Georgia Institute of Technology

Center for Robotics & Intelligent Machines

Humanoid Robotics Lab

Hubo Notebook

Peter Vieira

July 09

Progress

- Cleaned up nudgeHips() walking controller with Grey and Matt. Need to debug IK issues because the feet compensated way farther than they should have when we ran it in the air.
- Tomorrow's plan is to finish debugging the NAN's in the control daemon, debug the IK, and test hip nudging with slowing walking speeds.

Status

- Changed zmp-daemon and zmpgui to use pointers on the heap instead of primitives on the stack because the stack was being overflowed after increasing zmp traj ach channel size and zmp max traj size to accomodate slower walking gates.
- Matt, Grey and I debugged some issues in the Walker's nudgeHips() controller, but got detoured due to CAN wire issues.
  - When we homed once it worked. Homing the second time, the upper body didn't respond. The RWR motor control board CAN wire was getting pulled out or something during the first home, which caused the upper body to fail the second time around. This was repaired at the end of the day.
  - We tried after temporarily fixing the above issue and the legs stopped responding during the phase of getting into the initial position of the trajectory. Discovered that the control daemon was computing NANs somehow. Grey and I discovered some weird things going on using printfs but still haven't found the exact cause yet.

July 08

Status

- Figured out that the reason hubo_walk was not working and during make, errors involving boost/operators.hpp and ros/console-brdige/console.h and LogLevel. The reason was because I moved 'include "Hubo_Control.h"' from balance-daemon.cpp to balance-daemon.h, which include hubo.h which include syslog.h which has pound defines that conflict with ros. This is because hubo_walk includes balance-daemon.h.

July 07

Status

- Calibrated Hubo's legs. Realized that Hubo doesn't hang level on the hoist, which can significantly affect calibration if using a level.
- Tried balancing first and using the ankle roll offsets from the balancing in the Walker, but that didn't help things.

July 06

Status

- Determined that the jittery behavior when walking was due to using std::cerr instead of std::cout. This is because cerr is unbuffered and sends everything to the terminal immediately, causing a slowdown in the trajectory, where as std::cout buffers it and then sends a batch to the terminal. With std::cout it was much better, but possibly still affecting it a bit.

- Tested quasistatic walking in place using the nudgeHips function (see June 03). Even after calibrating the feet some, it still had the same problem. Maybe more calibration is necessary. I believe that for such feedback control to work properly, the feet need to be flatten before walking, and then the offset used throughout the walking trajectory, otherwise there's no way to ensure the legs and ankles are perfectly calibrated and it won't happen again.

Ideas

1. Always balance before walking and use ankle offsets in walking trajectory.
    - Doing this as is would mean when walking started, Hubo would "jolt" a bit forward to get to the initial ZMP_x offset, which is currently 0.04m.
    - Or using the COM height and the ZMP_x offset to compute the new desired IMU angle about the y (pitch) axis.
2. Use accelerometers in the feet to flatten the feet without affecting the body position.

July 05

Status

- Added installation to the CMakeLists.txt in hubo-motion-rt, and also added an uninstall cmake file so you can now type 'sudo make uninstall' in the build directory.
- Found old, uncorrected bug in control.table in hubo-motion-rt. The hip yaw and hip pitch for both legs were swapped, which affect how they get ordered in the leg joints arrays.
- Updated Walker.cpp to adjust both legs in both single and double support, and tested it on Hubo without actually changing the trajectory, but somehow it caused some jittery behavior. I will investigate this tomorrow.
- Added single and double support hip nudge gains to the hubo_walk panel in rviz, and added the gains to the balance-daemon accordingly.
- Updated leg lengths constants and limits in HuboKin.cpp and the control.table in hubo-motion-rt.

Issues

- Jittery behavior from nudgeHips() function in Walker.cpp.
- zmpdemo.cpp has TopicUnaligned... eigen error in the gui demo section.
- hubo_init get ACH_OVERFLOW from the hubo-state channel at times. Maybe a race condition since it uses threads.

July 04

Status

- Added nudgeHips() function to Walker.cpp.

July 03

Status

- Plan for walking:
  1. Try quasistatic walking with 0 step height, with the battery in.
  2. Try quasistatic walking with flattening of the feet.
  3. Try dynamic walking with flattening of the feet.
  4. Try running the zmp-daemon realtime using the force/torque sensor and maybe the accelerometer in the pelvis to determine the real ZMP and use that in the Preview Controller every timestep.
- 1. Quasistatic shifting back/forth worked very well without the battery.
- 2. Quasistatic walking in place worked well, except that the miscalibration in the left ankle roll caused the robot to fall to the right during single-support, which then caused the dropping leg to hit the ground and push the robot over. But the right side was perfect. Another possible explanation that Grey mentioned was that the new adapter piece between the right F/T sensor and the ankle is a bit thicker (or the calibration of the legs are such that the body is leaning to the left) causing Hubo to lean to the left, which then causing him to fall right when he lifts his right leg.
  - x-offset = 0.4m
  - y-offset = 0.0m
  - zmp-R = 1.00 x 10^8
  - lookahead time = 2.5s
  - startup = 1.0s
  - shutdown = 1.8s
  - double-support = 2.5s
  - single-support = 1.0s
  - pause time = 0.3s
  - step height = 0.06m
  - step distance = 0.0m
  - half stance width = 0.0885m
  - COM height = 0.45m
  - flattening gains = 0.001
  - decay gain = 0.5
  - min force threshold = 12.0N
  - max force threshold = 55.0N
- 3. Dynamic walking didn't work well with the various parameters Matt and I tried. Hubo's joint shifting and body leaning became out of phase by a step each time (ie. when his legs were such that they would make him shift left, he was leaning right on the right edge of his feet.).
- Matt and I discussed the implementation of torque feedback to nudge the hips in the opposite direction of the falling and I will be ready to test it on Friday morning.

$$\begin{bmatrix} f_x \\ f_y \\ f_z = 0 \end{bmatrix} = \begin{bmatrix} dt * K_p & 0 & 0 \\ 0 & dt * K_p & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_{x_{measured}} - \tau_{x_{expected}} \\ \tau_{y_{measured}} - \tau_{x_{expected}} \\ \tau_z = 0 \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} b_x \\ b_y \\ b_z = 0 \end{bmatrix} += \begin{bmatrix} \cos(\theta_{HY}) & -\sin(\theta_{HY}) & 0 \\ \sin(\theta_{HY}) & \cos(\theta_{HY}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z = 0 \end{bmatrix} \tag{2}$$

$$FootTF[LEFT] = legFK(qPrev[LEFT]) \tag{3}$$

$$FootTF[RIGHTT] = legFK(qPrev[RIGHTT]) \tag{4}$$

where τ_x and τ_y are the measured torques minus the expected torques for the x and y axes. f_x and f_y are the x- and y- position errors of the feet given the torque feedback error. b_x and b_y are the body x- and y- position errors with the hip yaws taken into account. This then gets premultiplied by the TFs from the body to the feet to get the new feet positions, on which we run the IK to get the new leg joint angles that will nudge our hips to counter any falling motion. If we integrate the body error, than we can save the accumulated error in the legs joint with two numbers and apply IK to.

Issues

- The left PC has login issues. To solve it, press Ctrl-Alt-F1 to go to the Terminal. Then type sudo chown groupname:username .Xauthority. Then press Ctl-Alt-F7 to go back to the login and try logging in.

July 02

Status

- Found a weird bug in zmpwalkgenerator.cpp, line 416 in github.com/swatbotics/hubomz/zmp/. If the center of mass IK fails then it gets the body transforms using the walker's local initContext, but it has not been set yet, so jvalues.size() = 0 and therefore the assert in KinBody::transforms() fails.
- Tested quasistatic walking with 1.5 double-support time. When it first moved to the left foot it did well, but when it shifted to the right, it went way too fast and overshot and fell over. We presume this may be because the double-support time is too short, but just increasing this doesn't seem like a good idea.
- Figured out that the footstep generation functions don't know anything about the zmp position so the final step ends with the body shifted to the side with the zmp_y at it's max. So stayDogStay() using the shutdown_ticks is necessary to bring the zmp_y back to zero.
- Tried much longer double-support time with quasistatic walking in place. The first shift to the left foot was perfect, but when Hubo put his right foot back down he fell off balance and then fell to the right as he shifted right. Will implement a feedback controller tomorrow.

Issues

- In quasistatic mode the single and double support times need to be very long, which creates a very large trajectory, which currently overflows the ach channel. So we can either increase the ach channel size (temporary plan), or move onto atomic trajectories.

July 01

Status

- Worked with Matt and found bug in hubomz code in zmpwalkgenerator.cpp where the if else statements for pause_time were switch, so it was interpolating the body rotation over the wrong number of iterations.
- Tested quasistatic walking on DRC Hubo 1, but when he leaned over the left foot, he ended up falling to the left side. My deduction is that the model is not correct (ie. the center of mass is too low) which caused Hubo to lean to far and thus put his ACTUAL center of mass outside his support foot.
- Parameters were: X-offset: 0.04, Startup time: 1.0, Shutdown time: 1.0, Double-support time: 0.9, Single-support time: 1.0, Pause time: 0.3, Step height: 0.1, Step distance: 0.22, Lateral distance: 0.0885, Center of mass height: 0.45.
- I added a function to the Walker class in hubo-motion-rt called balanceAboveFoot that will hopefully help account for the inaccuracies in the model and allow for quasistatic working to work.

Issues

- We discovered that the Force/Torque sensors in the ankles were reversed on all the signs when using Hubo-i, but when using hubo_init, just the right ankle was incorrect.
- Kiwon Sohn contacted KAIST and they said they purposely set the F/T sensors backwards and since defaulting to the stock firmware will revert them to being backwards, we should just change it in our code. This is not logical at all, but not much we can do about it now.

June 25

Status

- Ported our zmp-daemon code over to Matt's new hubomz repo which now works with the new DRC Hubo. Got everything working (ie. forward/backward, sidestepping and turning in place) except that there is something weird going on in the first step of the turn in place.
- Currently the repo is forked to my user account "pvieira3".

Issues

- Need to think about best and most logical way to use ZMPParams and where to put them. Currently I have put the ik_sense and the walktype in zmp-daemon and zmpwalkgenerator includes zmp-daemon. This led to changing some lines in zmpdemo as well.
- Need to fix turning in place.

June 22

Status

- Updated hubomz: merged grey/daemonize into master and cleaned up the code some.
- Now using master for hubomz and hubo_walk, manipulation for hubo-motion-rt and develop for hubo-ach.
- Push Hubo's plastic pieces back on and packed everything for Drexel.

June 21

Status

- On local PC start 'roscore' and 'rosrun rviz rviz'
- On hubo-vision start 'roslaunch openni_launch openni.launch'
- 'export ROS_MASTER_URI=http://COMPUTER_NAME:11311/'
- On local PC run 'rosrun rqt_reconfigure rqt_reconfigure' and check the camera/driver depth_registered box
- On local PC run 'roslaunch hubo_motion_ros hubo_manipulation.launch' (to see camera and hubo)
- On local PC run 'rosrun hubo_drc_vision segment_image'

June 20

Status

- Wrote a function to test the handle_angles function in the manipulation-daemon and tested it. It worked fine.
- Read through basic understanding of Point Cloud Library.
- Wrote function to convert pcl PointCloud2 from camera to a PointXYZRGB and then compute the point cloud normals and store the two together as a PointXYZRGBNormal to be used for segmentation of planes

June 19

Status

- Added function in manipulation daemon to handle raw joint angles for the arms instead of poses, because the IK was coming up with different angles than what was put into the FK on the ROS side.
- Change board limits for RSR and LSR so that going in toward Hubo's center the limit is 0.36 radians. Somehow they got messed up and were at 0.157 radians. Changed control.table to reflect this.
- Went through ROS tutorials: "Services & Parameters", "Using rqt_console and roslaunch", "Using rosed to edit files in ROS", "Creating a ROS msg and srv", "Writing a Simple Publisher and Subscriber", "Examining the Simple Publisher and Subscriber", "Writing a Simple Service and Client", "Examining the Simple Service and Client".

June 18

Status

- Discovered control table has conservative limits on shoulder roll joints. Currently +/- 0.3, but really should be +/-0.4. Discuss reasons with Grey. roslaunch hubo_motion_ros hubo_state.launch.

Todo

- Get more accurate TF to checkerboard (Add to code)
- Change manip daemon to use joint angles
- Write test code for joint angles manip cmd
- Get better calibration points

June 16

Status

- Fixed bug in zmp-daemon which caused discontinuity
- Got rid of unused code in zmp-daemon
- Merged risky/daemonize into grey/daemonize

June 13

Status

- While testing current control using KAIST's method, Grey discovered that we'd been using Dan Lofaro's filter b/c in the control daemon use_filter was set to false and then immediately set back to true.
- This cause lots of distortion using a .25s window, which screwed up the timing of the zmp walking.
- We tested trajectory without the filter and after fixing a bug it works fine now and the timing is accurate!

June 03

Status

- Started writing zmp atomic trajectories concept.
- Basically, slice trajectories up into their individual steps so we can just piece them together. This way we can generate all atomic trajectories for each walk type all at once, and then use them on the fly, which gets rid of the generation time when changing directions.

June 01

Status

- Wrote preview controller for model that includes angular momentum. It's located in ZmpPreview.h in hubomz/src
- Doesn't converge with current values

May 28

Status

- Changed zmp-daemon to send one trajectory and have zmpgui/walker reuse the periodic portion of it.
- This decreased the cpu usage of zmp-daemon from 100% to about 2%.

May 13

Status

- Got OpenGL zmpgui to run trajectories continuously.

May 12

Status

- Added keyboard controls to zmp-daemon for walking forward/backward, sidestepping and turning.

May 11

Issues

- Walktype "canned" doesn't account for initial conditions. Need to get rid of it and use footstep.cpp walk functions
- Where is best swap-in point? At end of double-support, have max CoM and joint velocities.

Thoughts

- Use last ZMPReferenceContext element in the ref vector in walker after adding 2nd to last step as the initContext for the next trajectory.

May 10

Issues

1. Discovered ach doesn't work between computers with different architectures. Need to made struct variables uint#, etc. and use __attribute__((packed))
2. Swapping in trajectories results in large joint velocities (ie. discontinuity). Something is wrong!

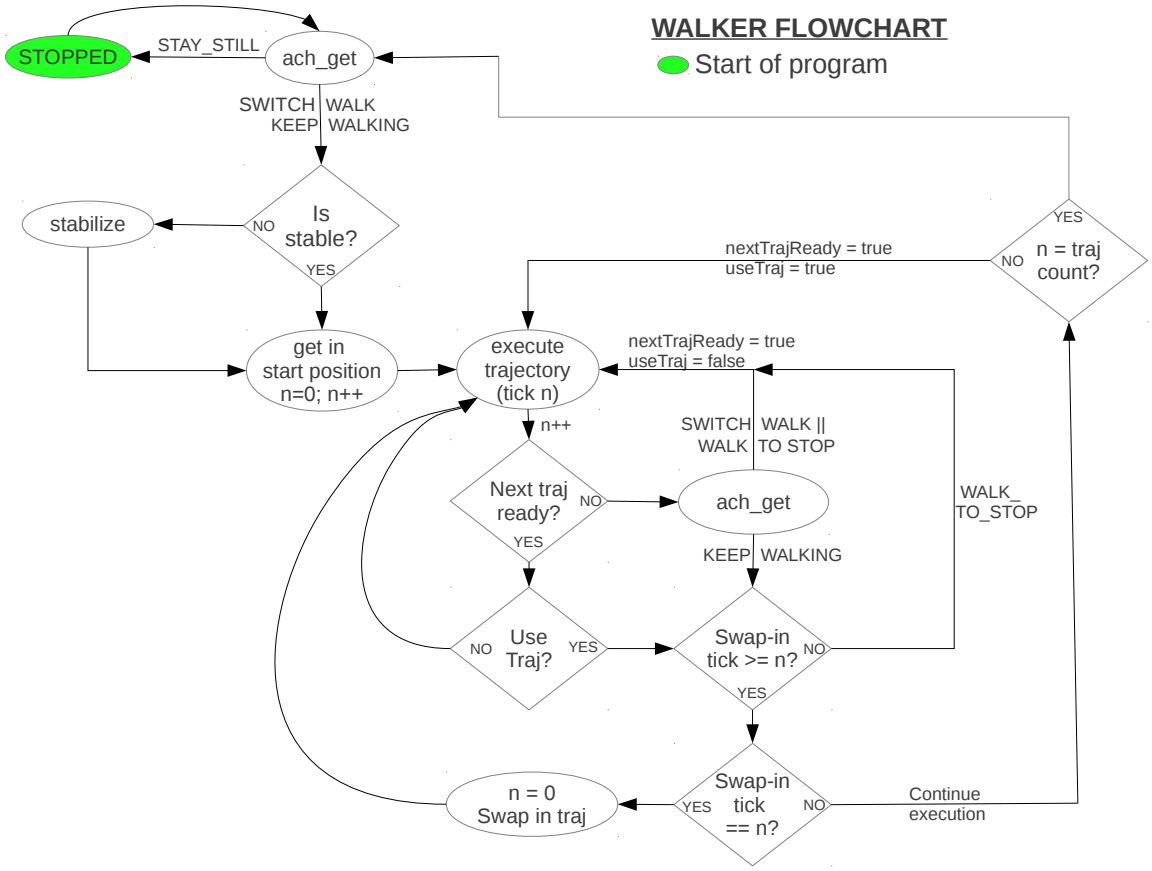
Thoughts

1. - Change structs that we're passing over ach to __attribute__((packed)) and specify variable sizes using inttypes.h (ie. uint8)
2. - Does walker modify joint angles at all?
   - Validate swap point inside zmp-daemon
   - Plot it

May 09

Status

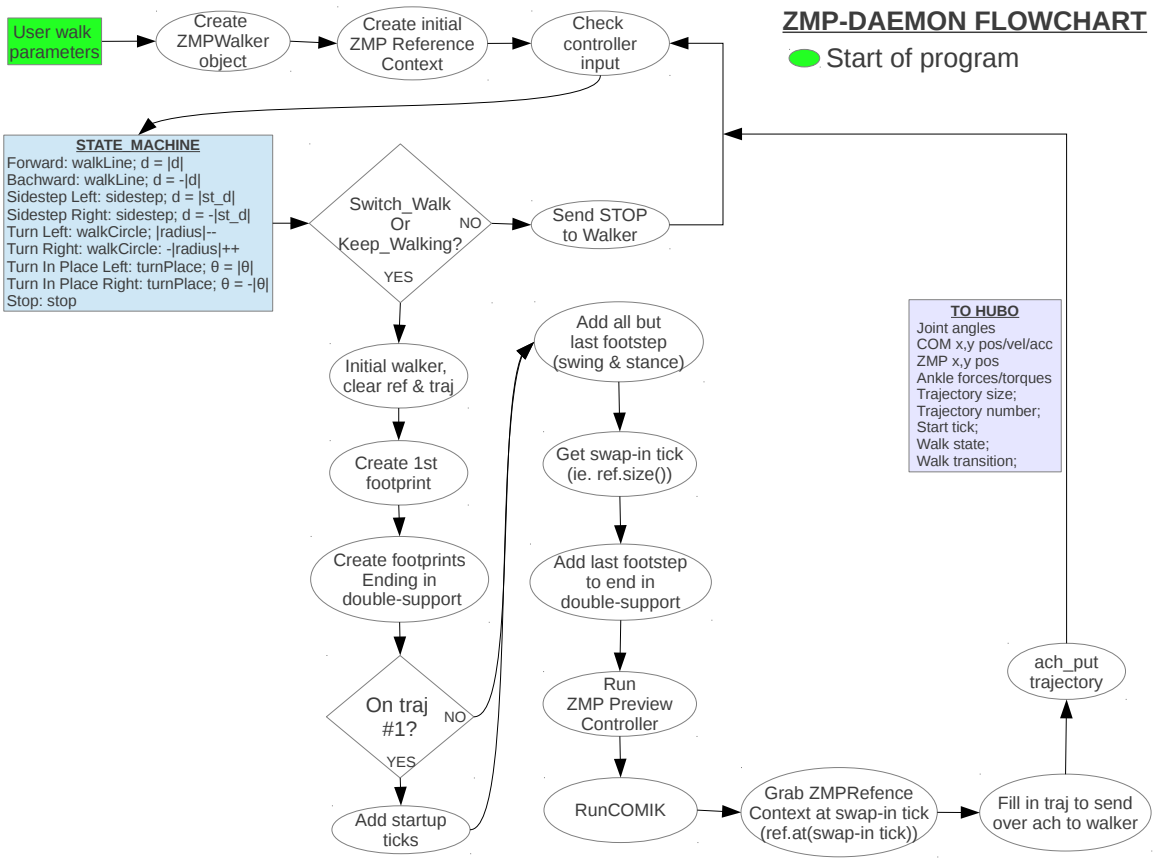
- Created walker flowchart and state machine



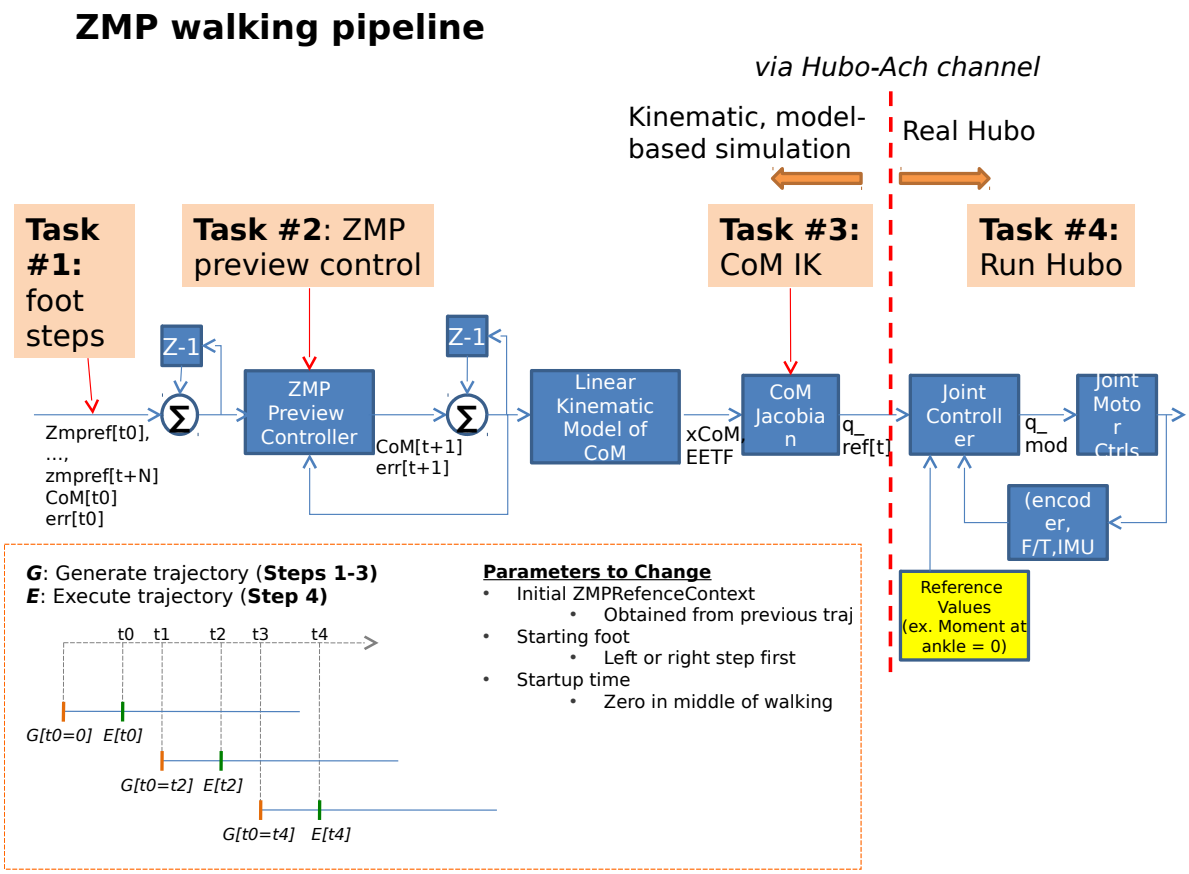
May 07

Status

- Made preliminary state machine for zmp-daemon and walker for continuous trajectory generation
- Created zmpdemo flowchart



- Edited walking-pipeline and added diagram of continuous walking receding horizon



Apr 29 - May 5

Status

- Generated evaluation times for the zmpdemo code for different types and number of footsteps.
- Gave final presentation with Xinyan on teleoperation.

Apr 22-28

Status

- Sungmoon and I then planned out a schedule of small tasks to complete achieve receding horizon online ZMP trajectory generation.
- Evaluate generation time of trajectories for different types and number of footsteps.
- Achieve receding horizon using a button on the keyboard to walk in a straight line.
- Implement receding horizon online ZMP trajectory generation using forward/backward and sidestep left/right using the keyboard.
- Nishiwaki, Koichi, et al. "Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp." Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. Vol. 3. IEEE, 2002.

- The next task after completing this will be to start incorporating the footstep planner and different walk types.

0.0.1 Future Milestones

- Incorporation of footstep planner and different walk types.

Apr 15-21

Status

- Set the motion computer up to have a static ip address.
- Read through the hubomz code and commented some of it. Primarily I commented the ZmpPreview.h file, which is the actual zmp controller. The CoM inverse kinematics was a little hard to understand, but I understand enough to move forward with the finite horizon.
- I read through the Nishiwaki (2002) paper which presents online generation of zmp walking trajectories. I modified Sungmoon's ZMP-Walking diagram and added the online receding-horizon method.
- Created presentation for TePRA with Rowland.

Apr 8-14

Status

- Juan and I repaired Hubo's right hand pinky and middle fingers. The pinky just need the set screw that hold the pulley on the motor shaft tightened. The middle finger was the one previously broken and we just re-glued it with krazy glue.
- I wrote a simple hand program to open and close the fingers using the keyboard. In doing so I discovered a bug and investigated the Hubo-ach code with Grey and came up with some possible culprits. Grey later fixed it. https://github.com/pvieira3/hubo-misc/blob/master/test-code/src/hand.cpp
- Set up everything for National Robotics Week and conducted the teleportation with the kids.

Apr 1-7

Status

- I helped Xinyan daemonize Liberty and cleaned up everything. https://github.com/pvieira3/Liberty-Daemon
- I updated the teleop-arms and teleop-legs code to accept arguments and make it easier to understand https://github.com/pvieira3/hubo-misc/blob/master/test-code/src/teleop-arms.cpp
- I made a separate Collision_Checker class. Currently, there's only a checker for the arms, but a legs one will be added in the future. https://github.com/pvieira3/hubo-misc/blob/master/test-code/src/Collision_Checker.cpp https://github.com/pvieira3/hubo-misc/blob/master/test-code/include/Collision_Checker.h
- I worked with Juan to get started tested grasping with the fingers. We discovered the control daemon had some bugs related to the fingers because they were never needed and therefore never tested and debugged. We notified Grey and he fixed the bugs.
- I thoroughly explained the impedance controller to Xinyan, which helped me understand some details I was unaware of.

- I updated the Fastrak class to accept the Fastrak or the Liberty tracker and renamed it to Teleop.cpp https://github.com/pvieira3/hubo-misc/blob/master/test-code/src/Teleop.cpp https://github.com/pvieira3/hubo-misc/blob/master/test-code/include/Teleop.h
- I fixed the compiler issues with Max's testIK.cpp code which he wrote to do a unit test on the FK/IK for Hubo.

Feb 18-24

Status

- Xinyan and I got the Polhemus Liberty tracker fully working and tested. We removed Fastrak completely and are now only using Liberty.
- I added a simple self-collision avoidance function based on an ellipse-shaped boundary. I added it to Hubo_Control.cpp and ellipse parameters to Hubo_Control.h in my forked repo on the collision-avoidance branch.
- I helped Andrew a bit to debug issues with hubo_state and with moving the hand to the location of the red object. He got that working! Videos are on his page.
- Helped get Max started on writing a UnitTest for the FK and IK.
- The working zmpdemo walking parameters for the video were:
  - ./zmpdemo ../myhubo.kinbody.xml -w canned -z 0.06 -y 0.0885 -d 0.01 -s 0.5 -A -c
  - 8 -p 2 -X 0.038 -h 0.5 -l 0.08
  - -w: walk type "canned", meaning straight line
  - -z: foot vertical lift-off height of 0.06m
  - -y: half distance between feet of 0.0885m
  - -d: double-support time of 0.01s
  - -s: single-support time of 0.5s
  - -c: max number of step = 8
  - -p: initial time spent with zmp stationary = 2s
  - -l: max length of footstep = 0.08m
  - -X: forward distance from ankle to zmp = 0.04m
  - -A: use ach
- Sungmoon and I tried tuning the walking parameters as well as tried adding gradually reduced step size steps at the end of the walk. This helped a little bit. Raising the COM z-position from 0.48m to 0.50m and changing the zmp distance in front of the ankle from 0.04m to 0.038m helped.
- Sungmoon and I wrote an impedance controller for the arm, using the torque sensors in the wrist.
- Code: https://github.com/pvieira3/hubo-misc/blob/master/test-code/src/impCtrl-arms.cpp
- Code: https://github.com/pvieira3/hubo-misc/blob/master/test-code/include/impedanceController.h

Mar 11-17

Status

- Rowland and I went through Matt Zucker's walking code and learned how he was spoofing input for the z-coordinate (height) of the feet in order to construct a zmp trajectory in the y-direction, which was being fed into his ZMP Preview controller, which spat out COM trajectory. This was then used with the inverse kinematics to obtain full-body joint trajectories.
- Rowland and I started adding code to get forward movement of the feet and body.
- Rowland and I finished adding forward walking to the code and kinematic visualization with Matt's help.
- I helped Saul understand and re-factor the zmpdemo.cpp code.
- I watched and learned from Matt about how we were going to break the code into multiple objects.
- I added the details for the runZMPPreview(), runCOMIK() and dumTraj() functions in our zmpWalkingGenerator.
- I helped out with the actual implementation of the code on Hubo and figuring out how to control Hubo's balancing such that walking would be possible.
- Code: https://github.com/golems/hubomz

Mar 4-10

Status

- Over the weekend Hubo's Right Hip Roll joint "broke" somehow. I took out the control board and discovered that ground trace from the power socket to one of the MOSFETs was burned out due to over-current. We swapped power, encoder and limit switch cables with the Right Hip Yaw and turned on motor control with Hubo-i and it worked fine. Therefore, there should be no issues mechanically, and just a replacement of the board should fix the problem. The cause of the failure is still not clear. We are aware that Hubo was designed with pretty much no safety factors, so if somehow he was being commanded to stand, etc. for too long without cooling down then an over-current may have occurred naturally. This just means we need to be more conscious of Hubo's limits and how we test him.
- Pushkar helped me install Dart/Grip again from source with a different install method for fcl. This got most of the tabs and worlds working, but not all of them. Still can't install from the apt repo for some reason.
- Worked on porting our Inverse Kinematics code over to Ivan's WalkingTab.
- I helped XinYan test his Polhemus Liberty daemon by using it to send Liberty sensor data from XinYan's computer to Hubo using achd and controlling Hubo's right arm via the sensor. It worked, but seemed a bit different than when using Fastrak. Need to do more testing, and test with the left leg to figure out why it's different.

Feb 25 - Mar 3

Status

- Ana and I added a function to the Hubo_Tech library that calculates the center of mass w.r.t. the neck, which is the world frame for the FK and IK. Ana and I starting writing a balance daemon that uses the center of mass model and the force/torque sensors in the ankles.

- The Ethernet cable connected to PC2 was tripped over and the plastic socket was snapped off the circuit board on Hubo. I got a new socket from James Steinberg in Van Leer, desoldered the old ethernet socket on the Smart Power board and soldered on a new one.
- I spent a day working with Ivan to figure out how to control joints easily in Dart.
- I finally installed Dart/Grip after doing a clean install of Ubuntu 12.04 but the worlds still wouldn't load.

Reflection

- For walking a clear state machine idea has been formed, but we just need to implement it.
- For Dart/Grip I will talk to Pushkar, etc. on Monday to resolve the issues.

Feb 18-24

Status

- Andrew and I tested running a program on PC1 and sending the commands over Ach to PC2 and while the daemons were running on PC2, but we are having issues. We are doing something wrong with the communication.
- I got the Polhemus Liberty working with PiTerm. I then wrote a simpler version that got rid of all the GTK GUI stuff. I still need to parse the string of data into an array and daemonize it.
- XinYan and I tested the leg FK/IK and discovered that the problem was Grey's ordering of the leg joints in control-daemon.h was different than that of Rowland and me. The Hip Yaw and Hip Pitch were switched, so Grey changed them to match our FK/IK.
- I also discovered that because we had that miscommunication about the joint ordering, the limits for the Hip Yaw and Pitch in our IK solution were swapped!
- XinYan and I got Fastrak working with the legs. However, we discovered that the leg IK doesn't handle the situation when the goal position is outside the workspace properly, and returns -nan as the goal joint angle for the Hip Pitch, Knee and Ankle Pitch. We are working on fixing this.
- Andrew and I were able to read the state channel from the Vision PC, but not write to the Ref channel.
- I fixed the Leg IK code. It was returning -nan for joints 3, 4 and 5 (RHP, RKN and RAP) when the goal was outside the workspace. I changed the square roots in the calculations of the joint angles to be csqrt (complex sqrt) and changed the radical to type 'double complex'. The norm between the goal position and the solution positions wasn't being calculated with the correct goal position because it was changed at the top of the function. The ordering of the limits for the leg joints was wrong as well, because the hip pitch and hip yaw values were switched.
- I wrote a program to control the position and orientation of the feet using two Fastrak sensors! It works really well, and has no issues when you move the sensors outside the workspace of the feet.
- Ana and I integrated her dart_Lite library into the hubo-motion-rt library and verified that it loads the URDF file in successfully.
- I helped get XinYan set up with Liberty and he got it working on his own laptop with Ubuntu via USB. He will continue to finish the writing of the read function and daemon.
- I took the neck off Hubo and tightened the loose screws so that if a camera is temporarily mounted on the neck, any unnecessary looseness wouldn't be a problem.

- Hubo 4-Limb Teleoperation: Hubo's four limbs are being controlled using 4 different motion tracking sensors from the Polhemus Fastrak device. I have a sensor taped to each foot and one in each hand. We are using the relative translation of the sensors and the absolute orientations relative to the Fastrak base station. These transformations are mapped to Hubo's hands and feet using the inverse and forward kinematics of the arms and legs. Using an instantaneous transformation to, for example, the Fastrak sensor in my left hand, we input that transformation, or pose, into the inverse kinematics for Hubo's left arm to determine the best joint angles solutions to produce that position and orientation of the hand. The inverse kinematics generates eight possible solutions, of which only a few could be possible, given joint limits and and workspace considerations. If no solution contains six joint angles that are within the joint limits, then the solution that gives a position closest to the goal position is found by taking the norm of the vector between the goal position and the position given a solution of joints that are scaled to comply with the limits.
- Code for teleop of the arms: https://github.com/pvieira3/hubo-misc/blob/master/test-code/src/fastrak-arms.cpp
- Code for teleop of the legs: https://github.com/pvieira3/hubo-misc/blob/master/test-code/src/fastrak-legs.cpp

Feb 11-17

Status

- Writing code to test the leg IK and FK. Will use it to first squat, and then balance on one leg.

Feb 04-10

Status

- Read papers on ZMP and Preview Control
- Gave a presentation to the class with Eric Huang and Grey about locomotion, which included static and dynamic walking, ZMP and Preview Control.
- Continued updating the code for standing on one foot.
- Fixed a bunch of mistakes in the Hubo Kinematics Tech Report and added to the abstract.
- Helped Grey run my squatting code to demo for ABC
- Worked with Clayton to create a better side-by-side video of Hubo cutting a rectangle in cardboard using Adobe Premiere.

Reflection

- Need to make sure we export from Adobe Premiere as mp4 format.
- Creating both 4:3 and 16:9 aspect ratio videos is useful for presentations and watching videos, respectively.

Jan 28 - Feb 03

Status

- Added comments to the Hubo_Tech header using Doxygen commenting style.
- Started writing program to have Hubo stand on one leg using center of mass for the feedback term

Reflection

- Need to generalize code more as opposed to hard coding actions.

Jan 21-27

Status

- Rowland, Grey and I successfully cut out a 15x10cm square in cardboard using Fastrak motion tracking for the right hand and for squatting.
- Rowland and I finished writing the TePRA paper and submitted it.
- Rowland and I also fixed the IK and FK code so that the teleoperation in workspace works for 3-DOF.
- Ana and I tested a simple version of center of mass calculation for the right leg. When we moved the leg, the location of the center of mass moved in the correct directions, but not with correct magnitudes.
- Hubo Cutting Rectangle in Cardboard: 01/25/2013. This is a video of Hubo cutting a 15x10cm rectangle in cardboard, with two different camera angles, a main closeup view of the cut, and a second, smaller view in the top-right corner of the board, Hubo and the teleoperator in the background. We balanced him using his ankle motors, which are velocity controlled using just a proportional term. They are compliant with the moments about his ankles, but resistant against the angle of his torso with respect to the vertical in order to keep him upright. He has a Dremel strapped to his hand with a drill bit insert. We use the inverse kinematics of the arm and a Polhemus Fastrak Motion Tracking device to move his arm horizontally. By lowering the tracking device, Hubo also lowers by bending at the knees. These two degrees of freedom allow Hubo to cut out the rectangle.

Reflection

- The modeling of the center of mass is going a little slow, because the URDF file is hard to interpret and doesn't seem accurate.

Jan 14-20

Status

- I was able to get Hubo to squat on Monday with just a bit of sway. The sway is due to the fact that the ankles are not matching the speed of the hips, which causes the torso to lean forward or backwards, so when Hubo stops moving he has to rotate his ankles to return his torso to the vertical orientation. This can be solved possibly by adjust the gains on the ankle pitch velocity.
- Grey and I performed cardboard cutting with the Fastrak and the dremel. We used the Drill IK that Rowland wrote and just varied the y-position in order to create a smooth cut along a straight line. We were able to get Hubo to cut an 8-inch line in the cardboard, but then the wrist broke loose due to experience a torque greater than its maximum allowable.
- We also performed teleoperation of the dremel arm while using another sensor to control squating. We achieved this by using the dimensions of Hubo and some simple trigonometry.
- I helped Andrew to get a webcam tracking a red piece of paper. Then, as the paper moved across the webcam's view, Hubo would move his arm in the same directions along the y-z plane. Andrew got this working using OpenCV and Boost for the vision and object tracking, and sockets in order to send the relative motions to Hubo's Linux computer. We need to adjust some of the transformations since Hubo's arm was not responding exactly as we expected.

- I wrote a program to control both of Hubo's arms with the Fastrak sensors, as well as his waist with a third sensor. It is slightly different than the original one that Grey wrote, which didn't perform as hoped, possible due to the IK solutions and possibly due to there only being six DOF controlling six DOF, coupled with Hubo's limited worskpace. I will test my version on Monday, January 21st.
- Hubo Squatting: 01/18/2013. This is a video of Hubo balancing and squatting. His ankles are velocity controlled using just a proportional term. They are compliant with the moments about his ankles, but resistant against the angle of his torso with respect to the vertical in order to stay upright. He squats by adjusting the velocity of his knee joints and then setting the hip and ankle joints to half the knee velocity.
- Hubo Balancing & Teleop (LEFT): 01/18/2013. This is a video of Hubo being controlled via teleoperation to balancing, squat and move his right arm. He balances using his ankles, which are velocity controlled using just a proportional term. They are compliant with the moments about his ankles, but resistant against the angle of his torso with respect to the vertical in order to stay upright. We use the inverse kinematics of the arm and move it horizontally using a Polhemus Fastrak Motion Tracking device. He squats by adjusting the velocity of his knee joints and then setting the hip and ankle joints to half the knee velocity.
- Hubo Wall Cut (RIGHT): 01/18/2013. This is a video of Hubo cutting a line in cardboard. He balances using his ankles, which are velocity controlled using just a proportional term. They are compliant with the moments about his ankles, but resistant against the angle of his torso with respect to the vertical in order to stay upright. He has a dremel with a drill bit strapped to his hand. We use the inverse kinematics of the arm and move it horizontally using a Polhemus Fastrak Motion Tracking device.

Reflection

- The balancing and squating work fine, but for anything more, we need the calculation of the center of mass. This way we won't have to rely on the IMU and we can just give a point of the floor for where to put Hubo's center of mass. This should make it possible to walk in a logical manner.
- The webcam object tracking control of Hubo's arm exhibited some good response, but I definitely mixed up the IK for the drill and the hand and thus the transformations are wrong. This is an easy fix though.

Milestones

- Get data for the wall cutting. We will get Fastrak data and Forward Kinematics data, and plot the two against each other.
- Pick an object up with two hands using Fastrak.
- Get the calculation of the center of mass and verify that it's correct

Jan 07-13

Status

- Finished writing the rough draft of the TePRA 2013 paper "Humanoid Robot Teleoperation for Dual-Arm Tasks with Power Tools" with Rowland O'Flaherty. This paper presents the implementation of inverse kinematics to achieve teleoperation of a physical Hubo+ robot platform. Using a closed-form kinematic solution and a basic feedback controller, teleoperation of simple tasks is realized. The Hubo+ platform will be used to compete in the DARPA Robot Challenge, which requires autonomous execution of various tasks; such as, cutting through walls and moving rumble. Joint limits and singularities are accounted for using the different cases in the kinematic solution; and a decision method is implemented to determine how to position the end-effector when the goal is outside the feasible workspace.

- Achieved balancing with Grey. We did this by using velocity control for the pitch and roll joints of the left and right ankles. We designed the controller so that the ankles are compliant with the moments about the ankle joints, but resistant against the waist IMU angle. These are both proportional terms. We originally tried using position control with P, PD and PID type controllers but found that oscillation would occur. The following is the code we used.

Reflection

- We had some trouble getting Hubo to bend at the knees and hips in order to lower, but Grey had an idea to set the ankle velocity equal to the hip velocity.

Milestones

- Get Hubo to act like a spring, where if you press down on his shoulders he'll lower, keeping his back vertical, and if you let off he'll rise back up.
- Cut through cardboard with the dremel by using the IK for the arm and bending his legs in order to increase range of drilling motion.
- Update the TePRA paper with results from dremel experimentation and hopefully moving "rubble."