CS775 Design Document : Locomotion Skills for Simulated Quadrupeds

Tanmay Randhavane (110050010), Alok Yadav (110050043)

March 30, 2014

1 Introduction

In this paper [2], the authors have tried to solve the problem of modelling various sets of motions and gaits of a quadruped. Various actions that the authors tried to simulate were simple walk, trot, pace, canter, gallop, jumps over obstacles, falls and recovery after falls, sitting, lying down, getting up.

As part of this open project, we will try to simulate some of the actions implemented by the author. We will focus on six primary gaits which are simple walk, trot, pace, canter, transverse gallop and rotary gallop. After modelling this part, we will try to optimize these motions using the optimization techniques discussed in the section 4 of the parent paper [2].

2 Implementation Details

The implementation will involve implementing following parts:

1. **The model quadruped**

   We will model the quadruped according to the dimensions mentioned in the parent paper [2]. It will have a shoulder height of 47cm, a hip height of 45cm and a total mass of 34 kg. The model will be constructed using 30 links: 4 links for each leg, 6 for the spine/back, 4 for the tail, and 4 for the neck-and-head.

2. **The gait controller**

   The gait controller consists of components that can be adjusted to get the desired motion of the quadruped. Following controllers will be implemented in our simulation:

   (a) **Gait Graph**

   Gait graph captures the timing and relative phasing of the stance and swing phases of each of the legs. Gait graph has an associated stride period, T. Gait graph has two parameters, a gait phase and a swing phase. Some aspects of the motion are modelled as a function of the gait phase whereas other aspects are modelled as a function of swing phase.

   Gait graph will be defined by the parameters, swing phase and gait phase. The aspects of motion that these parameters will affect will be determined using the experimental data.

   (b) **Virtual Forces**

   The motion in the qudruped is directed by torques applied on various links. There are two primary sources of the torques, the virtual forces and the joint control. The used virtual forces are explained in the following subsections wherever they are used.

A virtual force, F, is applied by using the jacobian transpose to compute the required internal torques according to $\tau = J^T F$. We will implement this formula using matrix multiplication to get the value of torque.

(c) **Joint Control**

The second source of torque is the joint control. Each joint in the modelled quadruped will have a proportial-derivative (PD) controller. The PD-controller when given a desired orientation and the current orientation of the joint, returns a desired value of the torque that should be applied at that joint. We will use an external library for the PD-controller. The sum of torques from joint control and virtual forces will give the final torque that is to be supplied to the Forward Dynamics Simulation which will then simulate the equations of motion.

(d) **Leg Frames**

Leg frame is the link which connects the leg of the qudruped to the rest of the body. Each leg frame is connected to two legs. The leg frames are the links of the spine.

(e) **Stance Legs**

Stance legs are responsible for controlling the motion of the leg frames. There is a PD-controller at the leg frame joint, which computes the desired value of torque to be applied, τ_{LF}. This torque is a sum of the torques applied on the links connected to the leg frame, i.e., the stance leg(s), the swing leg(s) and the neighbouring joints in the spine. In order to achieve the desired τ_{LF}, the hip joint does not have any PD-controller. Thus, the torque at hip joint is computed as $\tau_{stance} = \tau_{LF} - \tau_{swing} + \tau_{spine}$. Here, τ_{swing} is the sum of the torques for all the swing legs attached to the given leg frame and τ_{spine} is an analogous quantity for the spine.

The motion of the leg frames is guided using three virtual forces, F_h, F_v and F_D. First, F_h, is used to make the leg frame to follow a height trajectory, we use, $F_h = k_p e + k_d \dot{e}$ where e is the difference between the desired height and the current height. F_v is used to regulate the leg frame velocity according to $F_h = k_v(v_d - v)$. Third, F_D implements a phase-dependent force that is customised for each stance leg. It is measured as a function of progress in the gait.

These virtual forces will control only the overall function of the stance legs. To determine the orientations of the internal joints of the leg, Inverse Kinematics will be used.

(f) **Swing legs** Swing legs have desired swing foot trajectories determined for each gait. The target location of the foot is determined using a velocity-based foot placement model: $P_2 = P_{LF} + (v_d - v)s_{fp}$. Here, P_{LF} is the default stepping location relative to the leg frame, P_2 is the target location and s_{fp} is the scale factor. Given the target position, IK is used to determine desired angles for individual joints and PD-controllers in turn compute the required torque.

(g) **Spine**

The spine contains n number of links. The difference between the front and the rear leg frame's orientation will be distributed evenly among the n-

1 joints of the spine. The PD-controllers on these joints will then compute the required torques. There will be a function to compute the difference in orientations which will use quaternions.

(h) **Neck and head**

The trajectories of head and neck will be a function of the gait phase and this function will be determined automatically during optimization to reduce head wobble.

(i) **Gravity compensation**

This is used to remove the impact of gravity on the swing legs, neck, head and spine. This is achieved using the virtual force, $F = -mg$ at the center of mass of the relevant links. The center of mass will be a position which will be determined during model building.

3. **The overall control loop**

The overall control loop, in each time step, computes the torques using the virtual forces and joint control torques and provides them to the Forward Dynamics Simulator. We will use Open Dynamics Engine(ODE) to simulate the forward dynamics simulation. The algorithm provided in the appendix of the parent paper [2] will be used to implement the overall control loop.

4. **Optimization**

The default initialization allows only a basic version of gaits. We will optimize these gaits using the motion capture data from [1]. In the optimization, we will try to minimize the following objective function,

$$f_{obj}(P) = w_d f_d + w_v f_v + w_h f_h + w_r f_r$$

where f_d measures the deviation of the motion from the data, f_v measures the deviation of the desired speed, f_h measures the head accelerations and f_r measures whole body rotations. A greedy stochastic local algorithm will be used to minimize the objective function.

5. **Results**

The results will be examined across following dimensions:

(a) Optimization

The simulation first will be run with controllers designed by hand and then it will be compared with optimized simulation. Both the videos will be shown side by side to show the difference created by optimization using the real data.

(b) Robustness

The resulting gaits will be tested to robustness against pushes. We will test this by applying forces to both the leg frames from front, back, left and right. The recovery from all these eight perturbations means that the gait is robust.

(c) Comparison to Motion Captured Data

We will compare the gaits of the simulated quadruped with that of the motion capture data and try to see how the various parts (legs, head and neck, foot placement) match to that of the motion capture data.

References

[1] A. Abourachid, M. Herbin, R. Hackert, L. Maes, , and V. Martin. Experimental study of coordination patterns during un-

steady locomotion in mammals. *J. Experimental Biology*, 210:366–377, 207.

[2] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics*, 30(4):59:1–59:12, July, 2011.