

# PHYSICAL SIMULATION OF INARTICULATE ROBOTS

Guillaume Claret, Michaël Mathieu, David Naccache and Guillaume Seguin

École normale supérieure  
Département d'informatique  
45 rue d'Ulm, F-75230, Paris CEDEX 05, France  
{surname.name}@ens.fr except mmathieu@clipper.ens.fr

**Abstract.** In this note we study the structure and the behavior of inarticulate robots. We introduce a robot that moves by successive revolvings. The robot's structure is analyzed, simulated and discussed in detail.

## 1 Introduction

In this note we study the structure and the behavior of inarticulate robots. The rationale for the present study is the fact that, in most robots, articulations are one of the most fragile system parts. Articulations require lubricants and call for regular maintenance which might be impossible in radioactive, subaquatic or space environments. In addition, articulations are sensitive to dust (or humidity) and must hence be shielded from external nano-particles e.g. during martian sand-storms.

In this work we circumvent articulations by studying a robot that moves by shifting its center of gravity so as to flip repeatedly.

## 2 The Robot

The proposed robot's model is a regular polyhedron prolonged with hollow legs. Each hollow leg contains a worm drive allowing to move an internal mass  $m$  inside the leg<sup>1</sup> as shown in Figure 1. By properly moving the masses the device manages to revolve and hence move in the field.

Different regular polyhedra can be used as robot bodies. In this study we chose the simplest, namely a tetrahedron. Hence, the robot has two basic geometrical parameters,  $\ell$  the tetrahedron's edge and  $L$  the leg's length. Figures 2, 3 and 4 show the robot's structure.

The robot has three stable states, head-down (HD), head-up (HU) and side-down (SD). In the head-down and head-up states, the robot rests on three legs while in the side-down mode the robot rests on four legs. Possible transition modes are hence:

$$\text{head-down} \leftrightarrow \text{side-down} \leftrightarrow \text{head-up}$$

Note that a direct head-down  $\leftrightarrow$  head-up transition is impossible.

The robot's state and position are thoroughly characterized by three parameters:  $G = \{G_X, G_Y\}$  the  $\{X, Y\}$  coordinates of the robot's centroid,  $P \in \{\text{HD}, \text{HU}, \text{SD}\}$  the robot's current stable state and the angle  $\alpha$  formed between the  $X$  axis and the robot's reference direction. The reference direction, shown in Figure 5, is defined in two different ways depending on the robot's current state.<sup>β</sup>

<sup>1</sup> The internal mass must not necessarily be a dead weight. e.g it can be the battery used to power the worm drive.

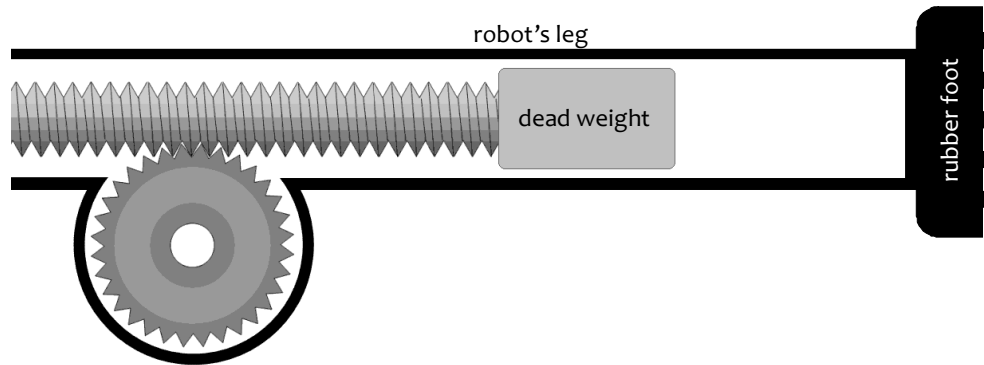


Fig. 1. Schematic Cross-Section of the Robot's Leg

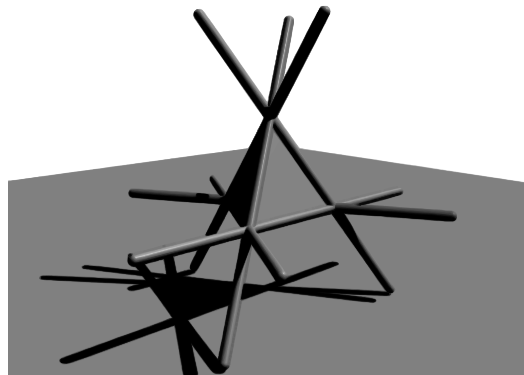


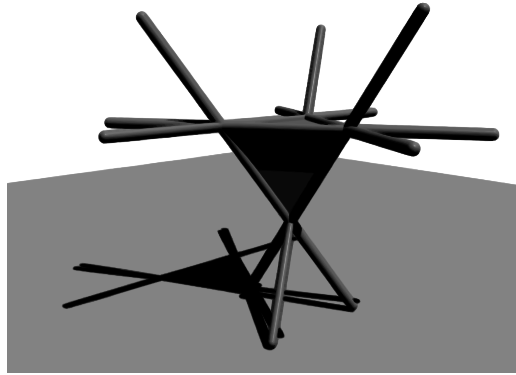
Fig. 2. Basic Robot Structure, Head-Up (HU) State.

### 3 Reachable Points

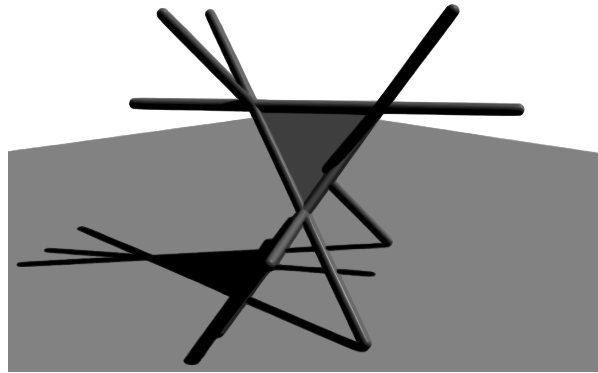
We define a *reachable point* as any space coordinate on which the robot can set the center of the rubber foot. It appears (although we did not prove this formally) that when the robot is constrained to a bi-state (i.e.  $HD \leftrightarrow SD$  or  $SD \leftrightarrow HU$ ) locomotion mode and to a delimited planar surface only a finite number of points can be reached (Figures 6 and 7) whereas if we allow tristate  $HD \leftrightarrow SD \leftrightarrow HU$  transitions, an infinity of points seems to become reachable (Figures 8 and 10). It might be the case that increasing the set of reachable points calls for walking further and further away from the robot's departure point and heading back to the vicinity of the departure point through a different path. Proving that an infinity of reachable points can be achieved in a delimited planar surface is an open question.

### 4 Pathfinding

To approximately reach a destination point, we first experimented a simple BFS (Breadth First Search) algorithm [4]. Before queuing potential revolving options, our implementation checked that the targeted position does not fall within an obstacle. This allowed locomotion with obstacle avoidance. The approach turned-out to be inefficient. Indeed, the  $HD \leftrightarrow SD \leftrightarrow HU$  locomotion results in the re-exploration of the already visited areas even though the algorithm



**Fig. 3.** Basic Robot Structure, Head-Down (HD) State.



**Fig. 4.** Basic Robot Structure, Side-Down (SD) State.

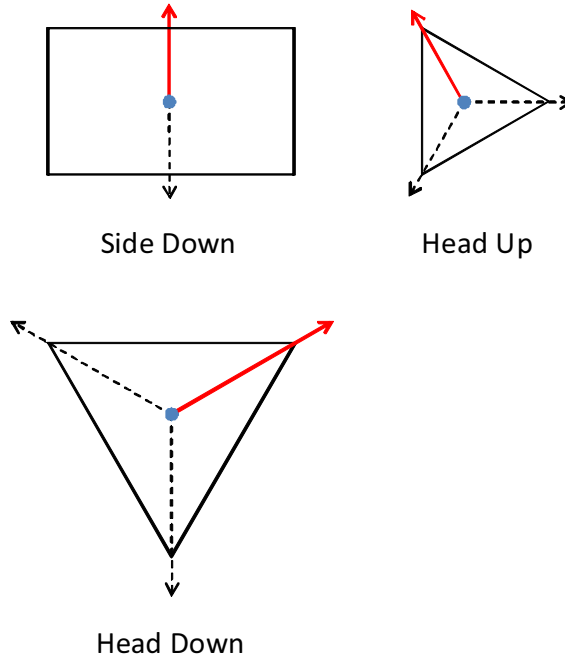
records all already visited configurations. This typically happens when the edge of a rectangle and the edge of a triangle nearly overlap (*cf.* Figure 10).

To improve performance we implemented an  $A^*$  algorithm [3]. This was done by modifying the BFS simple queue into a prioritized queue. Priorities were determined using  $\Delta_{\text{dep}}$ , the length of the path since the departure point and an estimate of the distance to destination  $\Delta_{\text{des}}$ . At any step, the next chosen path is the shortest, *i.e.* the one whose  $\Delta_{\text{des}} + \Delta_{\text{dep}}$  is the smallest.

The application of the  $A^*$  algorithm to obstacle avoidance is depicted in Figure 9. The yellow circle represents the arrival's target and the black rectangle is an obstacle. The obstacle avoidance C++ code can be downloaded from [2].

## 5 Simulation

A physics engine is computer software that provides an approximate simulation of certain simple physical systems, such as rigid body dynamics (including collision detection). Their main uses are in mechanical design and video games. Bullet [1] is an open source physics engine featuring 3D collision detection, soft body dynamics, and rigid body dynamics. The robot's structure was coded in about 60 Bullet code lines. Weights move up and down the legs using sliders (a slider is a Bullet object materializing the link between rigid bodies) as



**Fig. 5.** Reference Directions

shown in Figure 11. To illustrate the robot’s operation in real time, we added a target sphere to which the user can apply a force vector using the keyboard’s  $\leftarrow\rightarrow\uparrow\downarrow$  keys. As the target sphere starts to move, the robot starts revolving to follow it. We could hence visually conduct realistic physical experiments on various surfaces with the robot. *cf.* Figures 12 and 13. A movie showing such an experiment is available on [2].

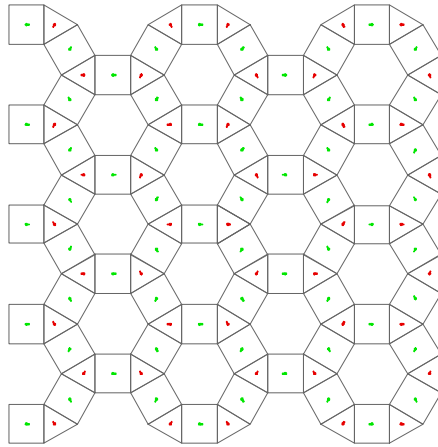
## 6 Further Research

This work raises a number of interesting questions that seem to deserve attention:

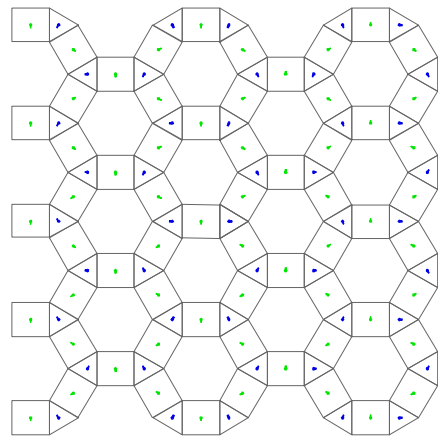
**Landing State Probability:** Assume that the robot is given a random 3D spin and is thrown on a planar surface. What are the probabilities  $\Pr_{\ell,L}[\text{HU}]$ ,  $\Pr_{\ell,L}[\text{HD}]$  and  $\Pr_{\ell,L}[\text{SD}] = 1 - \Pr_{\ell,L}[\text{HU}] - \Pr_{\ell,L}[\text{HD}]$  that the robot falls into each of the states?

**Energy:** It is equally interesting to compute the energy spent during locomotion and finding out if for a given locomotion task there exists an optimal worm drive lifting strategy. Indeed, it might be the case that weights must not necessarily be lifted until the end of each hollow leg but to a lesser energy-optimal height.

**Inertia:** Taking inertia into account is interesting as well: inertia allows to capitalize spent energy by keeping rolling instead of halting at each locomotion step. This is very apparent in the Bullet simulation but quite difficult to model precisely.



**Fig. 6.** Bistate Locomotion SD  $\leftrightarrow$  HU



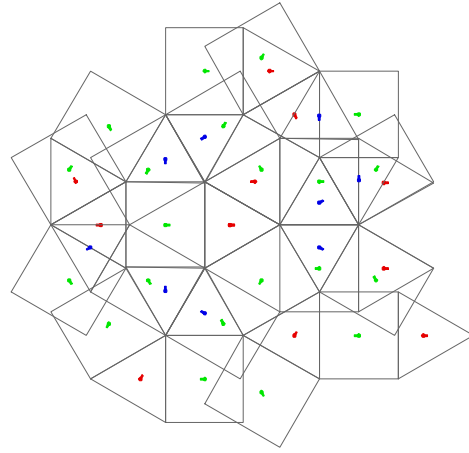
**Fig. 7.** Bistate Locomotion HD  $\leftrightarrow$  SD

**Slopes:** Finally, it is interesting to determine the robot's maximal climbable slope  $\alpha_c(\ell, L, m)$  as well as the robot's maximal controlled descending slope  $\alpha_a(\ell, L, m)$ . A controlled descending is a descent of a slope in which the robot can halt at any point *i.e.* not roll down a hill.

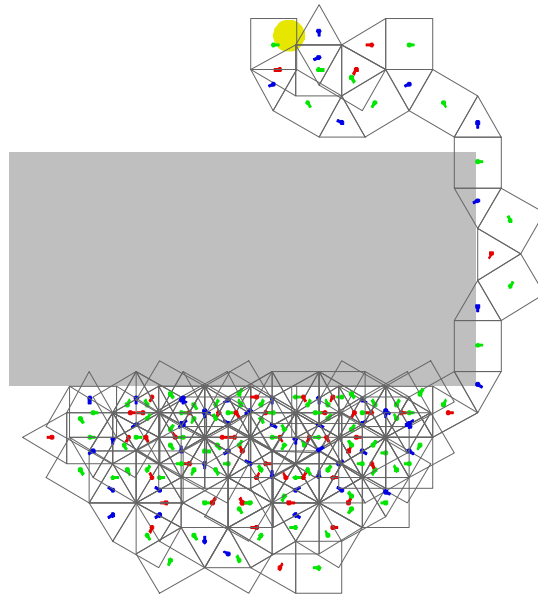
Last but not least, it would be interesting to physically construct a working prototype of the device.

## References

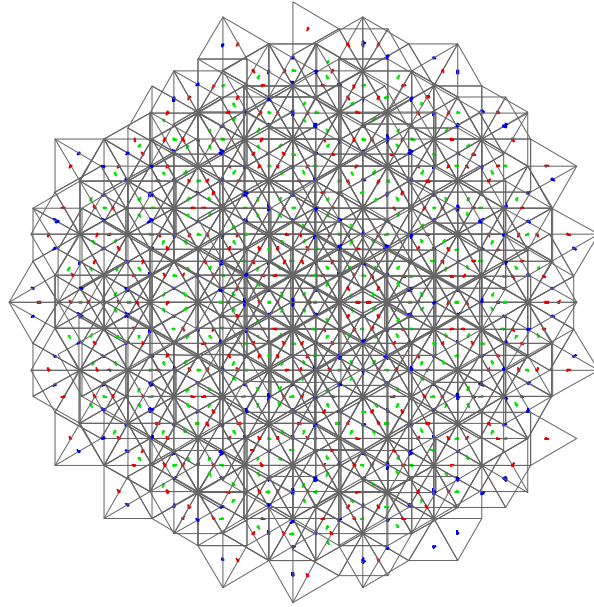
1. <http://bulletphysics.org/>
2. <http://guillaume.claret.me/bunach/>
3. P. Hart, N. Nilsson, B. Raphael, (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2): 100–107.
4. D. Knuth, (1997), The Art Of Computer Programming Vol 1. 3rd ed., Boston: Addison-Wesley.



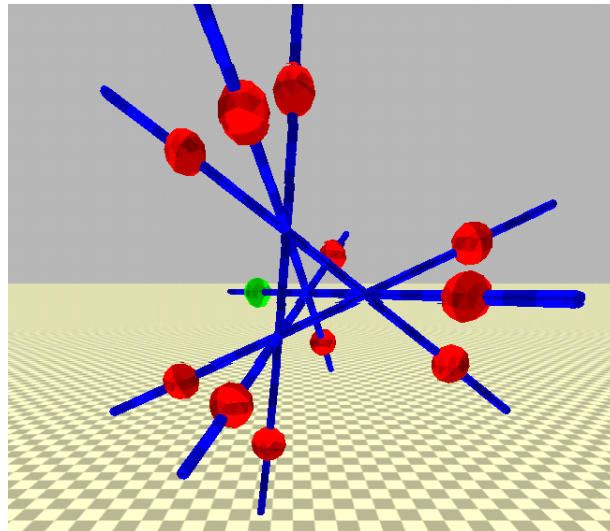
**Fig. 8.** Tristate Locomotion HD  $\leftrightarrow$  SD  $\leftrightarrow$  HU



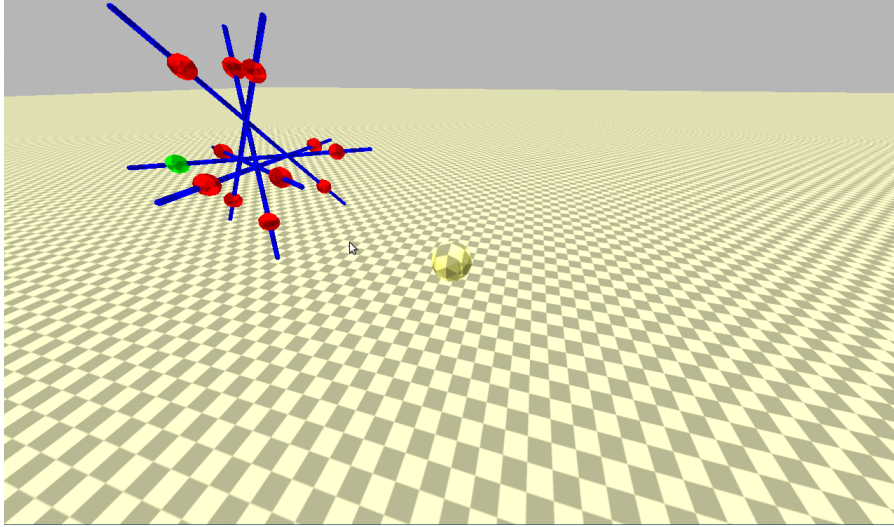
**Fig. 9.** A\* Obstacle Avoidance



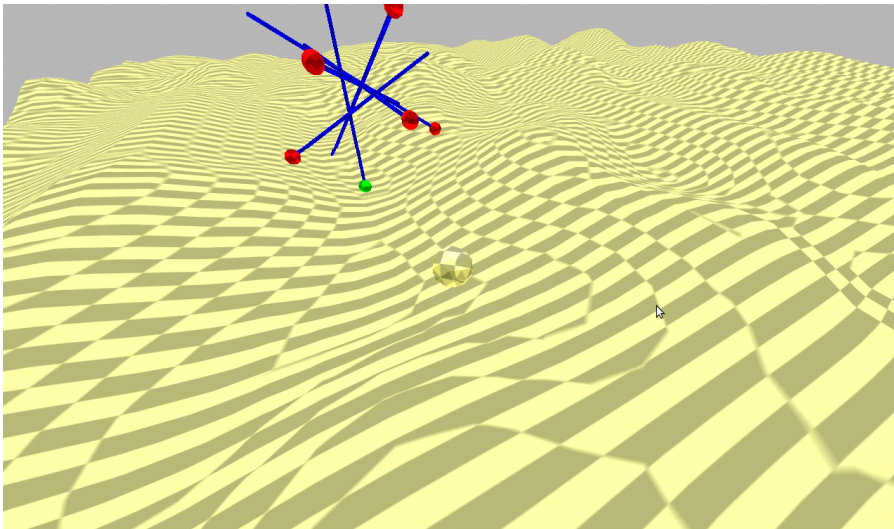
**Fig. 10.** Tristate Breadth First Search



**Fig. 11.** Bullet Simulation, Details of The Robot



**Fig. 12.** Bullet Simulation - Planar Locomotion



**Fig. 13.** Bullet Simulation - Non Planar Locomotion.