Motivation Primitives for a Tumbling Robot

Brett Hemes, Duc Fehr, and Nikolaos Papanikolopoulos
Center for Distributed Robotics
University of Minnesota
Minneapolis, MN 55455
{hemes, fehr, npapas}@cs.umn.edu

Abstract—The desire for a high mobility-to-size ratio in mobile robots has led to the exploration of many new methods of locomotion, one of which is tumbling. To the authors’ knowledge, there are very few tumbling robots in existence and no formalized methods for their control. In this paper we begin addressing these issues by presenting an approach for deriving motion primitives for tumbling robots. We apply our method to the specific case of a two-armed tumbling robot and include the final derived motion primitives. Additionally we discuss in general the motion of tumbling robots and introduce 3 useful gaits derived from the resulting primitives of our approach.

I. INTRODUCTION

One of the most desirable characteristics of miniature robotic systems is a high mobility-to-size ratio. For mobile robots, movement is a crucial requirement necessary for all but the simplest of desired tasks. It is often the case that real-world environments include complex terrains composed of varying surfaces, rapid changes in elevation, and a multitude of hazardous obstacles. Depending on a particular task’s environment, a certain level of mobility is required by a robot to succeed. Miniature robotic systems, however, do not always have the luxury of large wheels or tracks to cope with such complex environments. A form of locomotion with a higher mobility-to-size ratio would enable smaller robots to exhibit the required level of mobility for more complex terrain, greatly expanding the possible tasks accomplishable by miniature robotic systems.

Generally, a land-based robotic platform’s mobility, or ability to traverse terrain, is proportional to its overall size. This limitation is obvious in conventional wheeled and tracked platforms where bigger robots almost always outperform smaller robots. There has been significant work in developing new methods of robotic propulsion that take a step away from conventional designs, providing increased mobility on smaller scales [1]–[7].

In particular, we are most interested in the locomotion methods of the TETwalker [6] and Turbot [4], [8] platforms. Both of these platforms employ a tumbling motion to produce directed progress through their environment. With this method the robot flips and tumbles across the environment, effectively turning its entire body into a tool for overcoming obstacles. The TETwalker, a tetrahedral robot, achieves motion by controlling its center of gravity by changing the lengths of its struts. By moving the center of gravity outside of the robot’s stability margin, a tumble is achieved. Turbots take a more aggressive approach and use a set of arms (rigid or flexible) to propel themselves over complex terrain. This results in a capable, yet complex and somewhat non-intuitive motion.

In a discussion on the benefits of legged designs, Raibert [9] makes the claim that wheels are limited by the worst terrain they encounter while legged systems are generally limited only by the best footholds available. A tumbling design, we believe, falls somewhere in-between the two, improving greatly on the capabilities of wheeled designs while simultaneously avoiding most of the hardware complexity inherent in a legged system. We believe that this particular method of locomotion holds much potential and could be very successful from the mobility-to-size standpoint.

To the authors’ knowledge, there are very few tumbling robots in existence and no formalized methods for their control. We believe that this is due in part to two main factors. Firstly tumbling is not readily seen in nature and is therefore non-intuitive to human beings. There is an ever-present theme of biologically inspired designs in robotics and without a nature analog, tumbling robots have been ignored. Secondly the dynamics involved in the tumbling motion can be quite complex and are further complicated by non-uniform terrain. In this paper we begin addressing these issues with an approach for deriving motion primitives for tumbling robots. To show the usefulness of the presented method, we present the results of applying it to a two-armed tumbling robot.

II. TUMBLING ROBOTS

Tumbling is a largely unexplored method of locomotion that operates through repeatedly driving the system into states of instability, inducing dynamic motion that brings the system from the current state to another of lower energy, where energy is added through actuation. The result is a series of tumbles that produce a net displacement of the robot through its environment. Formally we define a tumble and tumbling robot as follows:

Definition A tumble is a dynamic state of instability during which the robot pivots about an axis formed by two or more contact points with the ground, accelerating downward with gravity and thus behaving as an inverted pendulum.

Definition A tumbling robot is any robot that, by the previous definition, tumbles as its primary means of locomotion with the body playing an active role in achieving such motion.
While one could argue that some current forms of robotic locomotion fit this description (dynamic bipeds for example), there is a main difference that separate tumbling robots from other dynamic mobile robots. This difference is implied by the second definition in that the body must play an active role in achieving tumbles. Conventional mobile robots traditionally have passive bodies or use them as a means of balancing whereas tumbling robots’ bodies often come in contact with the ground. While this can be detrimental to conventional robotic systems, it is highly desirable and often necessary in a tumbling system. The ground-body interaction provides a rich set of possible motions responsible for tumbling robots’ high mobility-to-size ratio.

III. Generation of Motion Primitives

Motion primitives are a valuable tool and prove useful in solving complex problems such as motion planning [10]. Due to the non-intuitive motion of a tumbling robot, deriving useful motion primitives can be quite difficult. In this section we present an approach that produces useful motion primitives along with convenient notation for representing the resulting primitives. We begin by discretizing both the inputs and possible configurations of the tumbling robot. By executing the complete set of discretized inputs starting from each of the discrete configurations, it is possible to construct a directed state graph that represents the relationship between various configurations of the robot. In the last step we associate global displacements with each of the state graph edges, the result of which is a compact representation of the motion primitives for the discretized inputs. In the following subsections we discuss in detail the aforementioned process and apply them to a physical two-armed tumbling robot.

A. Hardware

Before we continue with deriving motion primitives, we find it best to discuss the robot in which we apply our method. The chosen robot, the Adelopod, is a small two-armed tumbling robot we developed specifically for use in our research (see Figure 1). This particular robot has a trapezoidal body and two rigid arms. The system has four actuated degrees of freedom (DOF), where each arm forms a 2-DOF kinematic chain; the link-frame assignments for a single arm are shown in Figure 2. Without loss of generality to our method, we fix the rotation about \( \hat{Z}_1 \) and focus on the motion of the robot with only one degree of freedom per arm.

B. Discretization

The first step in deriving motion primitives is to discretize the possible configurations of the particular robot. For the example of our two-armed robot we have two inputs and have chosen to discretize by starting with arm angles equal to multiples of \( \pi/2 \), producing 4 possible configurations of each arm. The motion of this particular robot is a function of the arm angles and the body’s orientation, therefore we also discretize the body’s configuration as angles equal to multiples of \( \pi/2 \). This discretization produces a maximum of \( 4^3 \) possible states for our robot. However, as we will see in the following sections, the symmetry of the robot along with the elimination of unreachable states greatly reduces the number of unique states.

To simplify the transitions between configurations, it is useful to have the arms touching the ground whenever possible. Under the assumption that the robot is on a hard, level surface, the arms require a constant \( \Delta \theta \) for each configuration in order to assure contact with the ground. By this definition we can now represent the arm angles by the following equation:
θ_n = n (\pi/2) + \Delta \theta_n, \quad n \in \{0, 1, 2, 3\}. \quad (1)

The values of \( n \) and \( \Delta \theta \) for the left arm-body configurations of our robot are shown in Table I. The right arm-body values of \( n \) and \( \Delta \theta \) can be calculated with the following relations:

\[
n_{\text{right}} = (n_{\text{left}} + 2) \mod 4 \quad (2)
\]

and

\[
\Delta \theta_{\text{right}} = -\Delta \theta_{\text{left}}. \quad (3)
\]

Let us briefly pause here to introduce the notation in the first column of Table I; this notation is used throughout the rest of the paper to compactly represent our discretized configurations. Each of the \( 4^3 \) configurations is expressed as a 3-tuple with components representing the orientation of each arm along with the body. The order of the 3-tuple is as follows: left arm, body, right arm. Each element is defined as the orientation of the particular component looking in the negative direction onto the robot along the \( \hat{Z}_G \) axis (see Figure 2) where the orientations are described verbally as up, down, dot, or cross (additionally, don’t-cares are represented with a ‘−’). Figure 3 shows four pictorial representations of possible configurations along with a picture of the actual robot in the respective configuration (for example, Figure 3(b) shows the robot as described by the \( (\text{down, up, cross}) \) 3-tuple and is pictorially represented by Figure 3(f)).

![Fig. 3. Sample states with corresponding chart icons.](image)

We now have a unique representation for each of the \( 4^3 \) configurations along with the actual arm angles (from Equation 1) for each configuration, thus completing the discretization.

### C. State Graph

A tumbling robot, at any given time, is restricted in its motions by its current configuration. In other words, the next state, \( x_{k+1} \), has the relation

\[
x_{k+1} = f(x_k, u_k),
\]

where \( x_k \) is the current state and \( u_k \) is the discretized control input. Therefore the motion primitives must be arranged in such a way that their dependence is accounted for; we achieve this with a directed graph where each node corresponds to a configuration and each edge to a control input.

A graph is built as follows. We start by placing our robot in an arbitrary configuration as described in the previous section and execute a discretized control input (for our graph, a control input corresponds to moving one of the motors in a single direction, providing 4 possible inputs). We stop the robot when it has reached a new discretized configuration and then update the \( \Delta \theta \) of the arm not influenced by the executed control input so that it matches perfectly the new configuration (these updates are small and have little influence on the position of the robot). At this point we add a directed edge corresponding to the executed input to the state graph. By iterating over all possible inputs for every discrete configuration, it is possible to construct a directed graph that relates the various discrete configurations according to the control inputs.

Figure 4 shows the state graph derived for our tumbling robot by applying the above method (the graph was derived for the robot on a smooth, level surface with the robot operating at near-static speed). We have separated the graph into 4 sub-graphs for clarity, where each graph focuses on a different body configuration. As presented, the graph exhibits abundant symmetries. For example, Figure 4(c) can be produced by rotating Figure 4(a) by 180° and subsequently reflecting each state about the vertical axis. Other symmetry in the graph is apparent about axes from the lower left to the upper right of each sub-graph. Additionally some sub-

<table>
<thead>
<tr>
<th>Arm-body Configuration</th>
<th>n</th>
<th>( \Delta \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>+7°</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0°</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-7°</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0°</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>+7°</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0°</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-7°</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0°</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>+17°</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0°</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-17°</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0°</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>+15°</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0°</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>-15°</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0°</td>
</tr>
</tbody>
</table>
graphs exhibit symmetry about axes from the lower right to the upper left; this can be seen in Figure 4(d) and also, though less apparent, partially in Figure 4(b). These symmetries are a desirable characteristic and greatly reduce the memory requirements necessary to represent the graph on an actual robot.

Other useful properties of the graph include the identification of transitions that do not affect the body and transitions that induce a tumble. In each sub-graph, there exists a 3-by-3 region of configurations centered about \((\text{dot}, -, \text{dot})\) in which transitions have no effect on the body. We refer to these as free moves and are useful for changing the relative orientation of the arms in respect to the body. Any unidirectional edge of the graph corresponds to a transition that induces a tumble and is thus irreversible.

We also wish to point out that there are some configurations that do not exist on the state graph. In general, these are any states that are of the form \((\text{cross}, -, \text{cross})\), \((-\text{cross}, \text{cross})\), or \((\text{cross}, \text{cross}, -)\). These are states of unstable equilibria and are not reachable from any state through the execution of the discretized control inputs. They serve no useful purpose and, therefore, have been excluded from the graph.

D. Displacement

Up until this point we have concerned ourselves only with the configuration of the robot. For the final step in our method, we associate global displacement with each edge in the state graph to produce the actual motion primitives. For the purpose of deriving the primitives, we are only interested in the robot’s position and orientation on the \(\hat{X}\hat{Y}\hat{G}\)-plane\(^1\).

\(^1\)All of the unit vectors in this section use the notation of Figure 2.
Therefore we define the state of the robot, \( \mathbf{x} \), as a vector containing its 2-D position and orientation. Each transition is then characterized by the difference, \( \Delta \mathbf{x} \) between the states.

\[
\Delta \mathbf{x} = (\Delta x, \Delta y, \Delta \phi)^T. \tag{5}
\]

All of these values are expressed in the robot’s frame. From Equation 5 we get the update equation,

\[
\mathbf{x}_{k+1} = \left( \begin{array}{c} x_k + C_{\hat{Z}_G} (\phi_k) \Delta x_k \\ y_k + C_{\hat{Z}_G} (\phi_k) \Delta y_k \\ \phi_k + \Delta \phi \end{array} \right), \tag{6}
\]

for the state of the robot, where \( C_{\hat{Z}_G}(\phi_k) \) is the rotation about \( \hat{Z}_G \) of \( \phi_k \).

Because the robot’s body is unrestricted in its rotation about \( \hat{X}_0 \) (necessary for tumbling), the calculation of \( \Delta \mathbf{x} \) is not as straightforward as one might assume. The issues revolve around the representation of the robot’s orientation. If represented as a single vector, there exist orientations in \( \mathbb{R}^3 \) that result in an undefined orientation in \( \mathbb{R}^2 \). Additionally, as the robot tumbles, the orientation vector loses meaning.

To overcome this issue we instead represent the orientation of the robot as the \( \hat{Y} \hat{Z}_0 \)-plane of the robot’s frame. With this representation, we define orientation in \( \mathbb{R}^2 \) as the direction of the line created by the intersection of the \( \hat{Y} \hat{Z}_0 \)-plane with the \( \hat{X} \hat{Y}_G \)-plane. This representation gives us two possible values of \( \phi \), however, if we arbitrarily assign \( \phi \) one of the two directions and limit \( \Delta \phi \) to the interval \([-\pi/2, \pi/2]\), \( \phi \) remains well defined over all discretized state transitions.

The elements of \( \Delta \mathbf{x} \) are calculated with the following expressions:

\[
\Delta y_k = \frac{\Delta P_k \cdot Y_{(k)}^{(k)} \mid_{\text{proj}}}{\| Y_{(k)}^{(k)} \mid_{\text{proj}}} \tag{7},
\]

\[
\Delta x_k = \frac{\Delta P_k \cdot \left( C_{\hat{Z}_G}(-90^\circ) Y_{(k)}^{(k)} \right) \mid_{\text{proj}}}{\| Y_{(k)}^{(k)} \mid_{\text{proj}}} \tag{8},
\]

and

\[
\Delta \phi_k = acos \left( \frac{Y_{(k)}^{(k)} \cdot Y_{(k+1)}^{(k+1)} \mid_{\text{proj}}}{\| Y_{(k)}^{(k)} \mid_{\text{proj}} \| Y_{(k+1)}^{(k+1)} \mid_{\text{proj}} \|} \right) \tag{9},
\]

where \( \Delta P_k = G^{k}_{k+1} - G^{k}_k \) is the difference of the robot’s position between time steps \( k \) and \( k+1 \) in the global frame.

In Table II, we list the elements of \( \Delta \mathbf{x} \) of our robot for five of the state transitions of Figure 4. We derived each of these values experimentally using a Vicon motion capture system over 25 runs of each primitive. Additionally Figure 5 depicts the positional results graphically. With this characterization of displacement, our state chart becomes a map of the robot’s motion primitives.

\[2 \text{By this method, } \phi \text{ is undefined only when the } \hat{Y} \hat{Z}_0 \text{-plane is parallel to the } \hat{X} \hat{Y}_G \text{-plane, which does not exist in our state chart.} \]

### IV. Gaits

In addition to providing a compact representation of the previously derived motion primitives, the state chart can also be used to develop periodic gaits for the robot. By exploiting the properties of the state chart, we are able to derive useful sequences of motion that exhibit desirable characteristics. For instance forward motion can be generated by traversing the state chart while choosing primitives with large forward displacement, minimal lateral motion, and small changes in orientation. Conversely, turning motion can be generated by choosing primitives with large successive changes in orientation. As examples, we have included three periodic gaits developed in this manner and show them in Figure 6.

Figure 6(a) shows a gait developed to produce forward motion. In this situation, we have chosen primitives for their large positive \( \Delta y \). Additionally, we have chosen the sequence

<table>
<thead>
<tr>
<th>Transition</th>
<th>Results [mm, mm, deg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \Delta x ) 24.9 ( \var(x) ) 2.9</td>
</tr>
<tr>
<td>2</td>
<td>( \Delta y ) 12.6 ( \var(y) ) 1.0</td>
</tr>
<tr>
<td>3</td>
<td>( \Delta \phi ) 5.0 ( \var(\phi) ) 0.1</td>
</tr>
<tr>
<td>4</td>
<td>( \Delta y ) 49.8 ( \var(y) ) 0.7</td>
</tr>
<tr>
<td>5</td>
<td>( \Delta \phi ) 12.4 ( \var(\phi) ) 0.8</td>
</tr>
</tbody>
</table>

Fig. 5. Graphical depiction of \( G^{k}_{k+1} \) with \( G^{k}_k = (0, 0)^T \) and \( \phi_k = \pi/2 \) for the 5 transitions given in Table II.
of primitives to stay close to the aforementioned lines of symmetry. By doing so, it is possible to alternate between identical sequences on either side of the symmetry, which effectively eliminates lateral and orientational offsets. This results in a gait of large forward displacement with small periodic oscillations along the forward axis (the transitions of Table II were chosen to demonstrate these qualities).

The turning gaits were developed by selecting states from the chart and traversing in a single direction until small loops were found that produced large orientational displacements with minimal positional displacement. The turning gait of Figure 6(b) is easily identifiable by traversing loop 3 of Figure 4(a) while the gait of Figure 6(c) can be derived by following loop 2 of Figure 4(a).

These gaits comprise only a small sub-set of the possible sequences provided by the state graph, however, the ability to both turn and move forward enable the execution of directed motion.

V. CONCLUSION AND FUTURE WORK

Tumbling robots, due to the active participation of the body in their motion, exhibit large mobility-to-size ratios. Additionally, their high mobility is achieved with a minimal amount of hardware complexity. Despite these advantages, the tumbling nature of their movement poses some interesting challenges. In this paper we address these challenges and start laying the foundation necessary to fully harness the capabilities of robotic tumbling locomotion. Specifically we give definitions of both tumbling motion and tumbling robots and present a method of deriving motion primitives for tumbling robots along with the example of a two-armed tumbling research platform. We conclude by introducing a few useful periodic gaits capable of producing simple turning and forward motions.

Although we performed our method for a particular robot, we believe that with some modification, the ideas presented in this paper can be extended to other tumbling robots such as the TETwalker [6]. We believe that our work provides a foundation for future research and serves as a natural transition to the development of motion planning for tumbling robots.

In our derivation of the motion primitives, we made several assumptions regarding the terrain and operating speed. In future work we hope to relax these assumptions, thus extending our work to account for variable terrain and motion involving the dynamics of the robot. Additionally we would like to explore the effects of continuous inputs, thus departing from our discretized approach.

ACKNOWLEDGEMENTS

This work has been supported in part by National Science Foundation through grants #IIS-0219863, #CNS-0224363, #CNS-0324864, #CNS-0420836, #IIP-0443945, #IIP-0726109, and #CNS-0708344.

We would also like to thank Professor Bernard Mettler from the University of Minnesota Interactive Flight Research Lab for access to the lab’s motion capture system. Additional thanks to Nuri Kundak for his assistance in obtaining data.

REFERENCES