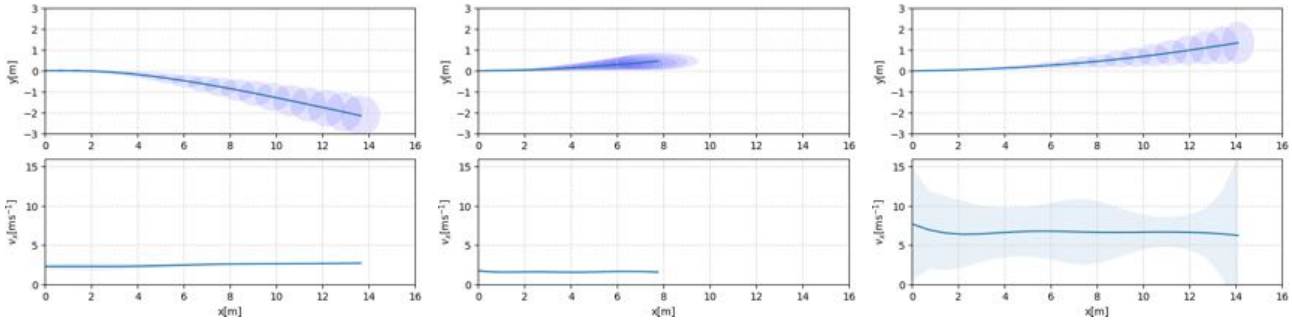
Fig. 6. Trajectory over a steep hill with the corresponding velocity and height profiles. The green arrows and circles mark the transition points. The red and blue ones correspond to the human-driven and reconstructed trajectories, respectively.

is derived from the theoretical motion primitives as shown in Section IV. This clearly shows that the theoretical model generalizes well to real world applications. The most significant benefit, though, lies in the velocity profile shown in Figure 5b. It shows that the reconstruction is also capable of following the human driving behaviour, thus implicitly incorporation human preferences in the environment without an explicit model. It should also be pointed out, that planning is done in three spacial dimensions as can be seen from the corresponding height profile shown in Figure 5c.

Figure 6a shows a human-driven trajectory down the slope of a hill. This is a particularly relevant example as that road is very gravelly and therefore leads to slippage, hence illustrating well the adaptability of the method.

In Figure 7 the extracted motion libraries for specific regions (road, hill, industrial) is shown as trajectories and their corresponding velocity profiles. It can be seen that the rough terrain of hill causes the motions to be shorter than for road and the industrial area, whereas they mostly differ in their velocity profile. This therefore captures the site regulations of the speed limit.

(a) Industrial area. (b) Offroad Hill. (c) Road.



(d) Industrial area. (e) Offroad Hill. (f) Road.

Fig. 7. Primitive weight distributions for specific situations and selected dimensions. For each dimension the different weights correspond to the K components of the basis function.

VI. Conclusion

This paper presented results of extraction of effective movement primitives that captures the movement limitations added by the environment implicitly by processing a set of human executed task trajectories that can be utilised to create a library of terrain specific movement primitives for autonomous navigation. It showed a method to automatically segment a vehicle trajectory and represent those segments in the form of movement primitives that are merged into a motion library. Feasible trajectories can be retrieved from the library and executed on the robotic vehicle, such that the vehicle moves efficiently and adequately in the environment. The paper has covered how to use a kinematics model to compare the prior knowledge of the robot's movements with data acquired from driving in an unknown environment. The system is also capable of acquiring driving data and in an online fashion extracting movement primitives from the data. Therefore, it allows for immediate adaption to unforeseen events such as damage to the robot that influences its motion behaviours without fully incapacitating it.

eMP has shown to be capable of incorporating human driving behaviours into the planning of high-dimensional global trajectories.

Future work will include planning trajectories with acquired effective movement primitives in a local sense that also incorporates information about the local environment such as obstacles, roughness and inclines.

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 7 1968.

[2] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[3] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 8 1996.

[4] S. Schaal, *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*. Tokyo: Springer Tokyo, 2006, pp. 261–280. [Online]. Available: https://doi.org/10.1007/4-431-31381-8_23

[5] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 11 2012, pp. 323–329.

[6] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 1 2016. [Online]. Available: https://doi.org/10.1007/s11370-015-0187-9

[7] A. K. Tanwani and S. Calinon, "Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 235–242, 1 2016.

[8] S. Gómez-González, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *CoRR*, vol. abs/1808.10648, 2018. [Online]. Available: http://arxiv.org/abs/1808.10648

[9] D. Kulic, W. Takano, and Y. Nakamura, "Online segmentation and clustering from continuous observation of whole body motions," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1158–1166, 10 2009.

[10] T. Nakamura, T. Nagai, D. Mochihashi, I. Kobayashi, H. Asoh, and M. Kaneko, "Segmenting continuous motions with hidden semi-markov models and gaussian processes," *Frontiers in Neurorobotics*, vol. 11, p. 67, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2017.00067

[11] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Learning movement primitive libraries through probabilistic segmentation," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 879–894, 2017. [Online]. Available: https://doi.org/10.1177/0278364917713116

[12] B. Wang, J. Gong, R. Zhang, and H. Chen, "Learning to segment and represent motion primitives from driving data for motion planning applications," *CoRR*, vol. abs/1809.06394, 2018. [Online]. Available: http://arxiv.org/abs/1809.06394

[13] A. K. Tanwani and S. Calinon, "Small-variance asymptotics for non-parametric online robot learning," *The International Journal of Robotics Research*, vol. 38, no. 1, pp. 3–22, 2019. [Online]. Available: https://doi.org/10.1177/0278364918816374

[14] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889008001772

[15] S. Calinon, "Learning from demonstration (programming by demonstration)," in *Encyclopedia of Robotics*, M. H. Ang, O. Khatib, and B. Siciliano, Eds. Springer, 2019.

[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*, 5 2009, pp. 763–768.

[17] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1327 – 1339, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889012000607

[18] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, 5 2002, pp. 1398–1403 vol.2.

[19] A. K. Tanwani, J. Lee, B. Thananjeyan, M. Laskey, S. Krishnan, R. Fox, K. Goldberg, and S. Calinon, "Generalizing robot imitation learning with invariant hidden semi-markov models," *CoRR*, vol. abs/1811.07489, 2018. [Online]. Available: http://arxiv.org/abs/1811.07489

[20] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters, "Demonstration based trajectory optimization for generalizable robot motions," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 11 2016, pp. 515–522.

[21] D. Kulic, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012. [Online]. Available: https://doi.org/10.1177/0278364911426178

[22] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 4 2007.

[23] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962. [Online]. Available: https://amstat.tandfonline.com/doi/abs/10.1080/00401706.1962.10490022

[24] M. Eaton, P. Euclid, C. U. Library, and D. U. Press, *Multivariate Statistics: A Vector Space Approach*, ser. Lecture notes-monograph series. Cornell University Library, 2008. [Online]. Available: https://books.google.com.au/books?id=8U5TnQAACAAJ

[25] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2616–2624. [Online]. Available: http://papers.nips.cc/paper/5177-probabilistic-movement-primitives.pdf

[26] M. D. Buhmann, "Radial basis functions: Theory and implementations," *Radial Basis Functions*, vol. 12, 01 2003.

[27] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970. [Online]. Available: http://www.jstor.org/stable/1267351

[28] B. Paden, M. ÄŇĂąp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, 04 2016.

[29] J.-P. P. Laumond, *Robot Motion Planning and Control*. Berlin, Heidelberg: Springer-Verlag, 1998.

[30] C. Sanderson and R. R. Curtin, "A user-friendly hybrid sparse matrix class in C++," *CoRR*, vol. abs/1805.03380, 2018. [Online]. Available: http://arxiv.org/abs/1805.03380

[31] O. S. R. Foundation. (1999) Robot operating system (ros). [Online]. Available: http://www.ros.org/

[32] P. Egger, P. V. K. Borges, G. Catt, A. Pfrunder, R. Siegwart, and R. Dube, "Posemap: Lifelong, multi-environment 3d lidar localization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 3430–3437.

[33] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[34] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 4312–4319.