Human-inspired robotic levering using Periodic Dynamic Movement Primitives

Boris Kuster, Matevž Majcen Horvat

Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana
E-mail: boris.kuster@ijs.si, matevz.majcen@ijs.si

Abstract

Autonomous robotic disassembly (recycling) of electronics requires a variety of skills. One of these is levering, which allows the robot to apply greater forces using a fulcrum, providing a mechanical advantage. While such tasks can be accomplished in some cases using prerecorded robot trajectories obtained by demonstration, a more general approach to levering is preferred. The issue is an automatic adaptation to the different object shapes. This work presents an algorithm for performing human-inspired robotic levering exploiting periodic dynamic movement primitives and force/torque feedback-based control for contact point determination. The algorithm autonomously adapts to the different object shapes and provides for the successful accomplishment of the levering task.

1 Introduction

With the recent shift towards environmentalist policies, robotic disassembly of items for waste reduction by recycling has received increased attention. Humanity produces a great deal of electronic waste [1]. The application of intelligent and flexible robots to replace human workers can decrease recycling costs in the long term. On the other hand, recycling faces the problem of small batches and a great variety of recycled items. In such circumstances, the effort for robot programming to perform autonomous disassembly of generic electronics is one of the reasons for the slow deployment of robotics-based solutions. During the disassembly of electronics and other items, various robotic skills are needed, one of which is levering. It is a process whereby mechanical advantage can be gained using a rigid beam (lever) and a fixed hinge (fulcrum), allowing a greater force to be exerted on the load (the levered object). Some operations where levering is needed include removing pins and nails or separating different device parts. While this is an easy task for humans, as they can rely on vision, force, and pressure sensing supported by previous experience and generalisation capability, robotic disassembly applications still commonly use pre-recorded trajectories. These trajectories, however, need to be carefully adapted and reprogrammed for each disassembled object. Dynamic movement primitives were shown to be applicable to assembly and disassembly tasks [2] as well as to the task of motor control in humanoid robots [3]. In this paper, we propose to improve the generalisation of the levering movement using Periodic Dynamic Movement Primitives (PDMPs) and combine them with external forces and torques estimation, acting on the end-effector. One of the benefits of the proposed framework is adaptation without additional vision information, which can be unreliable due to the poor lighting conditions and occlusions often encountered in recycling processes. The proposed algorithm for performing human-inspired robotic levering was implemented on a collaborative Franka Emika Panda robot and applied to disassembling heat cost allocators (electronic devices).

2 Preliminaries and related work

Dynamic movement primitives (DMPs) were proposed as an efficient way for learning and control of complex robot behaviors [4]. The basic form of DMPs consists of a differential equation (a linear second order damped spring attractor system) along with an added nonlinear forcing term, which allows the adaptation of the simple second order attractor dynamics to a specific robot movement [5]. Eq. (1) is the nonlinear differential equation defining a DMP. This model can be written in first-order notation, as shown in Eqns. (2, 3). Here, τ is a time constant, while α_z and β_z are positive constants that represent the damping and proportional terms, respectively. With the appropriate choice of α_z and β_z, the system will be critically damped and $\mathbf{y}$ will monotonically converge towards $\mathbf{g}$. In the context of robotics, $\ddot{\mathbf{y}}, \dot{\mathbf{y}}$ and $\mathbf{y}$ represent the acceleration, velocity and position of a joint, respectively. $\mathbf{g}$ represents the goal, the desired final position of the movement. If the forcing term $\mathbf{f}$ is set to zero, these equations represent a globally stable second-order linear system with $(\mathbf{z}, \mathbf{y}) = (0, \mathbf{g})$ as a point attractor.

$$\tau\ddot{\mathbf{y}} = \alpha_z(\beta_z(\mathbf{g} - \mathbf{y}) - \dot{\mathbf{y}}) + \mathbf{f} \tag{1}$$

$$\tau\dot{\mathbf{y}} = \mathbf{z} \tag{2}$$

$$\tau\dot{\mathbf{z}} = \alpha_z(\beta_z(\mathbf{g} - \mathbf{y}) - \mathbf{z}) + \mathbf{f} \tag{3}$$

To avoid explicit time dependency, the phase $\mathbf{x}$ has been introduced. It follows the first order linear dynamics shown in Eq. (4). The advantage of this approach is that, for example, we can stop the evolution of time to account for perturbations during trajectory execution.

$$\tau\dot{\mathbf{x}} = -\alpha_x \mathbf{x} \tag{4}$$

Periodic dynamic movement primitives can be modeled in a similar fashion as discrete DMPs, shown in Eqns. (5, 6). Ude *et al.* [6] presented an implementation of PDMPs. In this work, we follow their implementation.

$$\dot{\mathbf{z}} = \Omega(\alpha_z(\beta_z(\mathbf{g}-\mathbf{y})-\mathbf{z})+\mathbf{f}(\phi,r)) \tag{5}$$

$$\dot{\mathbf{y}} = \Omega\mathbf{z} \tag{6}$$

The phase variable ϕ is introduced to avoid explicit time dependency. The phase is assumed to move with constant speed, as shown in Eqns. (7, 8). The frequency of oscillation is defined as Ω.

$$\tau = \frac{1}{\Omega} \tag{7}$$

$$\dot{\phi} = \Omega \tag{8}$$

The forcing term $\mathbf{f}$ can be used to create more versatile point attractor dynamics, so that a more specific path can be followed from the current position to the goal, for example avoiding an obstacle between the current position and the goal. The forcing term usually takes the form shown in Eq. (9) for PDMPs, where ψ_i are fixed basis functions, while w_i are adjustable weights [7].

$$\mathbf{f}(\phi,r) = \frac{\sum_{i=1}^{N} \boldsymbol{\psi}_i(\phi)\cdot\mathbf{w}_i}{\sum_{i=1}^{N}\boldsymbol{\psi}_i(\phi)} r \tag{9}$$

In general, a demonstrated robot trajectory $\mathbf{y}_{demo}$ is recorded in joint or Cartesian space. Velocities and accelerations can be calculated as derivatives of the recorded positions or recorded directly. Then, the trajectory is encoded into a DMP by setting the goal to be the last recorded position. For PDMPs, the goal is set as shown in Eq. (10). Eq. (1) can be reformulated as Eq. (11).

$$\mathbf{g} = 0.5\cdot(min(\mathbf{y})+max(\mathbf{y})) \tag{10}$$

$$\mathbf{f}_{target} = \tau^2\ddot{\mathbf{y}}_{demo} - \alpha_z(\beta_z(\mathbf{g}-\mathbf{y}_{demo})-\tau\dot{\mathbf{y}}_{demo}) \tag{11}$$

For PDMPs, this equation takes the form of Eq. (12), where r is the amplitude of the oscillator.

$$\mathbf{f}(\phi(t),r) = \frac{\ddot{\mathbf{y}}}{\Omega^2} - \alpha_z(\beta_z(\mathbf{g}-\mathbf{y}) - \frac{\dot{\mathbf{y}}}{\Omega}) \tag{12}$$

The forcing function values are approximated by a linear combination of weight functions, shown in Eq. (9). Weight functions for Periodic DMPs are shown in Eq. (13). The weights $\mathbf{w}_i$ are calculated by applying standard regression techniques [7]. Centers of the weight functions, $\mathbf{c}_i$, are evenly distributed along the trajectory, while $\mathbf{h}_i > 0$.

$$\boldsymbol{\psi}_i(\phi) = exp(\mathbf{h}_i(cos(\phi-\mathbf{c}_i)-1)) \tag{13}$$

Various improvements have been proposed for DMPs. Ude *et al.* propose defining DMPs for non minimal, singularity free representations of orientation, with rotation matrices and quaternions [7]. Nemec *et al.* [2] propose extensions, which enable the generalization of movement back and forth for Cartesian Dynamic Movement primitives, while also encoding the robot end-effector quaternion orientation. Saveriano *et al.* [4] describe the various existing DMP formulations and discuss their advantages and disadvantages. Ijspeert *et al.* [5] show how to optimize DMP parameters to minimize various costs, for example the total jerk of the trajectory or the end-point variance. Deniša *et al.* [8] propose compliant movement primitives (CMPs), which encode both the kinematic trajectory as well as the corresponding joint torques. This allows for compliant robot behavior, which is advantageous when operating in unstructured environments alongside humans.

3 Levering setup

Fig. 1 shows the typical levering setup. The lever is attached to the robot flange, therefore the terms lever and tool can be used interchangeably. The used tool in this task is a screwdriver. The levered part lies within the object. Subsequently, we will refer to the levered part as part. To increase mechanical advantage, the lever is positioned against the fulcrum.

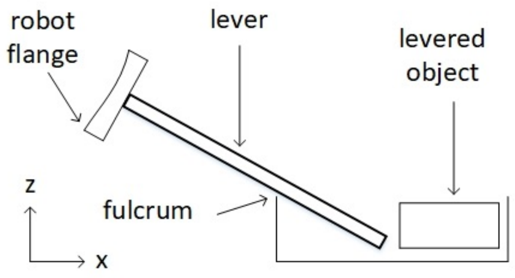

Figure 1: Elements of a levering process.

In Fig. 2, object coordinate system(c.s.) (X, Y, Z axes), the robot flange c.s. (X_0, Y_0, Z_0), the nominal tool c.s. (X_1, Y_1, Z_1) and the tool c.s. (X_2, Y_2, Z_2) are shown. The Tool Center Point (TCP) is located at the tool c.s. Initially, the nominal tool c.s. and the tool c.s. are identical, the tool c.s. is changed only after the location of the fulcrum is known. In our work, we make use of force-torque (FT) measurements (F_{x0}, F_{y0}, F_{z0}, M_{x0}, M_{y0}, M_{z0}), which are calculated in the robot flange c.s. The lever (tool) angle α is shown in Fig. 2.

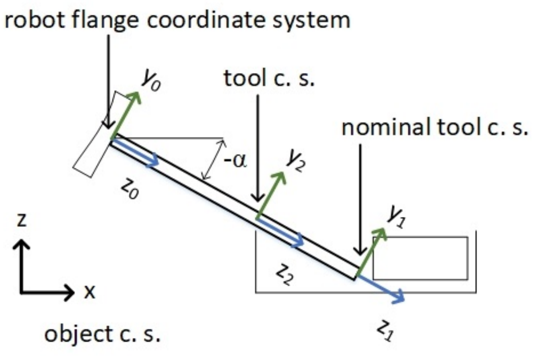

Figure 2: The object, robot flange, nominal tool and tool coordinate systems.

4 Fulcrum and part contact point search algorithm

Our search algorithm is parameterized with the following input parameters:

- initial Cartesian position (x, y, z) in robot coordinate system.
- the c.s. rotation angle β.
- torque threshold to detect the part and fulcrum, M_{thr}.
- force F_{z0} threshold to detect the part, F_{thr}

The robot TCP must be initially positioned above the part and the approximate location of the part and fulcrum must be known in advance. This approximate information can be obtained, for example, by a vision system, or by pre-positioning the part within a holder with a known pose. The input parameters for the algorithm are the initial Cartesian position (x, y, z) between the fulcrum and part (in the robot c.s.) and the angle of rotation around the z axis, β. This position and the angle define the plane in which we will perform the levering, as shown in Fig. 1). The initial Cartesian position must be located between the fulcrum and the part.

To detect the part, we can take advantage of the fact that our object has a flat lower plane, which we detect by moving in the negative z direction until determining contact by measuring F_{z0}. The location of the levered part (load) can be trivially determined by moving along the positive x axis until detecting contact by measuring M_x. After the threshold value M_{thr} is reached, the current position is recorded as the part position.

To contact the fulcrum, the tool angle α is decreased until contact is detected by measuring M_{x0}. The TCP is recorded. For approximately detecting the location of the fulcrum, we can take advantage of the fact that we know both the robot flange and TCP positions. After slightly increasing the threshold value M_{thr}, further rotation is performed and the position is again recorded. In a 2D plane, the approximate position of the fulcrum can be determined by the intersection of the two lines connecting the robot flange to the TCP.

The robot's TCP, which was previously located at the end of the lever, is moved to the position of the fulcrum, as seen in Fig. 2 (the TCP is reduced in the negative z_0 axis). This ensures that, when changing the tool angle α, the lever is rotated around the fulcrum and always stays in contact with it.

5 Executing the levering

We have observed that humans often use periodic movements when levering, especially when they do not know the force required to dislodge an object with the lever. In doing so, they slightly increase the force on the lever in each period.

Our levering algorithm is parameterized with the following input parameters:

- initial amplitude of the sinusoidal cycle, set to $A = 10°$ in our experiments.
- duration of the sinusoidal cycle, set to $t_c = 10\ s$.
- number of weights for encoding the PDMP, set to $n = 25$.
- goal scaling for each iteration, set to $g = g + 0.2 \cdot A$.
- initial radius of the PDMP, set to $r = 1$.

To mimic this behavior, we generate a single degree-of-freedom (DOF) sinusoidal cycle with an amplitude A and cycle time t_c, which represents the angle of the end-effector relative to the initial angle α at which the robot is in contact with the fulcrum and part. This angle is the most important parameter in levering.

The specified sinusoidal pattern is encoded into the PDMP as per section 2. We use n weights.

After the lever is in the initial position, touching both the part and the fulcrum, the execution of the PDMP begins.

While the PDMP is being executed, the measured force-torque signals are used to detect when the levering was successful, as per Section 6. If the entire trajectory is completed without encountering a success signal, the PDMP goal value is increased as specified by the input parameters of the algorithm. This effectively increases the force applied to the levered part during each iteration. The original generated trajectory (lever angle) is shown in Fig. 3, along with the decoded PDMP trajectory. Here, three iterations are shown, and after each iteration the goal angle is increased.

6 Detecting levering completion using force-torque measurement

To detect when levering is successful without using vision or other feedback signals, we can use force-torque (FT) measurements $(F_{x0}, F_{y0}, F_{z0}, M_{x0}, M_{y0}, M_{z0})$ on the robot end effector.

The algorithm for detecting levering success is parameterized with the following input parameters:

- component of the FT measurement, in our case $i = 4$, which corresponds to M_x.

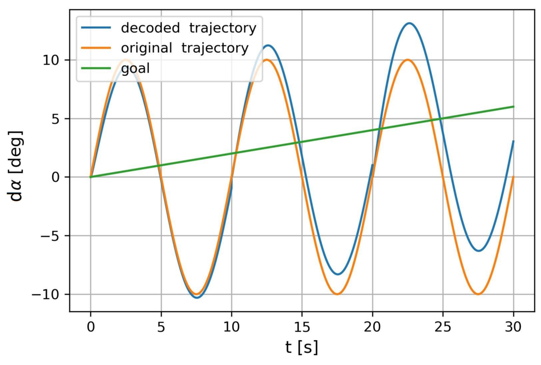

Figure 3: Three iterations of decoded PDMP with an increasing goal.

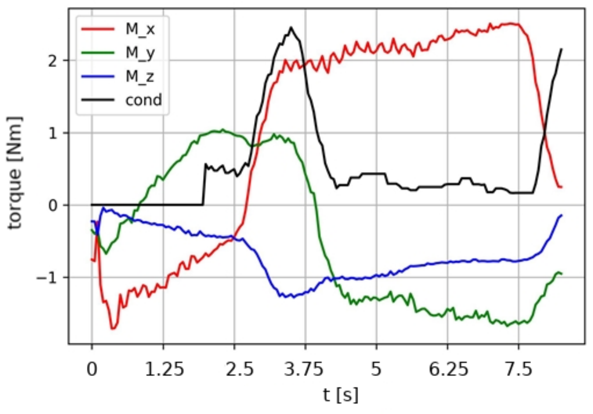

Figure 4: Measured torques and the condition variable.

- torque value used as a threshold, $M_{max} = 2\ Nm$.
- time window in which the threshold condition is calculated, $t_w = 1\ s$.
- the secondary cut-off condition, α_{max}, set to 0°.

We observe that M_x shows a sharp drop-off at the point in time when the part (load) is dislodged, as shown in Fig. 4. There are two possible conditions for levering success. Firstly, the levering is successful if the difference between the maximal and minimal torque value within the width of the signal observation window, t_w, is greater than the parameter M_{max}, as shown in Eq. (14). The current sample time is denoted as t_k. The measured torques, as well as the value of this threshold condition $cond$ are shown in Fig. 4. An additional constraint is that this condition can only be triggered while the lever angle α is increasing, meaning it's applying force to the part.

$$max(M_x(t)) - min(M_x(t)) > M_{max}, t = t_k - t_w, ..., t_k, \tag{14}$$

When recycling old electronics, it can sometimes happen that the levered part is very loose and does not provide a large resistance force, so a drop in M_x will not be detected. Therefore, the secondary condition for levering success is if the lever angle α is higher than the parameter of the second cut-off condition α_{max}. In our specific case, shown in Fig. 1, the contact point with the part is always lower than the fulcrum in the object z axis. Therefore, if the lever's angle is greater than horizontal ($\alpha > 0°$), the levering stops.

7 Conclusion and further work

We presented a human-inspired levering algorithm, composed of two sub-tasks: searching and levering. The search algorithm automatically detects contact points after being roughly pre-positioned between and above the part and fulcrum locations. The levering algorithm is accomplished using periodic dynamic motion primitives framework. Both algorithms are parametrized to quickly adapt to different use-cases. The success of the levering action is detected based on a pre-defined criteria of a torque measurement drop-off. However, this value needs fine tuning. To overcome this problem, we propose to analyze the force torque signals using the dynamic time warping (DTW), which compares a reference signal of successful levering torques with the observed torques in the current levering iteration.

References

[1] E. Commission, D.-G. for Environment, M. Tesar, B. Karigl, C. Lampert, C. Neubauer, J. Oliva, and J. Wolf, *Study on quality standards for the treatment of waste electrical and electronic equipment (WEEE) : final report*. Publications Office, 2021.

[2] B. Nemec, M. Simonič, T. Petrič, and A. Ude, "Incremental policy refinement by recursive regression and kinesthetic guidance," in *International Conference on Advanced Robotics (ICAR)*, (Belo Horizonte, Brazil, Brazil), 2019.

[3] S. Schaal, *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*, pp. 261–280. Tokyo: Springer Tokyo, 2006.

[4] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *CoRR*, vol. abs/2102.03861, 2021.

[5] A. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural computation*, vol. 25, 11 2012.

[6] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[7] A. Ude, B. Nemec, T. Petrić, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2997–3004, 2014.

[8] M. Deniša, T. Petric, A. Gams, and A. Ude, *A Review of Compliant Movement Primitives*. 10 2016.