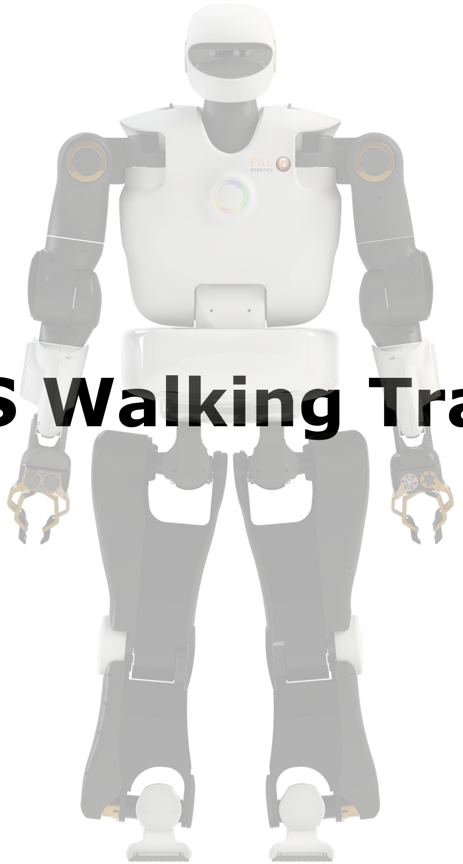
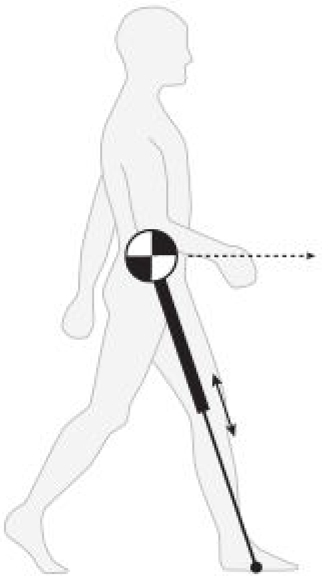
TALOS Walking Training



Why is it difficult to walk (for a robot) ?

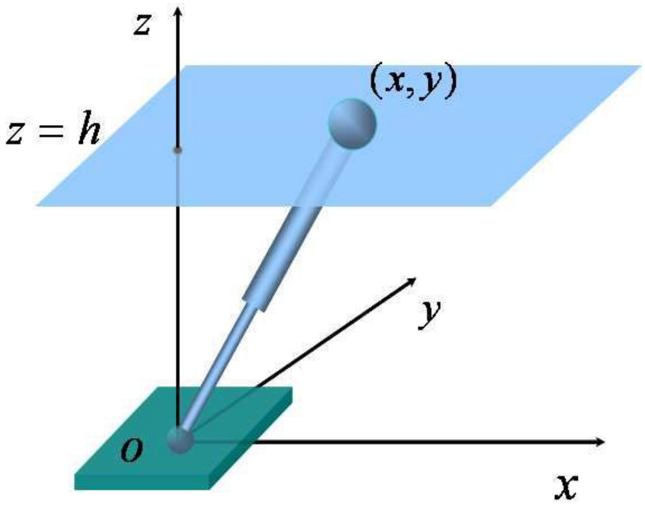
- high dimensional
- non-linear
- hybrid
- underactuated

Simplified Model



PAL ROBOTICS

Linear Inverted Pendulum Model (LIPM)



Dynamic equations for flat plane:

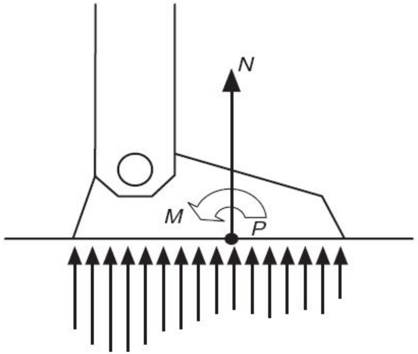
$$\ddot{y} = \frac{g}{z_c} y - \frac{1}{m z_c} \tau_x ,$$

$$\ddot{x} = \frac{g}{z_c} x + \frac{1}{m z_c} \tau_y ,$$

(x, y, z) : Center of Mass position
m : mass of the pendulum
z_c : CoM constant height
τ_x : torque around x axis
τ_y : torque around y axis

TALOS T.S.

Zero Moment Point (ZMP)



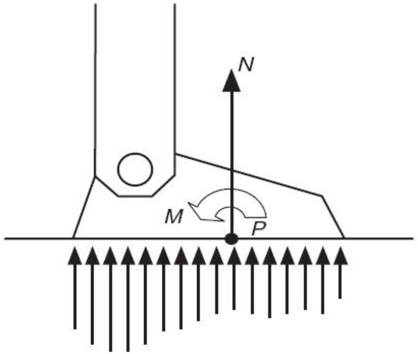
The ZMP was introduced by Vukobratovic and Stepanenko in 1972.

The distributed floor reaction force can be replaced by a single force N, acting on the Zero Moment Point (ZMP)

PAL ROBOTICS

TALOS T.S.

Zero Moment Point (ZMP)



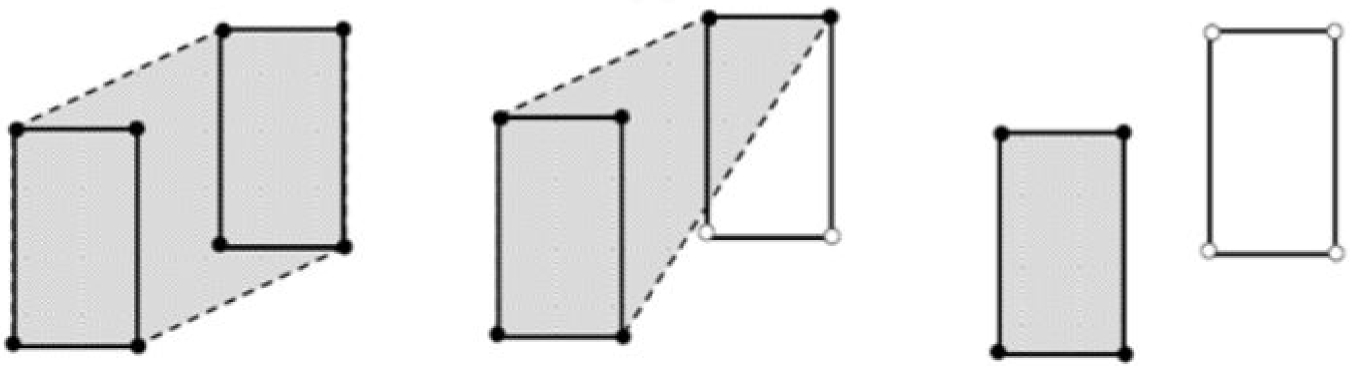
The necessary and sufficient condition for dynamic equilibrium is that for the point P on the sole where the ground reaction force is acting:

$$M_x = 0,$$
$$M_y = 0.$$

PAL ROBOTICS

TALOS T.S.

Zero Moment Point (ZMP)



(a) both feet support

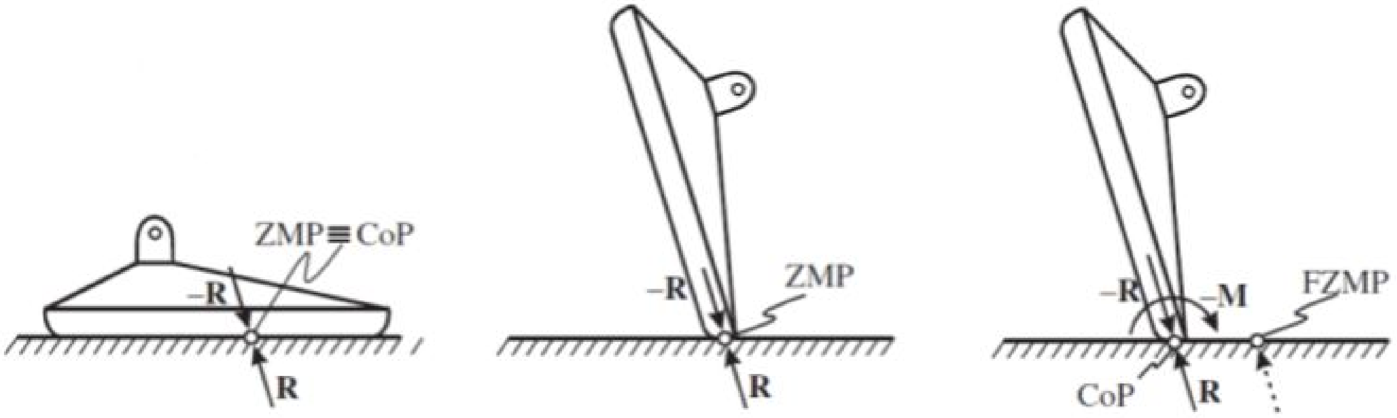
(b) right foot is touching the ground with toe

(c) single foot support

The ZMP has to stay within the support polygon

PAL ROBOTICS

Zero Moment Point (ZMP) & Center of Pressure (CoP)



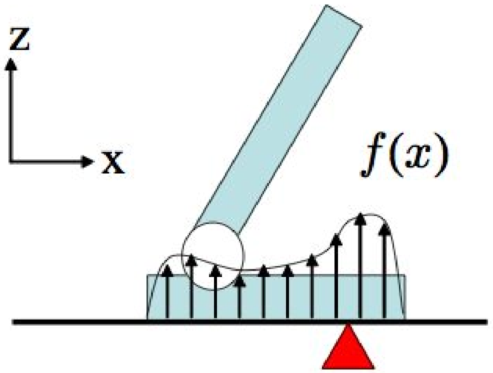
(a) Dynamically balanced case

(b) Tiptoe dynamic balance

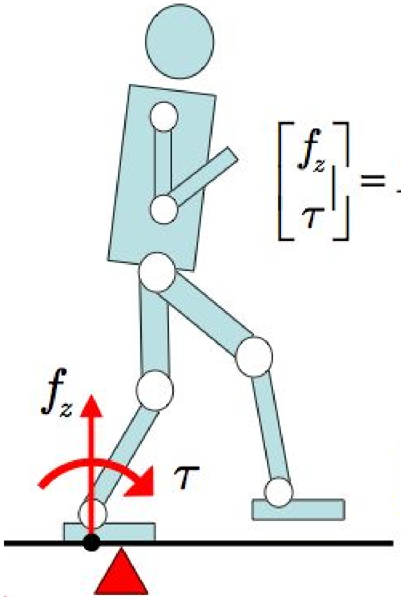
(c) Unbalanced case where the ZMP does not exist and the ground reaction force acting point is CoP while the point where $M_x = 0$ and $M_y = 0$ is outside the support polygon (FZMP)

PAL ROBOTICS

How to measure ZMP?



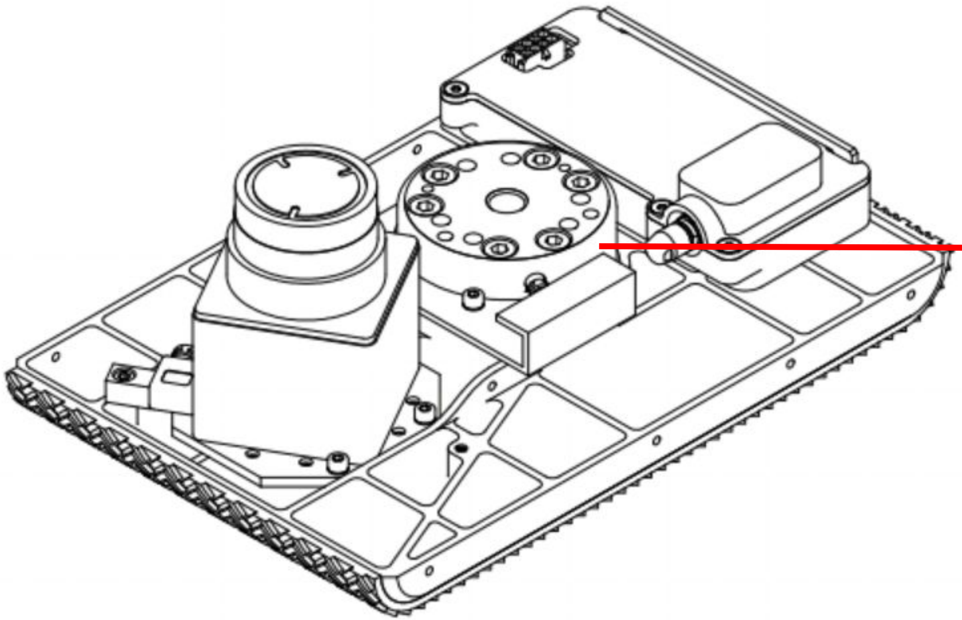
ZMP $p = \frac{\int x f(x)dx}{\int f(x)dx}$

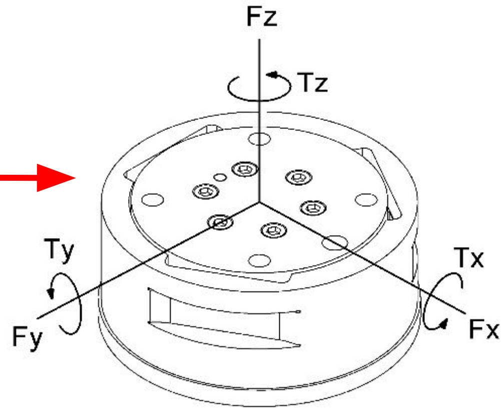


$\begin{bmatrix} f_z \\ \tau \end{bmatrix} = InvDyn(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$

ZMP $p = -\frac{\tau}{f_z}$

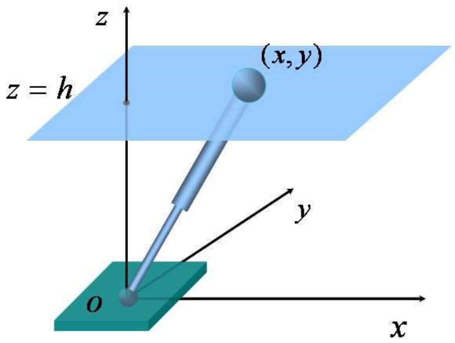
How to measure ZMP?





TALOS T.S.

3D-LIMP and ZMP



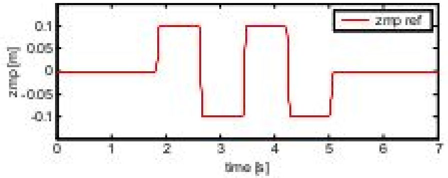
$$\ddot{y} = \frac{g}{z_c} y - \frac{1}{m z_c} \tau_x,$$
$$\ddot{x} = \frac{g}{z_c} x + \frac{1}{m z_c} \tau_y,$$

$$p_x = -\frac{\tau_y}{mg},$$
$$p_y = \frac{\tau_x}{mg},$$

$$\boxed{\begin{aligned} \ddot{y} &= \frac{g}{z_c}(y - p_y), \\ \ddot{x} &= \frac{g}{z_c}(x - p_x). \end{aligned}}$$

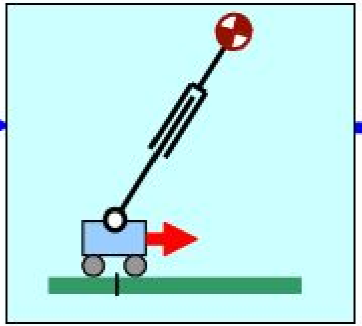
PAL ROBOTICS

Dual Systems - Inverted Pendulum

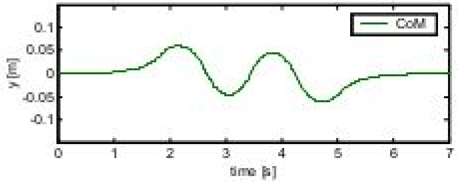


p

ZMP



$$\ddot{x} = \frac{g}{z_h}(x - p)$$

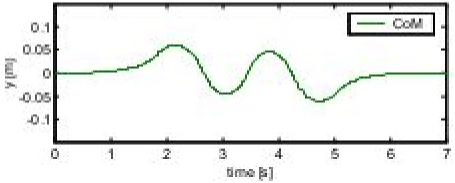


x

CoM

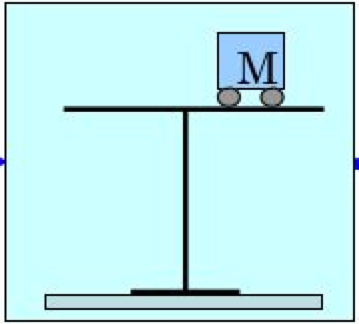
TALOS T.S.

Dual Systems - Table Cart Model

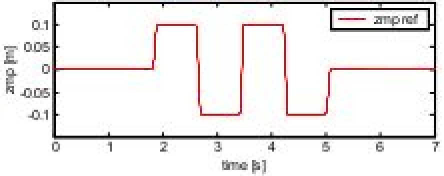


x

Cart trajectory



$$p = x - \frac{z_h}{g}\ddot{x}$$



p

ZMP

PAL ROBOTICS

ZMP Solver

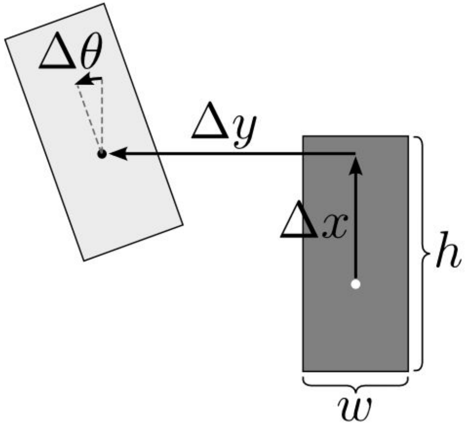
- Analytical solution method for planning simultaneously ZMP and COM trajectories

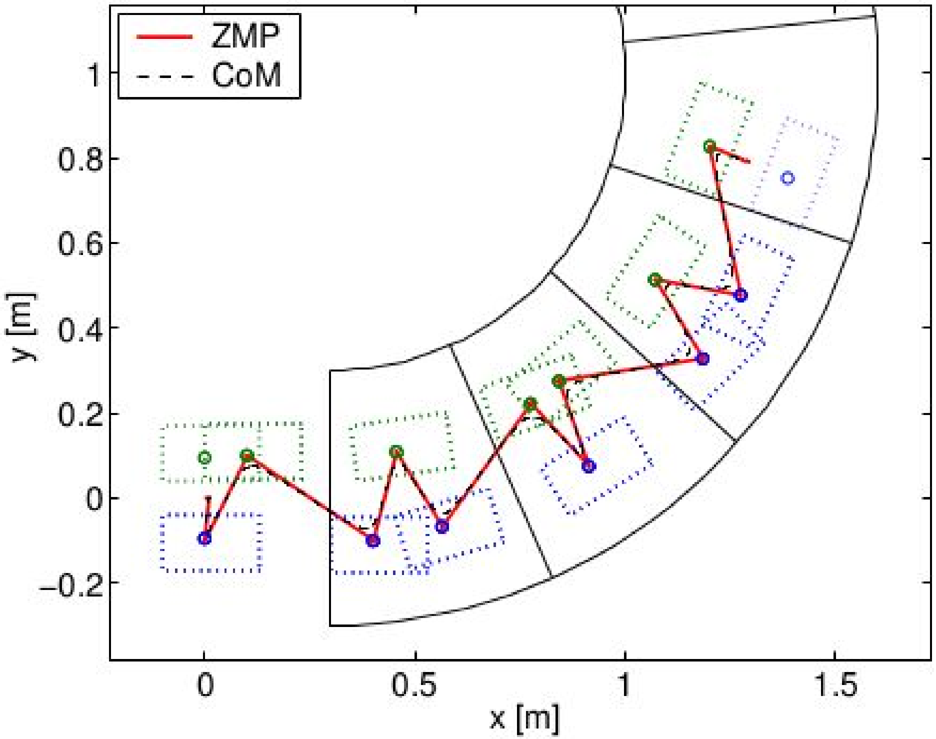
- Constraints on the ZMP/COM position and velocity ensure trajectory smoothness and balance

- **Reference:** "*An Analytical Method for Real-Time Gait Planning For Humanoids Robots". K. Harada, S. Kajita, K. Kaneko and H. Hirukawa (IJHR Vol. 3, No. 1 (2006) 1-19*

PAL ROBOTICS

Walking Pattern generation

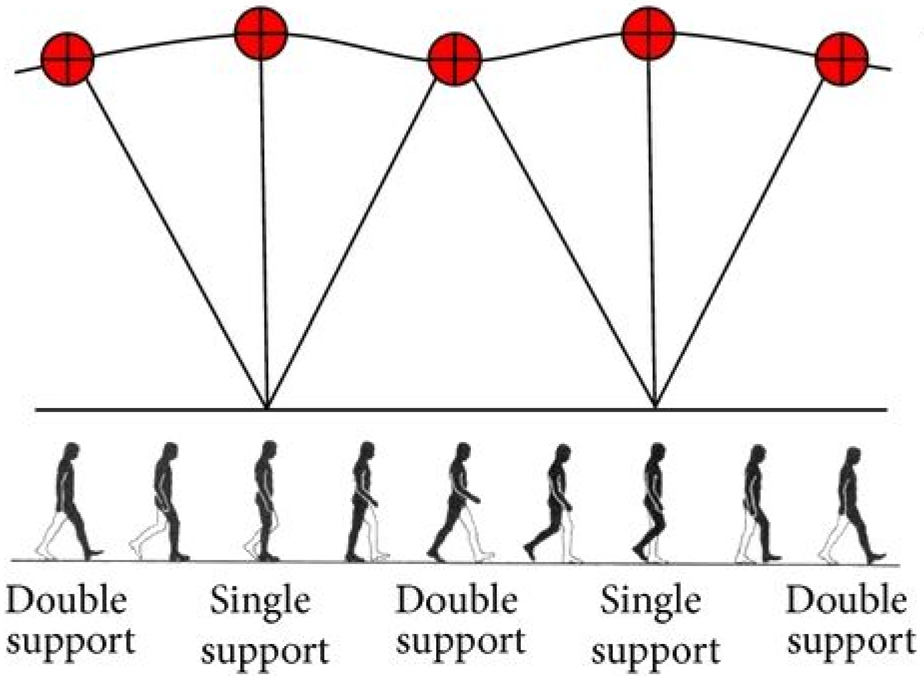




PAL ROBOTICS

TALOS T.S.

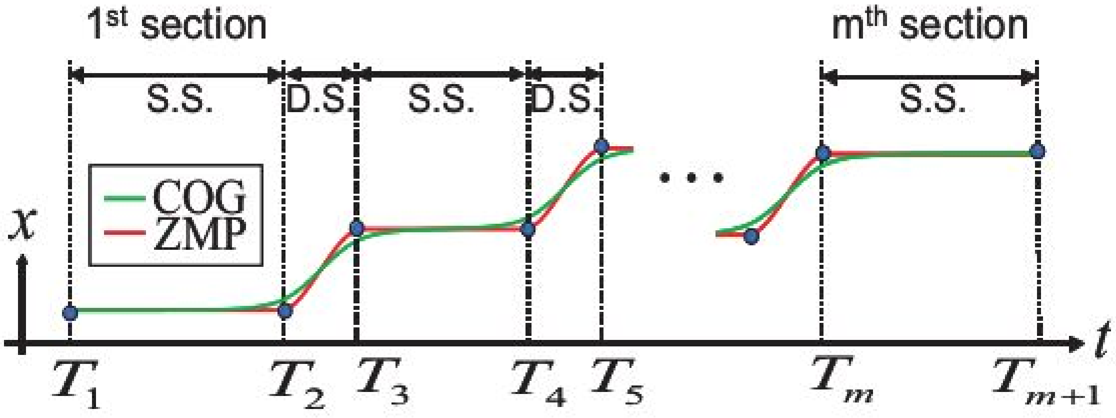
Walking Pattern generation



PAL ROBOTICS

TALOS T.S.

Walking Pattern generation

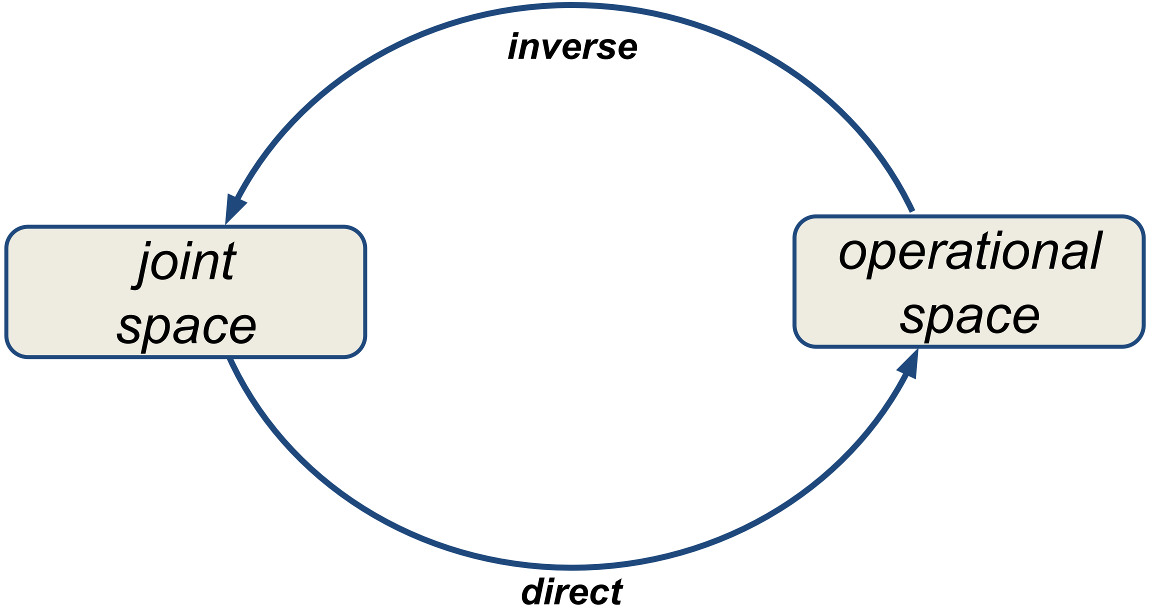


S.S. = single support phase

D.S. = double support phase

Step Time = Time_DS + Time_SS

Kinematics



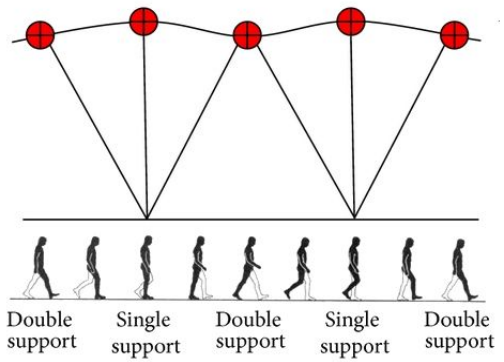
Inverse Kinematics

- Equations are in general non-linear and isn't always possible to find a closed-form solution
- Multiple solutions may exist
- Infinite solution may exist
- No admissible solutions due to kinematic structure
- Algebraic or geometric intuition
- Numerical solution techniques

PAL ROBOTICS

TALOS T.S.

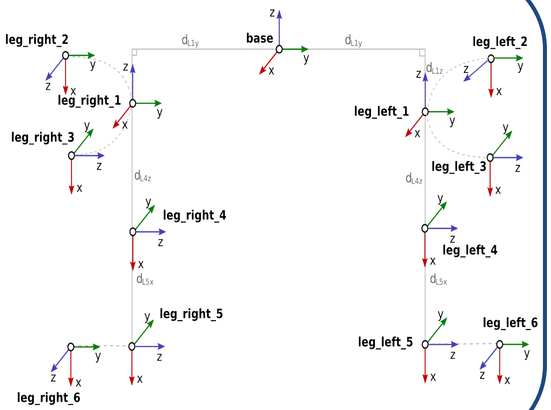
Inverse Kinematics



CoM
x

Foot
p

Leg
joints
q



Closed-form solution of leg IK

Given CoM and foot pose in *operational space*
we get a unique solution for *joint space* angle variables

PAL ROBOTICS

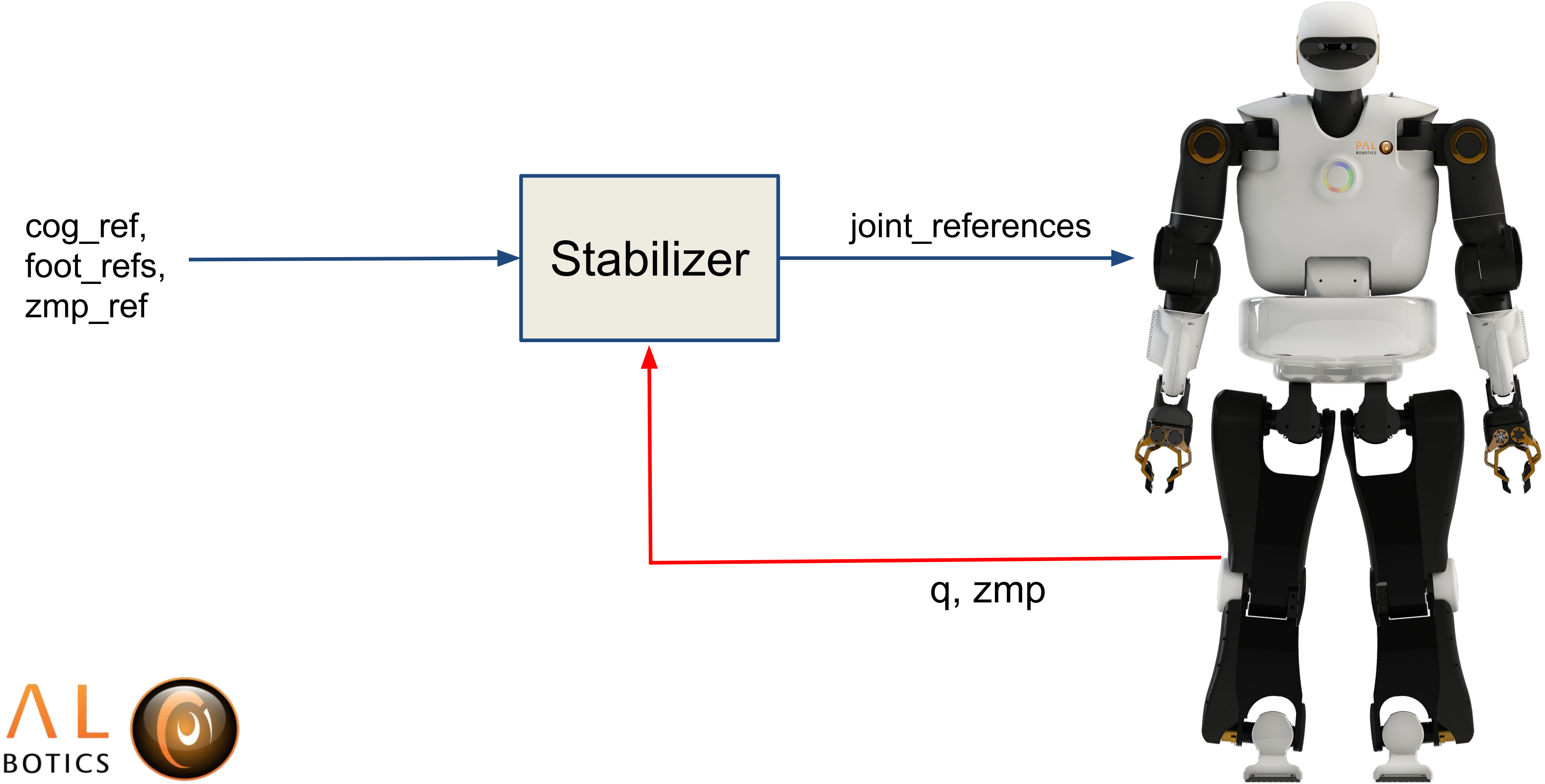
TALOS T.S.

Stabilizer

- Accounts for model inaccuracies
  - Nonlinearity
  - Multi-body dynamics

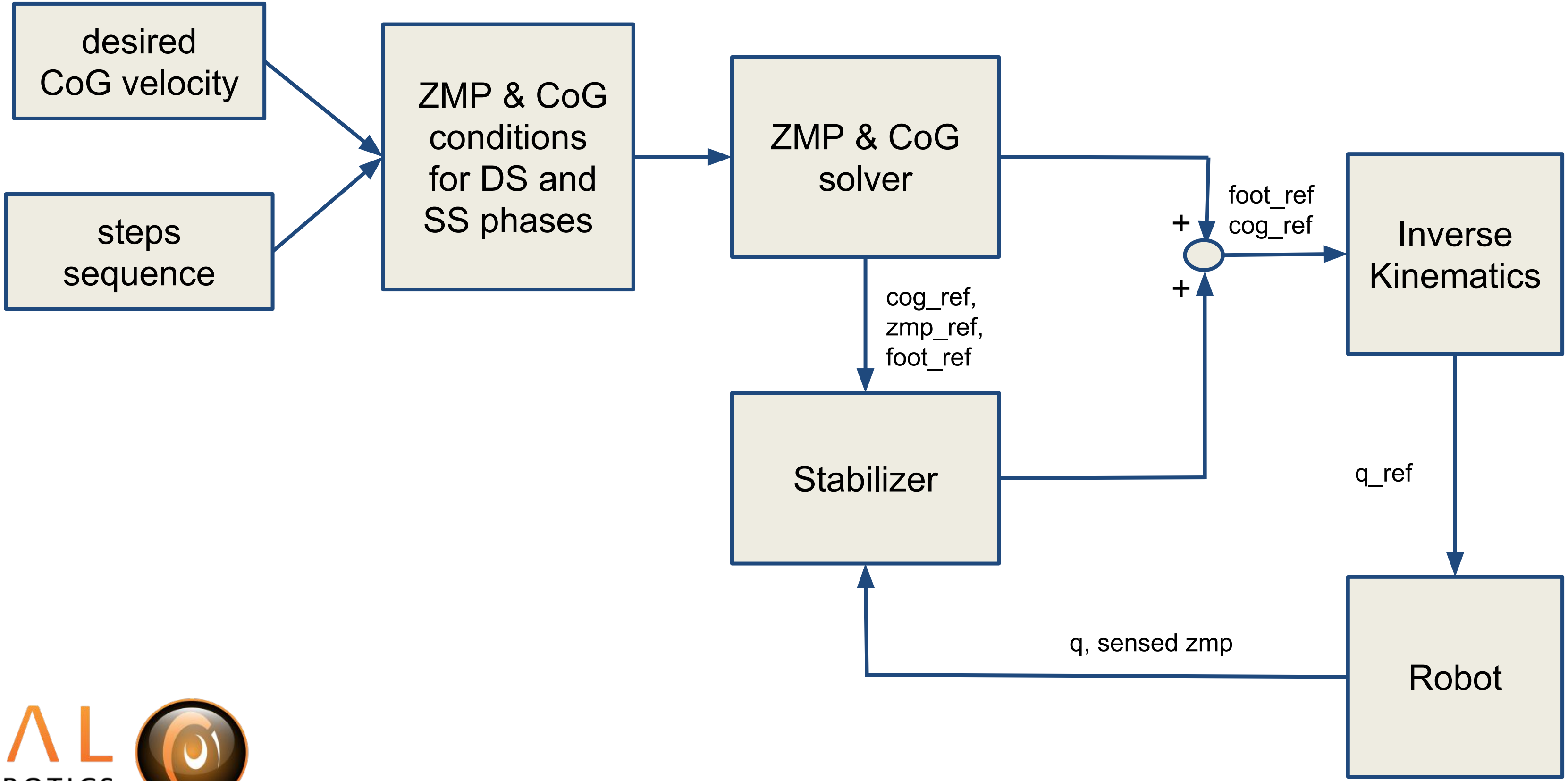
- Disturbances rejection
  - joint trajectory tracking errors
  - external forces (pushes)
  - oscillations due to rigidity of the robot
  - not perfectly flat terrain

Sensor feedback control



TALOS T.S.

Walking controller structure



TALOS T.S.

PAL ROBOTICS

Walking API

- Joystick
- Topic messages
- Service call
- Action goal

PAL ROBOTICS

TALOS T.S.

Walking API - Joystick

Walking API - ROS Topic Message

```
int main(int argc, char **argv)
{
  ros::init(argc, argv, "walking_client_topic");
  ros::NodeHandle n;
  ros::Publisher cmd_publisher =
            n.advertise<geometry_msgs::Twist>(WALK_CMD_VEL_TOPIC,1
        );

  geometry_msgs::Twist cmd_vel_forward;
  cmd_vel_forward.linear.x = 0.5;

  cmd_publisher.publish(cmd_vel_forward);
  return 0;
}
```

Walking API - ROS Service Call

```
int main(int argc, char **argv)
{
  ros::init(argc, argv, "walking_client_example");
  ros::NodeHandle n;
  ros::ServiceClient walking_client =
            n.serviceClient<walking_msgs::WalkSteps>(WALK_SERVICE);
  if(! walking_client.waitForExistence(ros::Duration(5.0)) )
  {
    ROS_ERROR_STREAM("...");
    return 1;
  }
 ....
```

PAL ROBOTICS

Walking API - ROS Service Call

```
walking_msgs::WalkSteps srv;
srv.request.nsteps = nsteps;
srv.request.step_length = step_length;
srv.request.step_time = step_time;

if (walking_client.call(srv)) {
  ROS_INFO("Succesfully called service WalkSteps");
} else {
  ROS_ERROR("Failed to call service WalkSteps");
  return 1;
}
```

Walking API - ROS Action client

```
void doneCb( const actionlib::SimpleClientGoalState& state,
             const humanoid_nav_msgs::ExecFootstepsResultConstPtr& result)
{
  ROS_INFO("Finished in state [%s]", state.toString().c_str());
}

void activeCb()
{
  ROS_INFO("Goal just went active");
}

void feedbackCb( const humanoid_nav_msgs::ExecFootstepsFeedbackConstPtr& feedback)
{
  ROS_INFO_STREAM("Got Feedback : steps " << feedback->executed_footsteps.size() << " executed");
}
```

PAL ROBOTICS

Walking API - ROS Action client

```
typedef actionlib::SimpleActionClient<humanoid_nav_msgs::ExecFootstepsAction> WalkingClient;

int main(int argc, char **argv)
{
  ros::init(argc, argv, "walking_client_example");
  ros::NodeHandle n;
  WalkingClient action_client(WALK_STEPS_ACTION_NAME, true);
  if(!action_client.waitForServer(ros::Duration(10.0) ) )
  {
    ROS_ERROR_STREAM("...error...");
    return 1;
  }
  …
```

PAL ROBOTICS

Walking API - ROS Action client

```
humanoid_nav_msgs::ExecFootstepsGoal goal;
humanoid_nav_msgs::StepTarget foot;
for(unsigned int i=0; i <= nsteps; ++i) {
  foot.leg = foot.leg == humanoid_nav_msgs::StepTarget::right?
       humanoid_nav_msgs::StepTarget::left : humanoid_nav_msgs::StepTarget::right;
  if(i < nsteps ) {
    foot.pose.x = step_length;
    foot.pose.y = -HIP_SPACING*(2.0 - 4*foot.leg);
    foot.pose.theta = 0;
  }  else {       // last step with zero lenght
    foot.pose.x = 0;
    foot.pose.y = -HIP_SPACING*(2.0 - 4*foot.leg);
    foot.pose.theta = 0;
  }
  goal.footsteps.push_back(foot);   // Create a list of steps
}
```

PAL ROBOTICS

TALOS T.S.

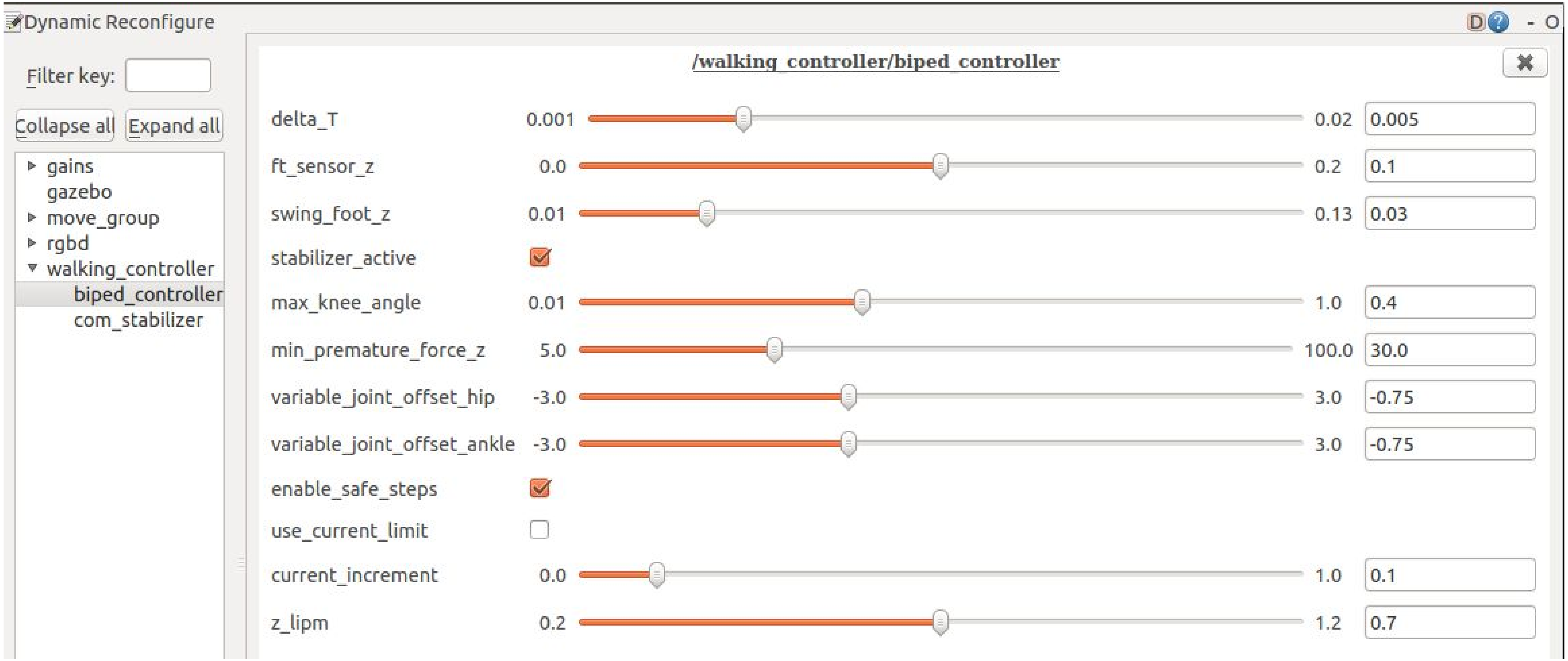
Walking API - ROS Action client

```
goal.feedback_frequency = 1.0;

action_client.sendGoal(goal, doneCb, activeCb, feedbackCb);

action_client.waitForResult(ros::Duration(20.0));
if (action_client.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
{
  ROS_INFO("The footstep list has been excecuted succesfully");
}
```

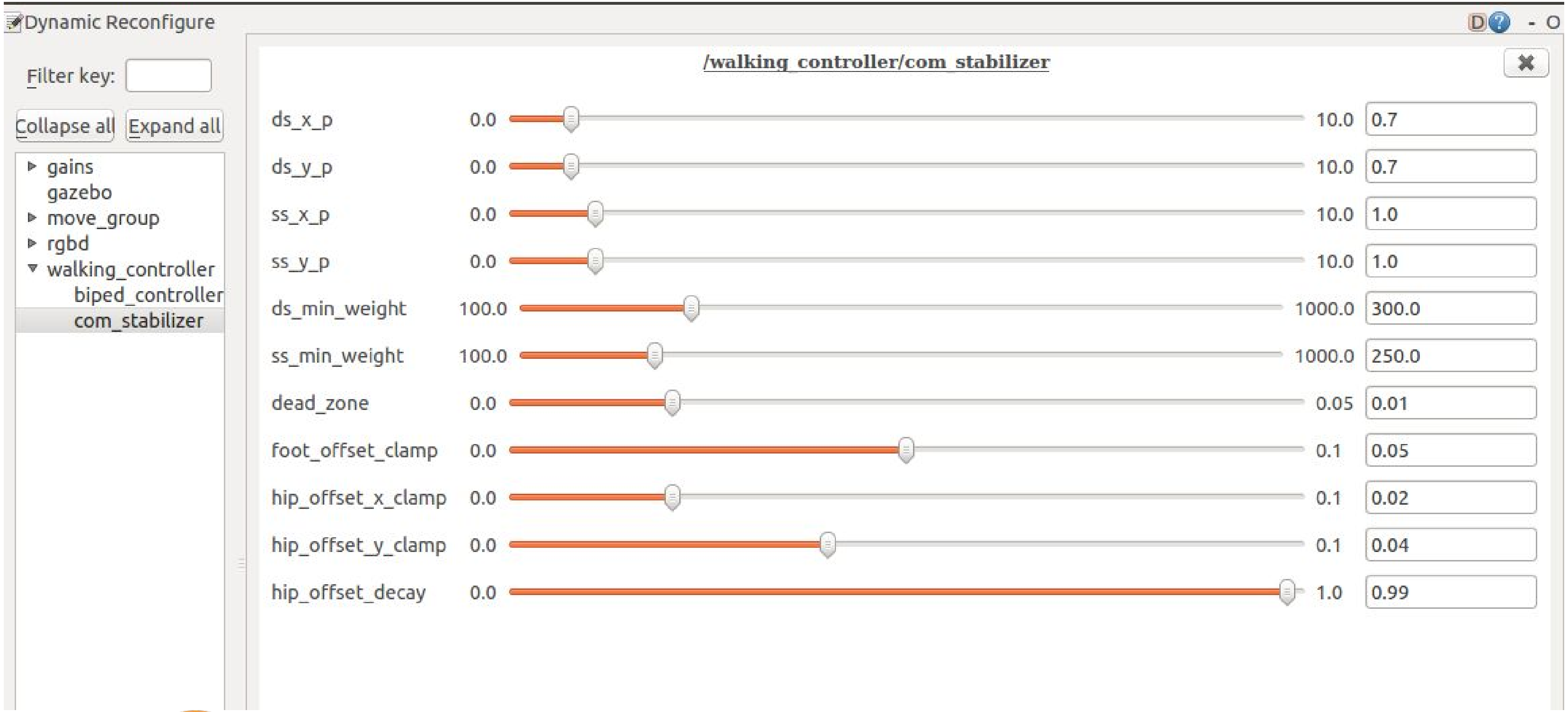
Walking parameters - Biped Controller



PAL ROBOTICS

TALOS T.S.

Walking parameters - Stabilizer



TALOS T.S.

PAL ROBOTICS