TRIBAL KNOWLEDGE

DASL + HUBO LAB

Alex Alspach • Roy Gross


Hubo
KAIST
Welcome to DASL

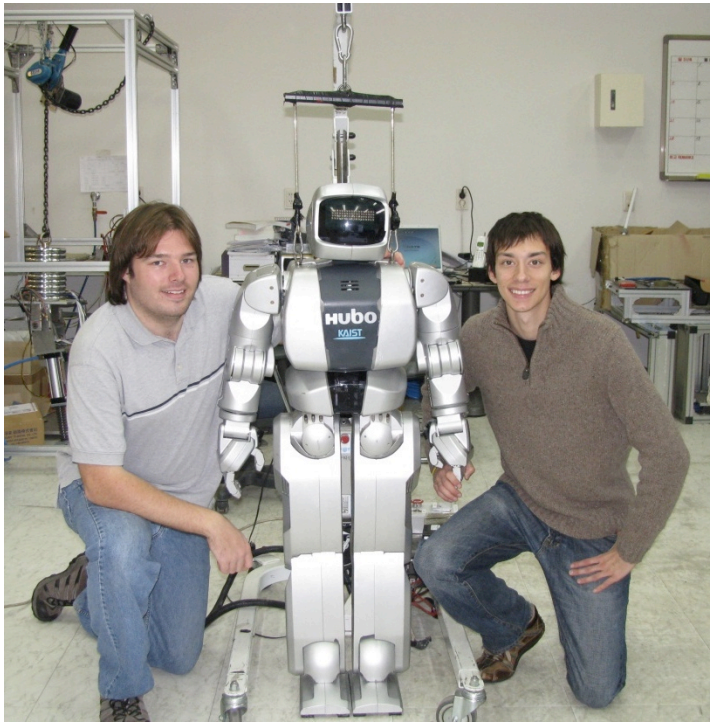
Topics of Discussion

- Co-Op at HUBO Lab 2008-2009
- 3D Model Generation
- HUBO Model Verification
- HUBO Assembly Manual
- Simulator Evaluation
- Other Work and Play

DASL Embedded

- September 2008-2009, the first two students are embedded into HUBO Lab in Daejeon, South Korea.
  - Roy Gross (MEM)
  - Bryan Kobe (ECE)

- Mission:
  - Establish a base for knowledge transfer between HUBO Lab and U.S. PIRE universities
  - Develop strong relationships between Korean and U.S. Partners
  - Establish external business relations
  - Enjoy a new culture



Knowledge Transfer

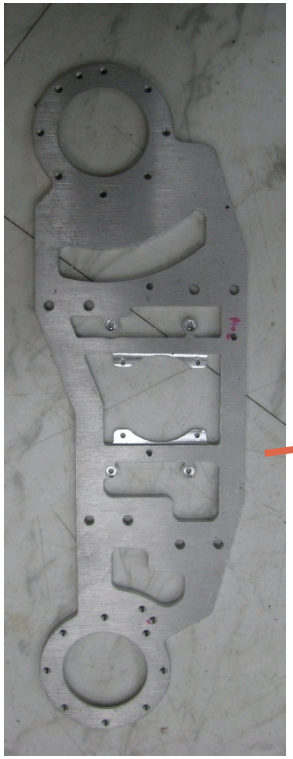
Original HUBO2 KHR-4 was designed in AutoCAD as 2D orthographic drawings

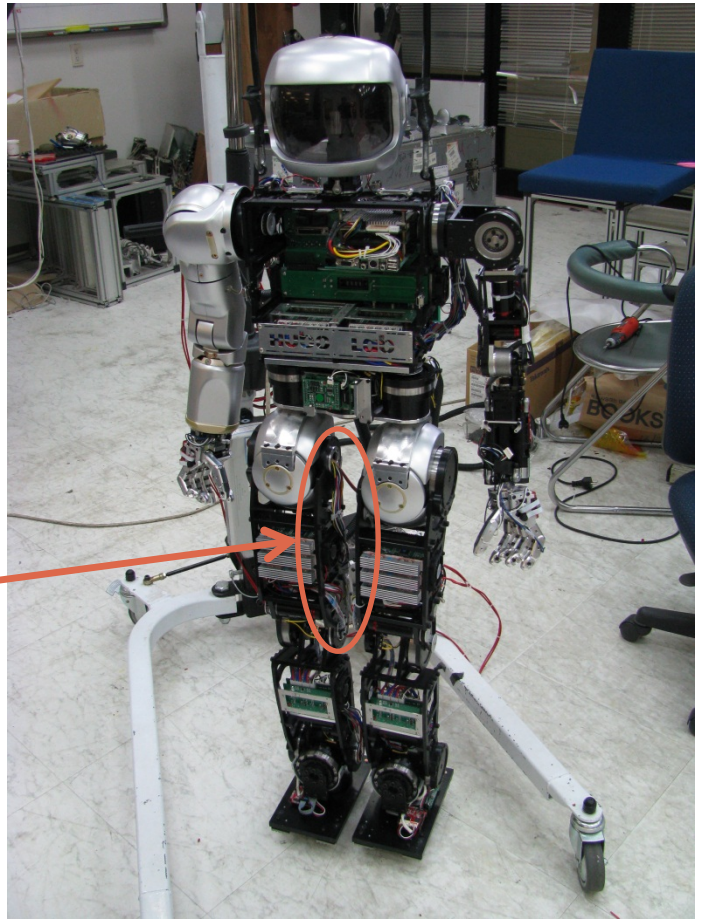
- Goal 1: Transfer data to versatile 3D solid model
- Goal 2: Learn and document manufacturing processes
- Goal 3: Generate approximate physical model (mass, inertial prop.)
- Goal 4: Lay ground-work for HUBO manual

CAD Transfer

HUBO upper-inside leg AutoCAD drawing

Identify actual part

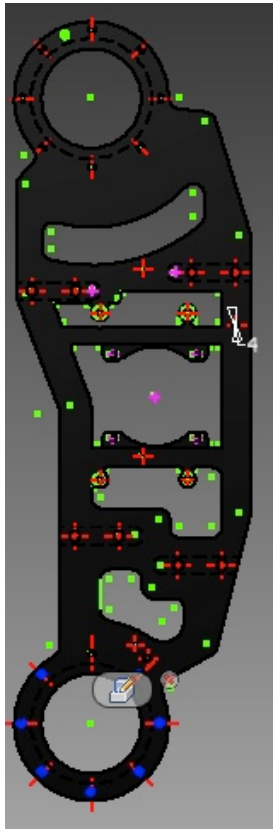


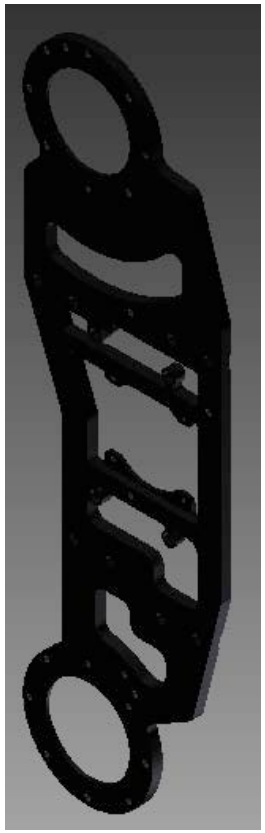


Identify part on assembly

CAD Transfer

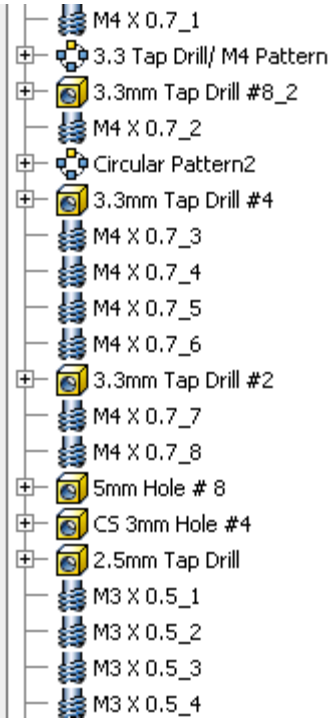
Transfer AutoCAD to Autodesk Inventor

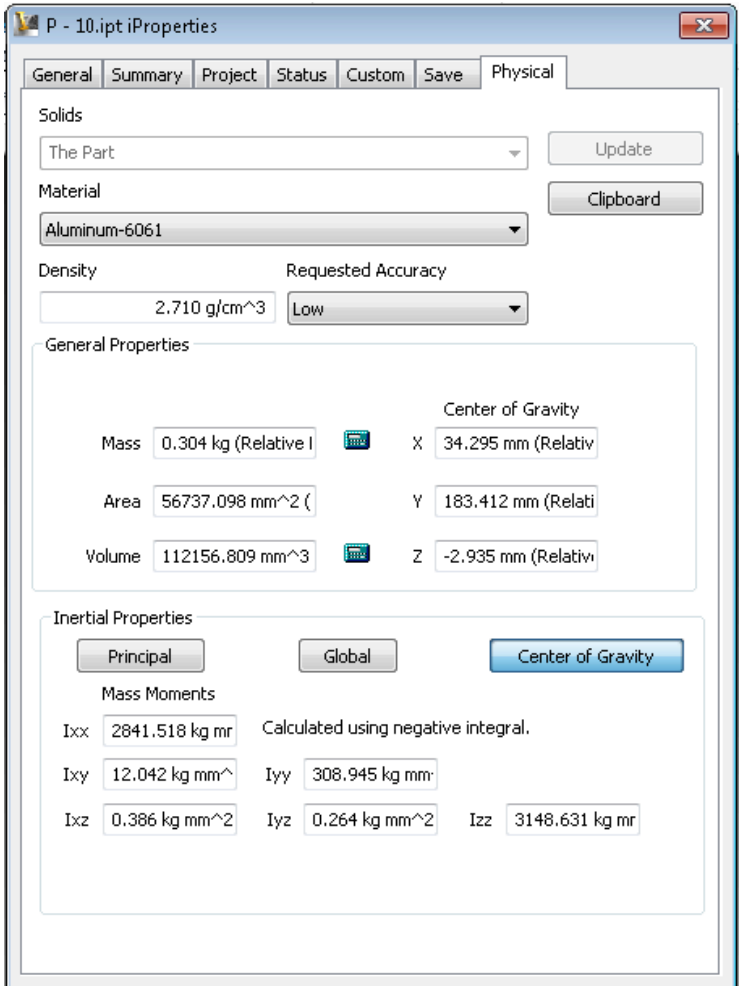




Finished 3D part

Manufacturing Notes

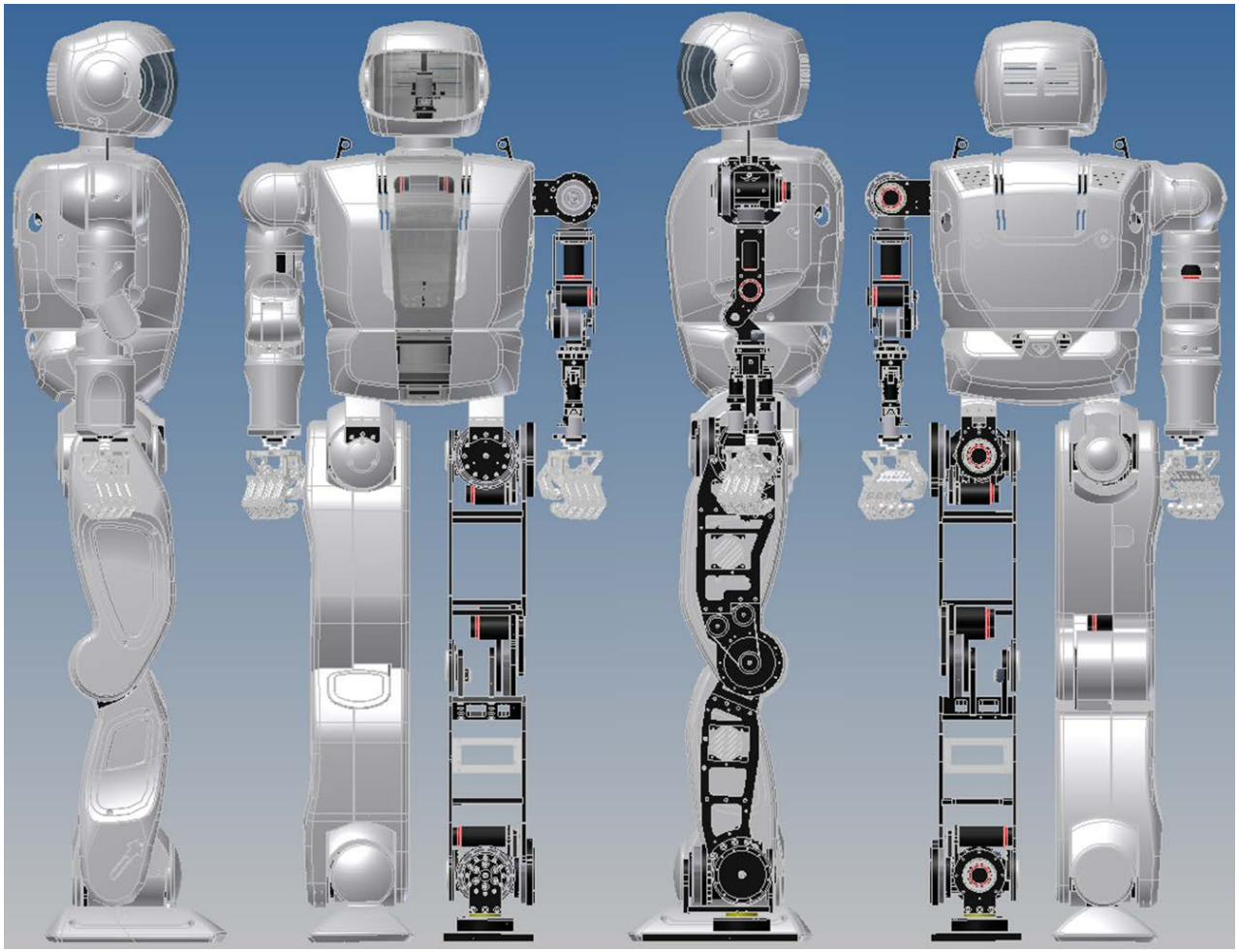




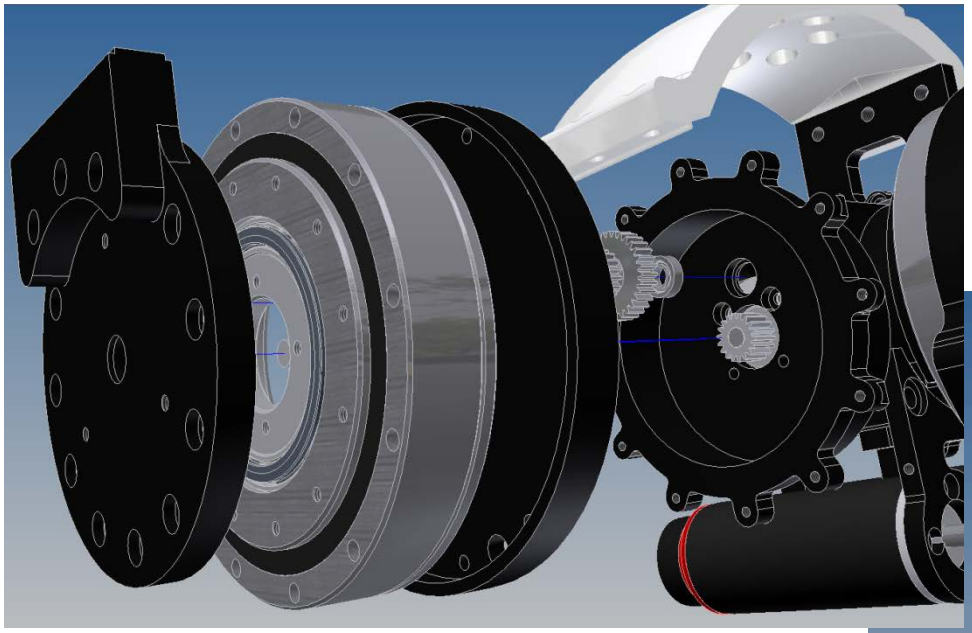
Physical Properties

CAD Transfer

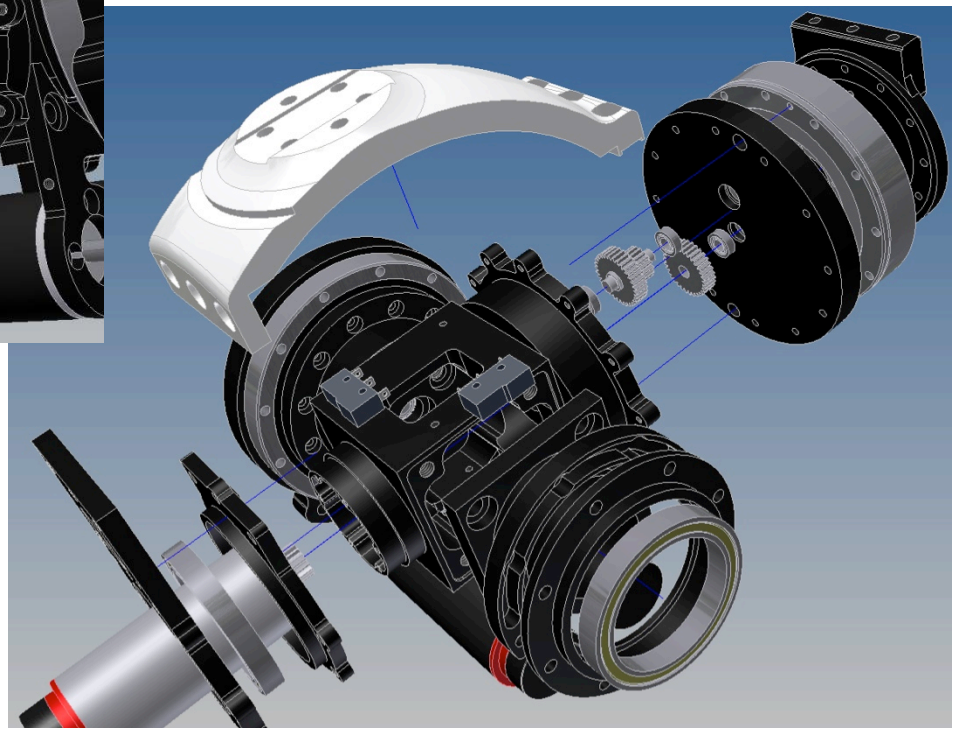
Repeat 200 Times…



CAD Transfer

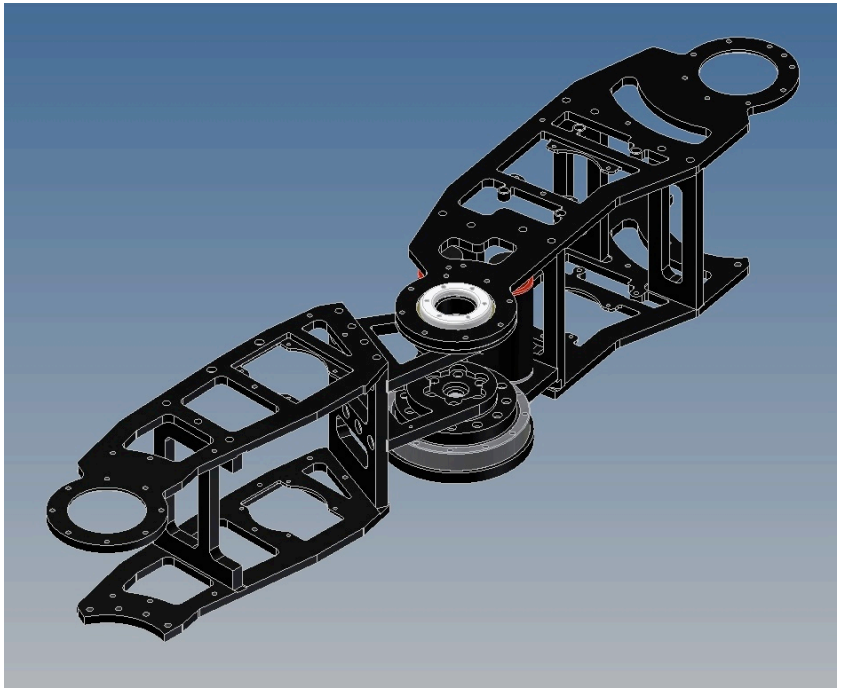


Internal components & Exploded views



Model Verification

- Replicate HUBO Lab quality manufactured parts using 3D model
  - Maintenance
  - Catastrophic repair
  - Development of new mechanical systems



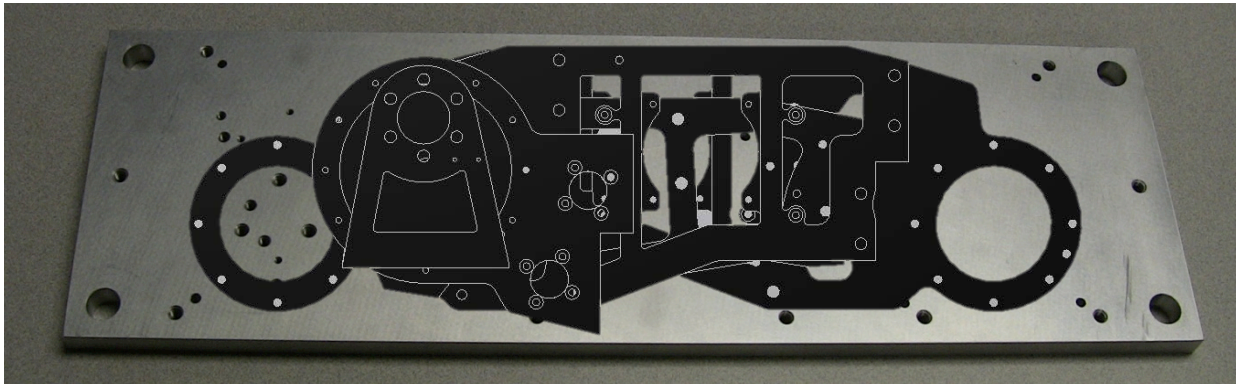
HUBO's right leg

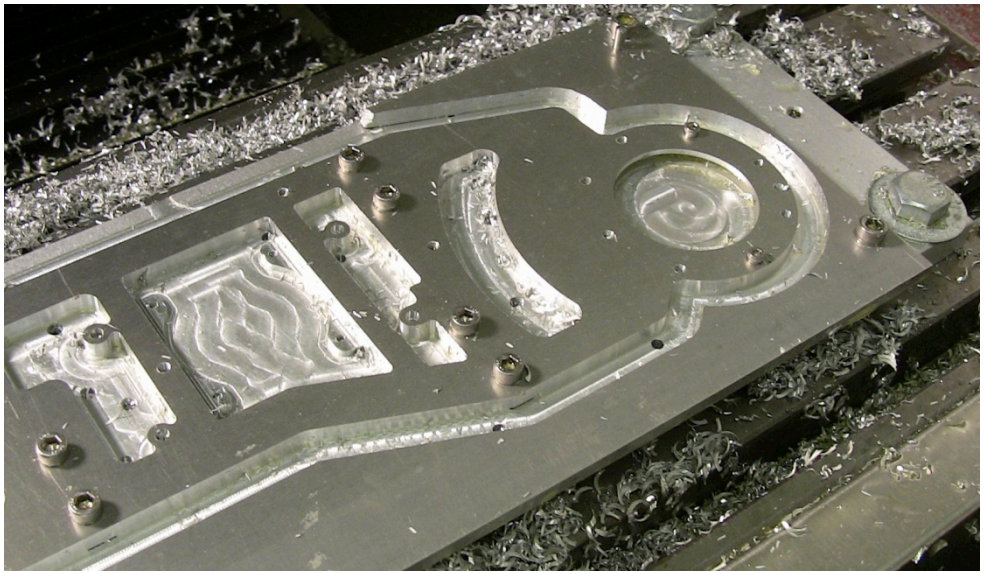
Model Verification

HUBO Lab CNC

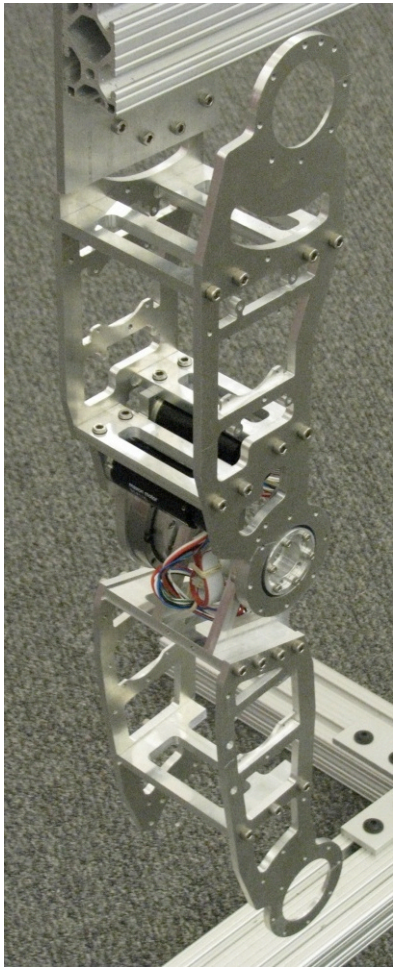
DASL CNC

Model Verification

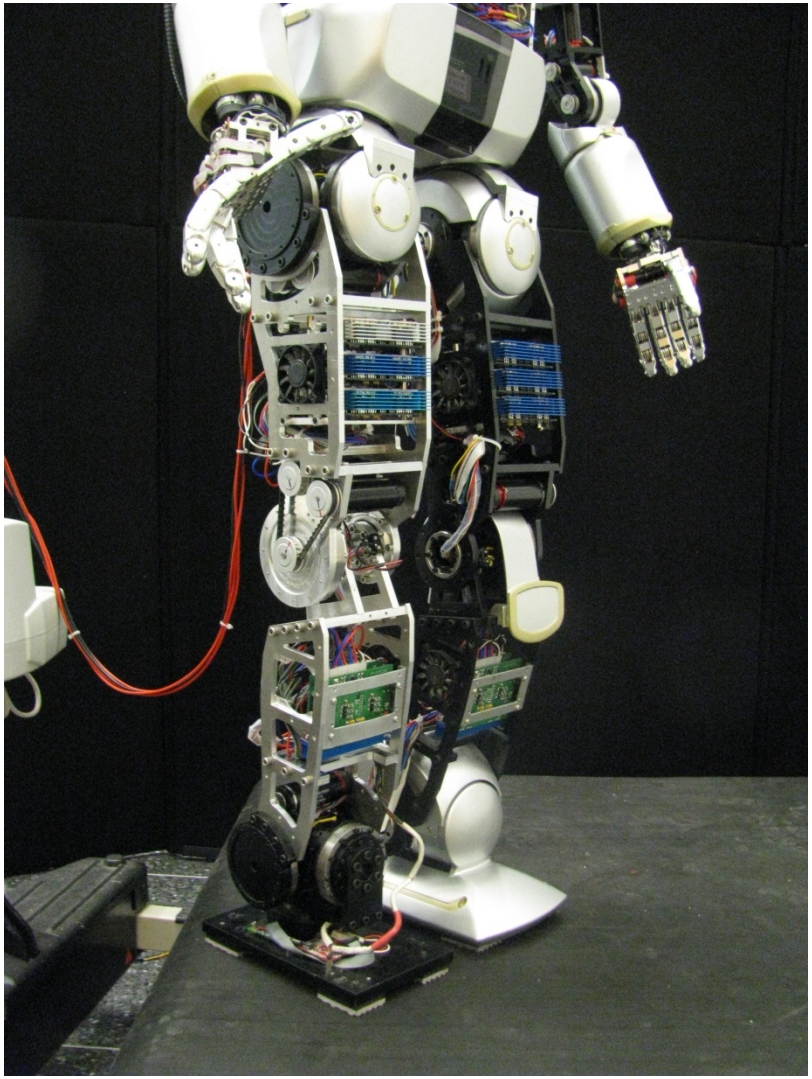




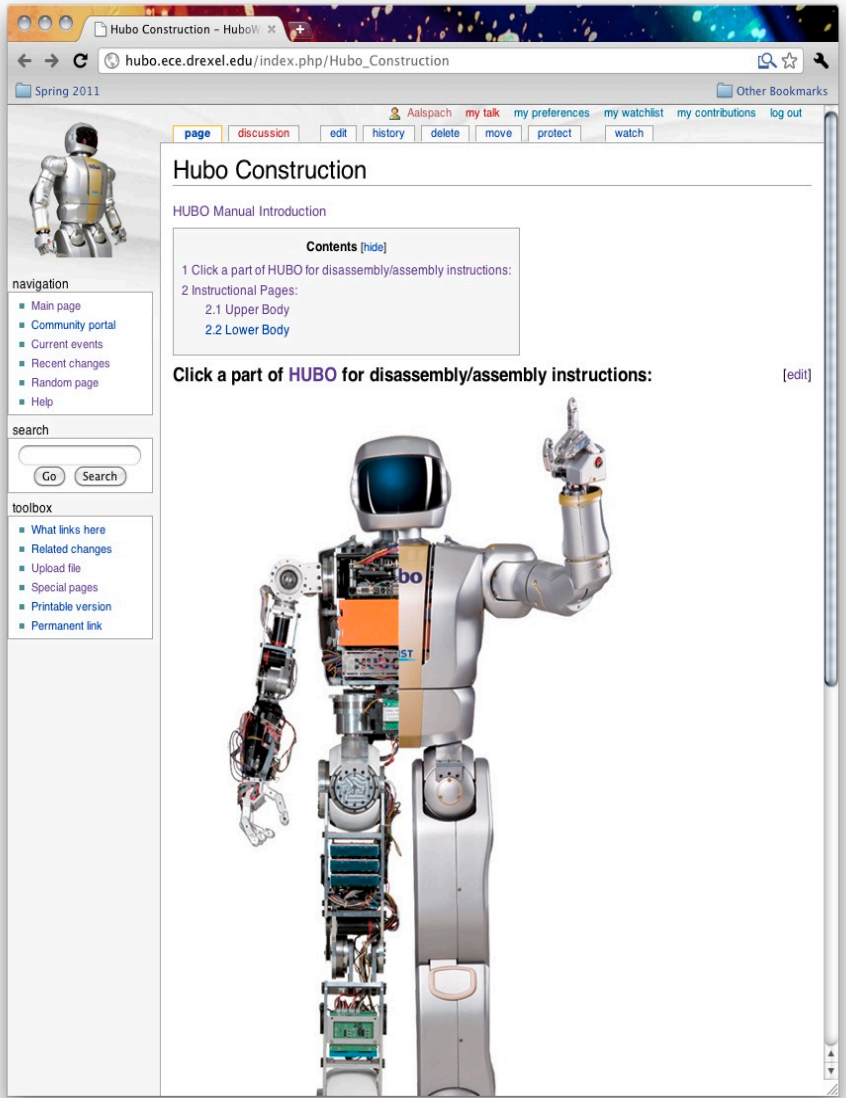
Model Verification

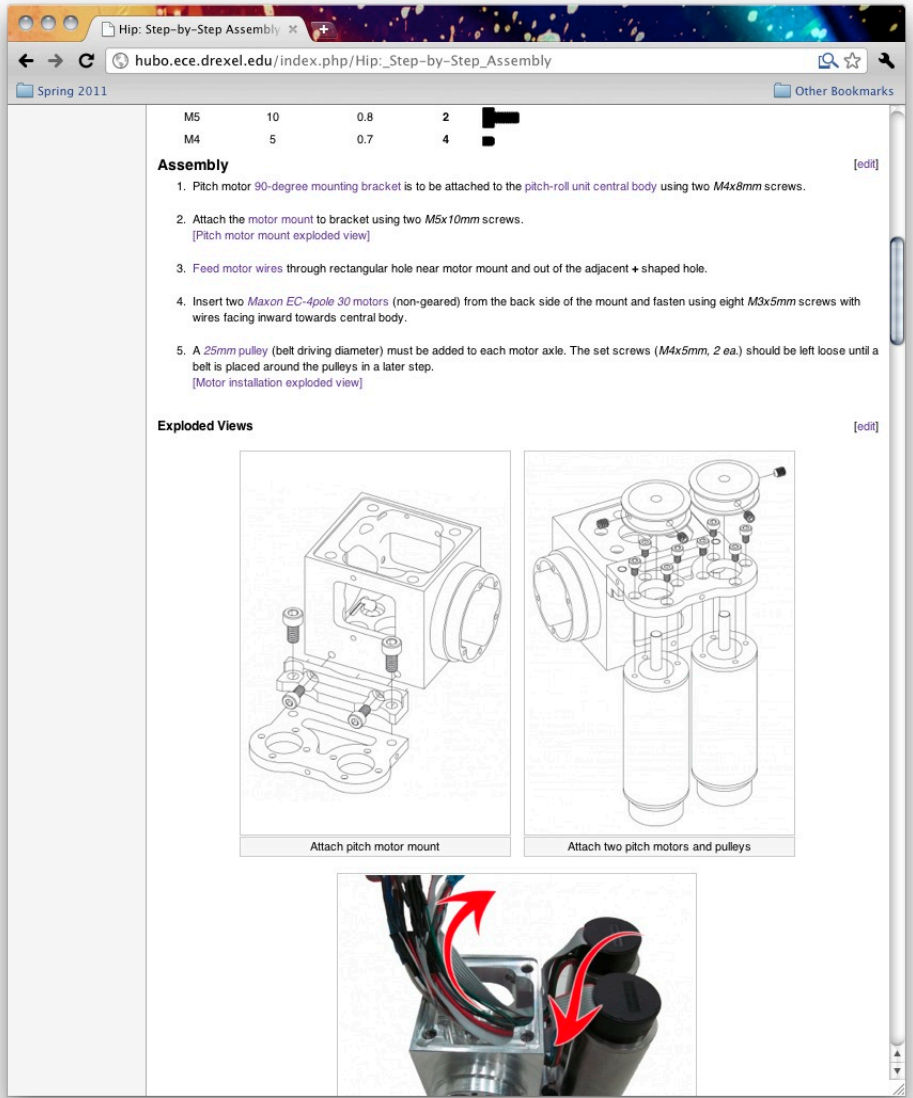


DASL-Machined Leg Replica

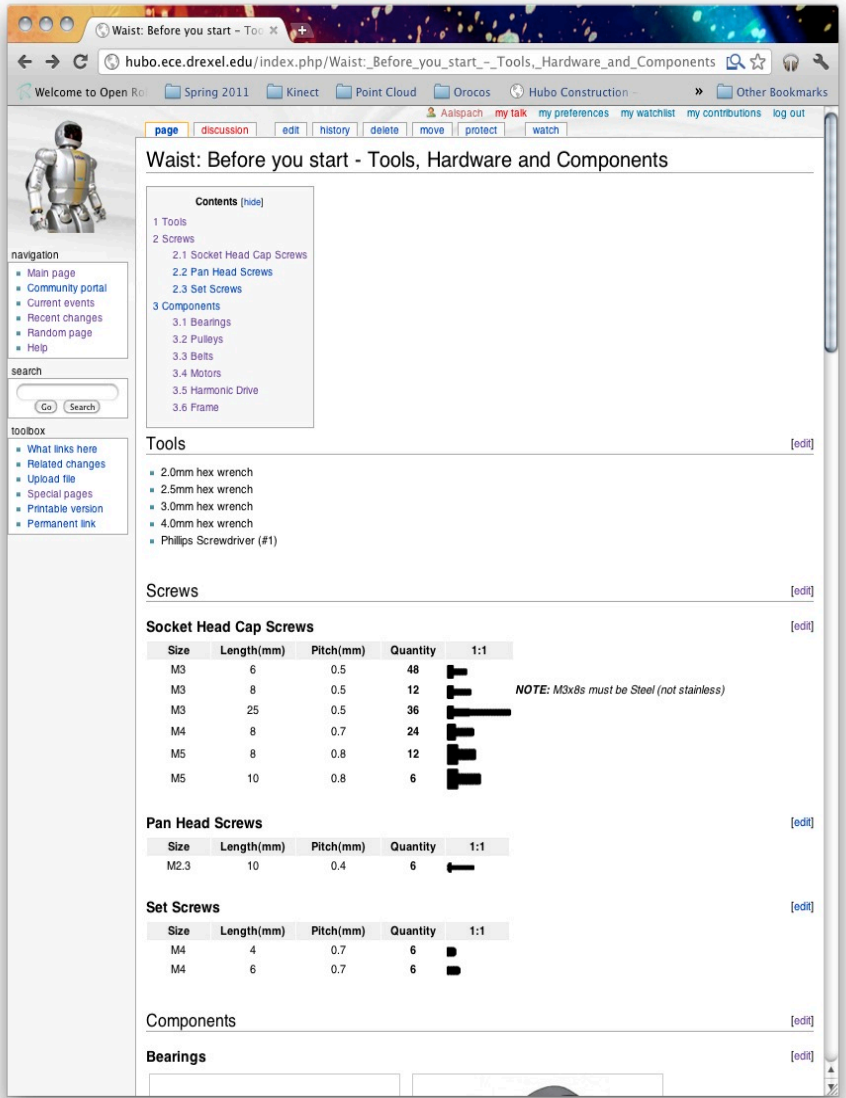


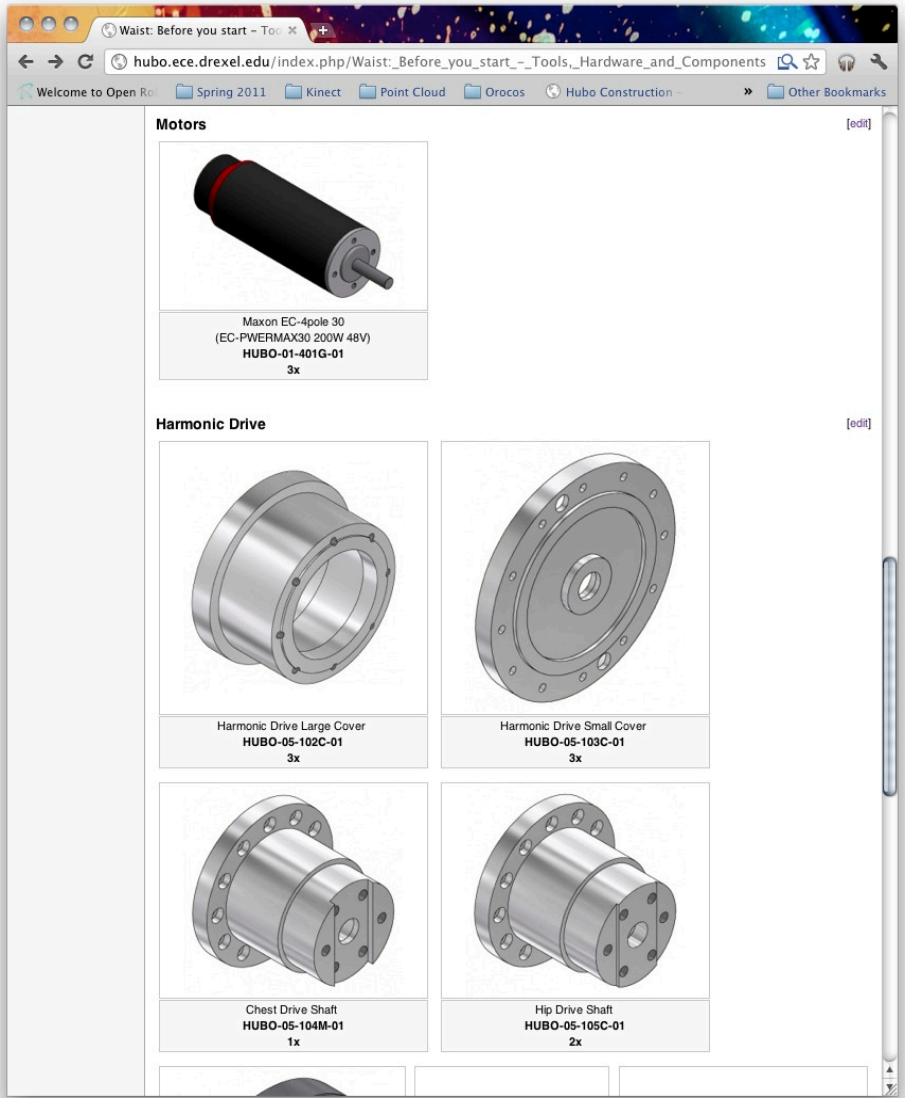
HUBO Assembly Manual



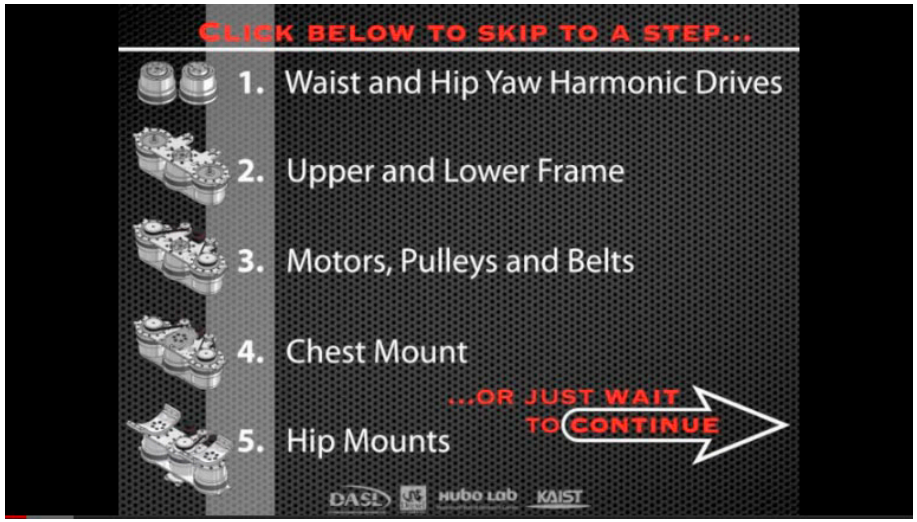


HUBO Assembly Manual

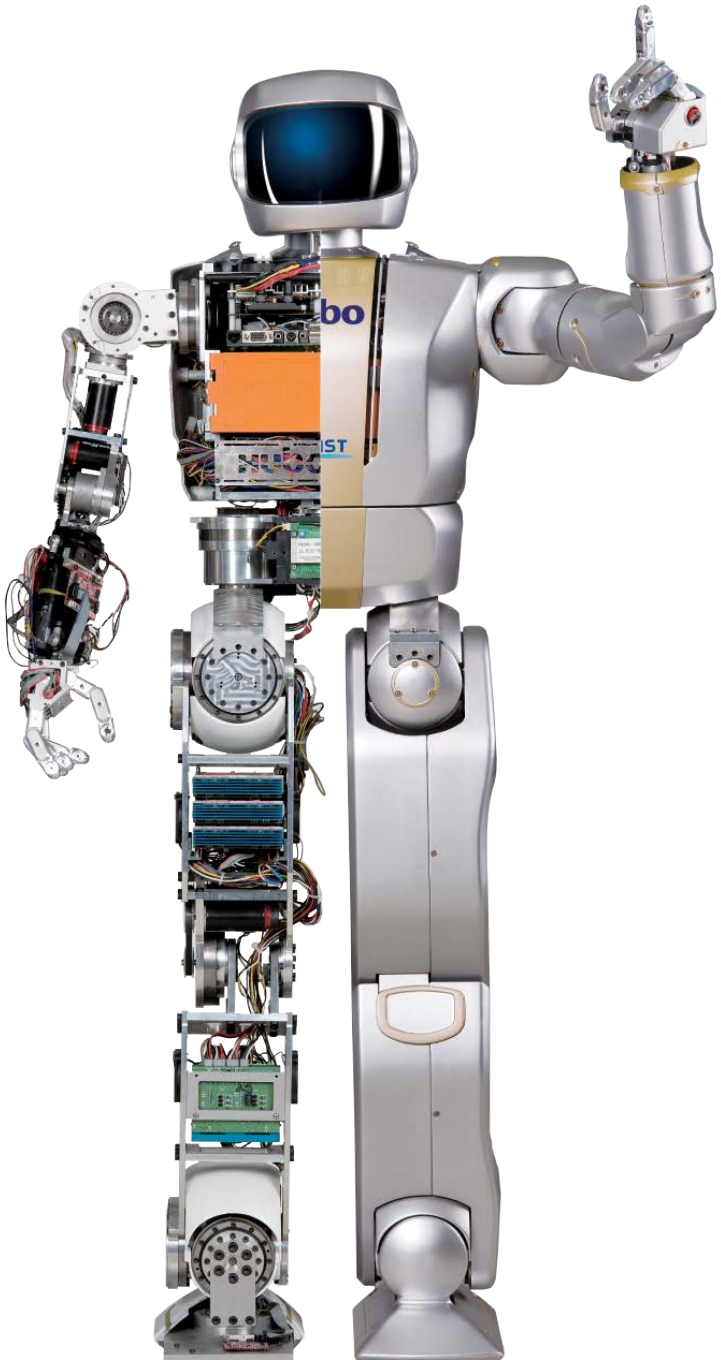




HUBO Assembly Manual - Videos



Manual Walkthrough

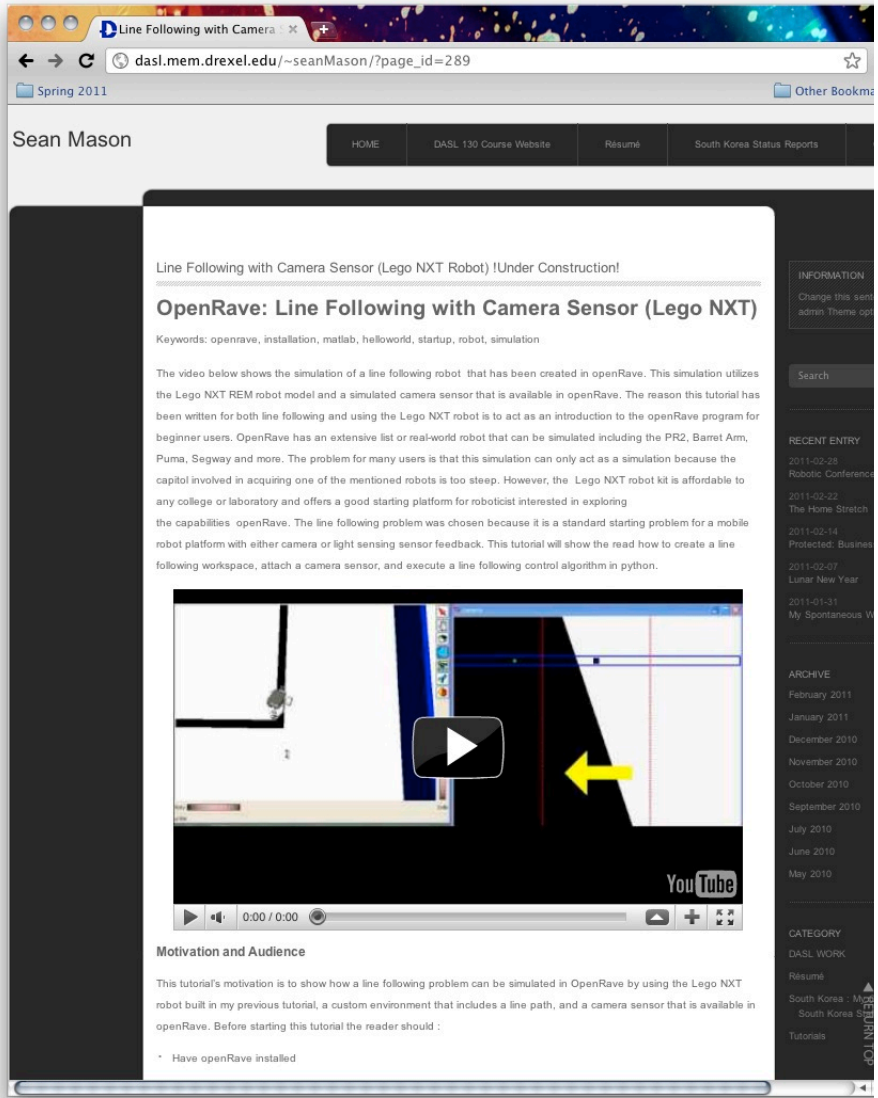


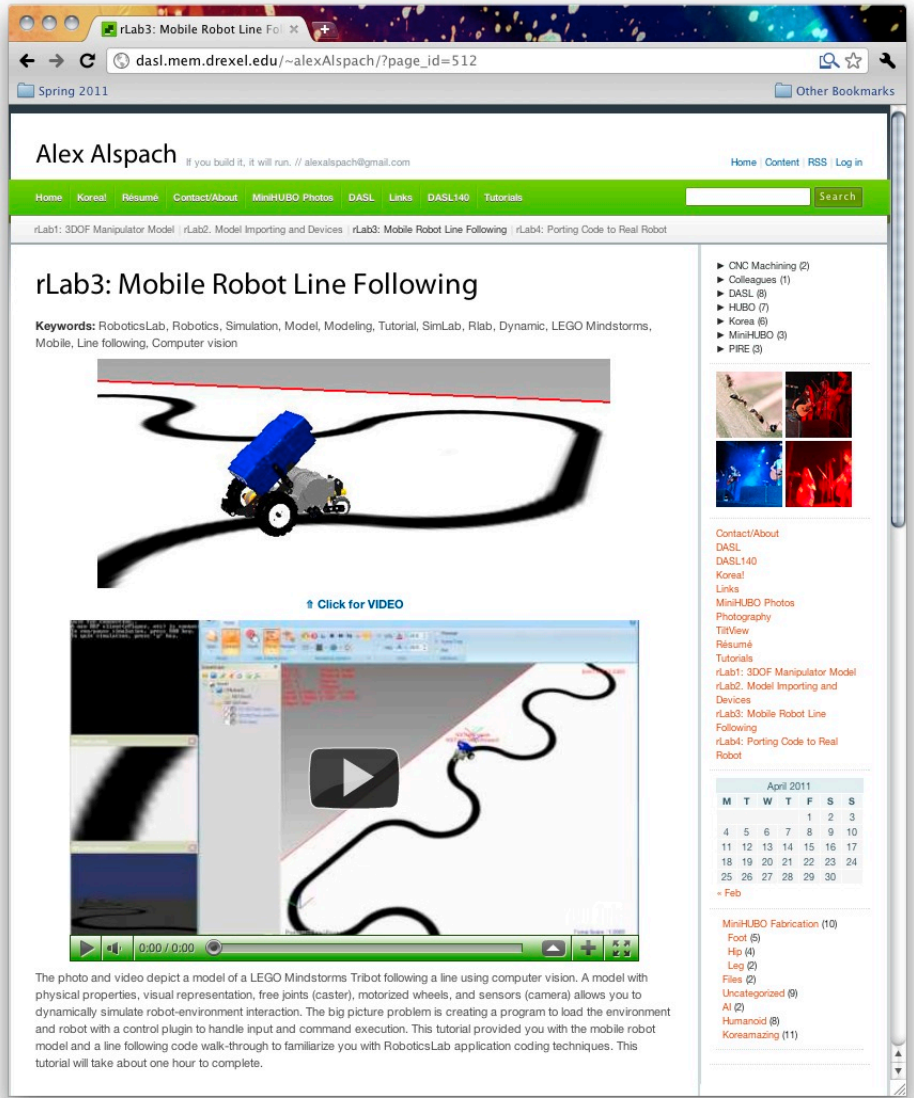
HUBO Assembly Manual Future

- User's Manual
  - Setup
  - Operation (virtual & online)
- Electronics Diagrams
- Code Documentation
- Common Issues and Solutions
- Question & Answer Forum
- Trouble Shooting Guides
- Downloads
  - Simulation Models and Physical Data
  - Controller Code for MiniHUBO, Virtual HUBO and HUBO

Simulations and Tutorials

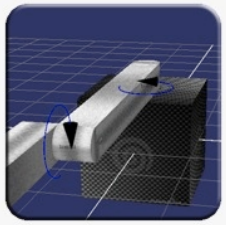
OpenRAVE RoboticsLab





Simulations and Tutorials

RoboticsLab



Tutorial 1:

3DOF Manipulator Model

November 3, 2010

Tutorial 2:

Importing Geometry and Devices

December 6, 2010

Tutorial 3:

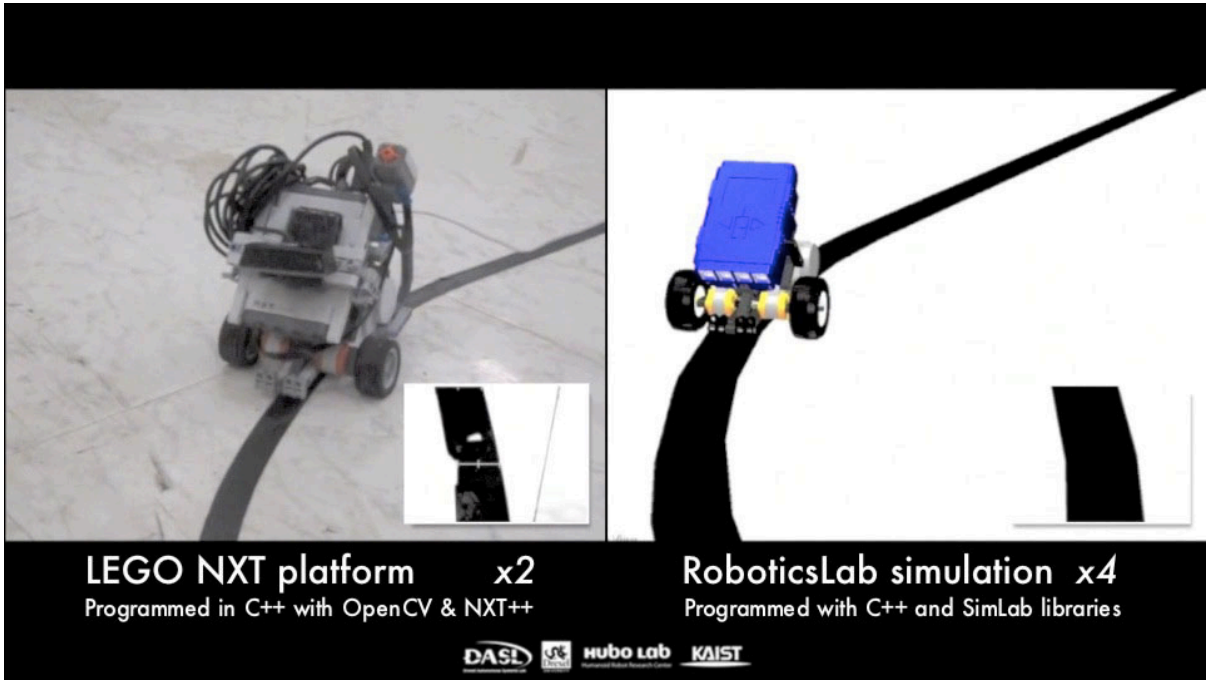
Mobile Robot Line Following

November 24, 2010

Tutorial 4:

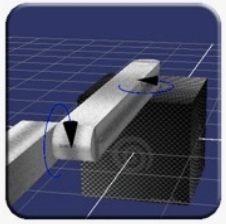
Porting Code to Real Robot

January 27, 2011



Simulations and Tutorials

RoboticsLab



Tutorial 1:

3DOF Manipulator Model

November 3, 2010

Tutorial 2:

Importing Geometry and Devices

December 6, 2010

Tutorial 3:

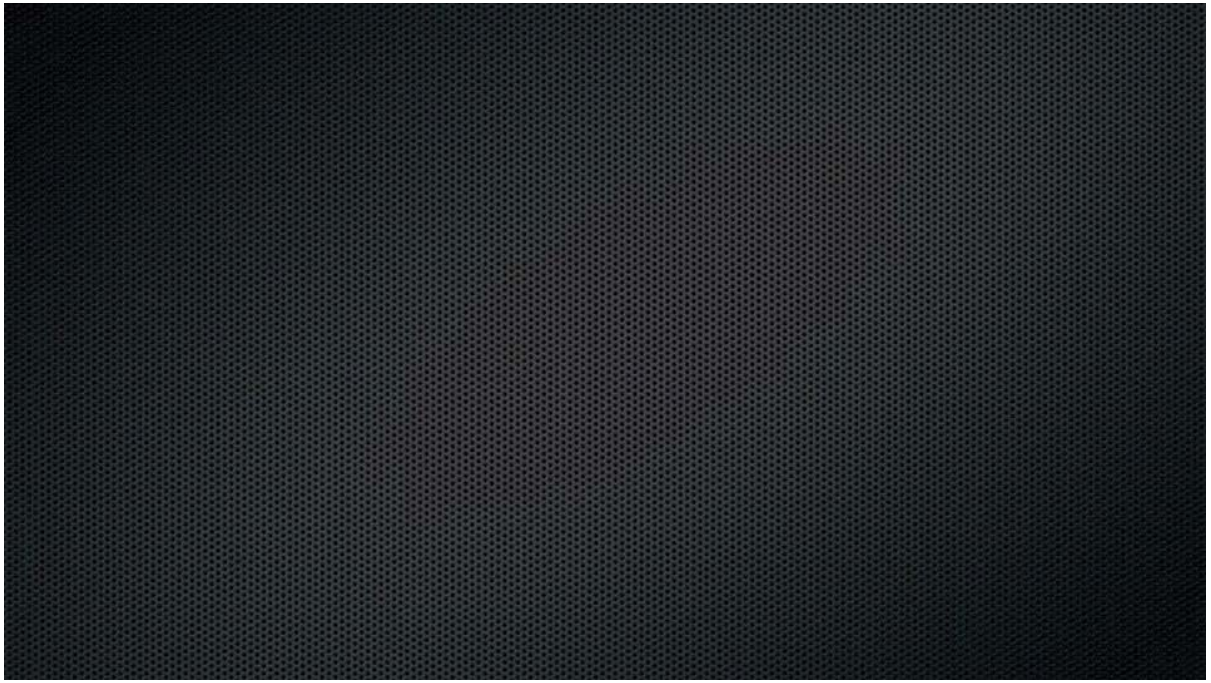
Mobile Robot Line Following

November 24, 2010

Tutorial 4:

Porting Code to Real Robot

January 27, 2011



Simulations and Tutorials

RoboticsLab

rLab2. Model Importing and

dasl.mem.drexel.edu/~alexAlspach/?page_id=610

Welcome to Open Ro | Spring 2011 | Kinect | Point Cloud | Orocos | Other Bookmarks

Requirements

- RoboticsLab and Visual Studio 2008 [if not yet installed, please refer to Tutorial 1]
- Mobile robot VRML files [Download TribotVRMLfiles.zip]
- Mobile robot model [Download NXTrem.aml]

Modeling Mobile Robot with Wheel Actuation and Vision Sensors

This section gives information on modeling a robot in rBuilder with actuated joints and vision sensors. This information can be applied generally for other forms of actuation on sensing.

Step 1: Creating Block Diagram of Robot Model.

⇑ Click to Expand

Open rBuilder.

As described in Tutorial 1, create a block digram of the LEGO Tribot to be modeled. This diagram will include a body, two wheels with revolute joints, a caster frame with a revolute joint, and a caster wheel with a revolute joint (*figure 1*).

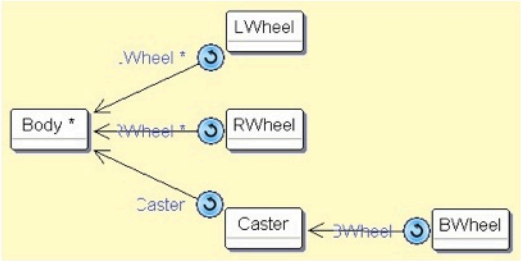


Figure 1: Block diagram of differential drive LEGO Tribot model.

Each joint is a revolute joint. Make sure to set the friction of each of these to *zero*. All bodies should have *zero* friction except the three wheels which should have a friction of *one*.

Physical parameters like COM and moment of inertia can be obtained from your preferred 3D modeling software. The properties used in the provided model were obtained from a similar example differential drive robot provided by RoboticsLab.

Step 2: Importing Render Geometry

⇓ Click to Expand

Step 3: Addition of Motor Devices

⇓ Click to Expand

Step 4: Addition of Vison Sensors

rLab3: Mobile Robot Line Fol

dasl.mem.drexel.edu/~alexAlspach/?page_id=512

Welcome to Open Ro | Spring 2011 | Kinect | Point Cloud | Orocos | Other Bookmarks

Step 4: Control Algorithm

⇑ Click to Expand

To control the robot based on visual feedback, we will need to acquire data from the camera, compute reactive differential wheel speeds, then execute the wheel commands. This algoritm will be contained within the "control_linetracer.cpp" file created at the beginning.

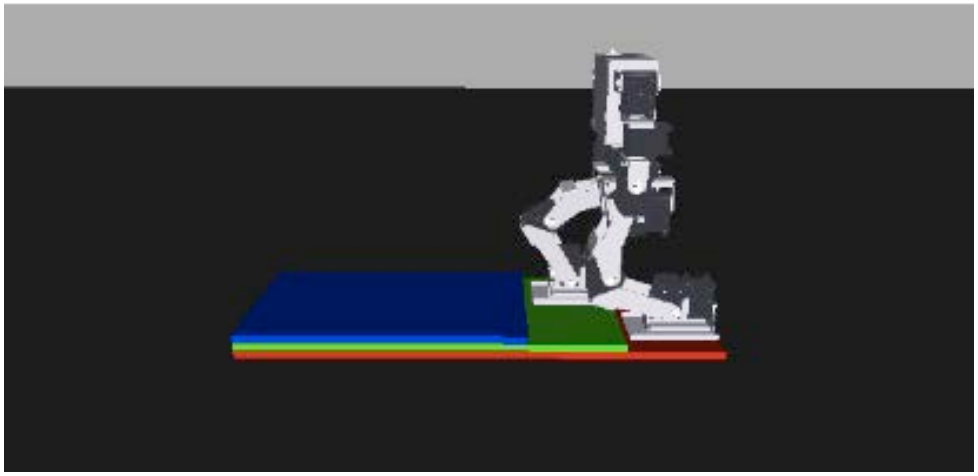
The line tracking is explained line by line in the comments below but here is an overview. Of the 320 rows of pixels available in the camera data, only two lines are analyzed: the middle row and the top row. These two rows are scanned pixel by pixel to find the first and last black pixel in the row, i.e. the line bounding pixels. Once these two pixels are found in each of the two rows, the two central pixels are computed. Effectively, this creates two points on a vector that the robot should follow during the next iteration. Proportional to the angle of the line, a neutral wheel speed and differential speed is computed. The wheel speed is the commanded and the process runs again.

To avoid confusion, the entire source code is posted below with comments for explanation. Code was mainly added with the exception of a part commented out. Original code provided is in gray.

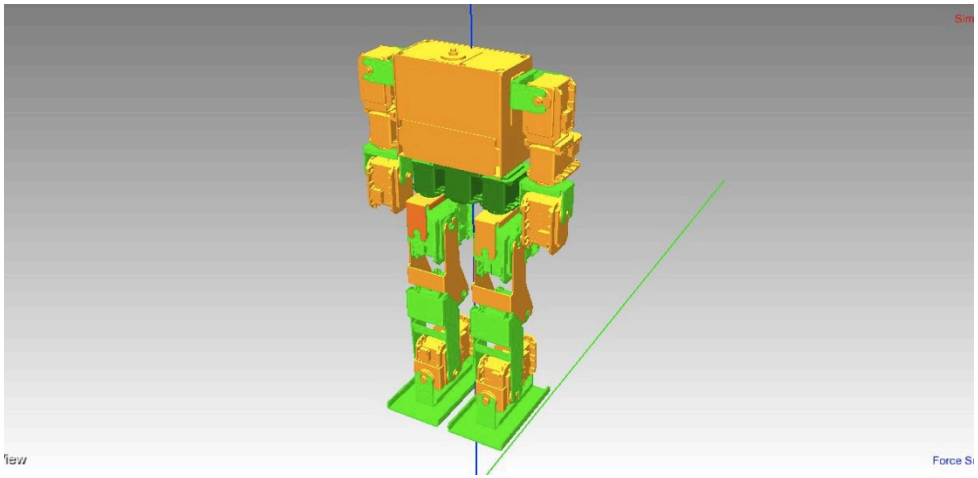
CONTROL_LINETRACER.CPP:

```
001 // control_linetracer.cpp
002 // Simulation control algorithm
003
004 /* RoboticsLab, Copyright 2008-2010 SimLab Co., Ltd. All rights reserved.
005  *
006  * This library is commercial and cannot be redistributed, and/or modified
007  * WITHOUT ANY ALLOWANCE OR PERMISSION OF SimLab Co., LTD.
008  */
009
010
011 #include "control_linetracer.h"
012 #include "control_linetracerCmd.h"
013
014 ////////////////// ADDED //////////////////////////
015 // for use in line tracking()
016 #define GRAY_THR 120    // threshold of gray intensity
017 #define WD_THR 20       // threshold of line width
018 ////////////////// END ADDED //////////////////////
019
020 control_linetracer::control_linetracer(rDC rdc)
021 #ifdef _USE_RCONTROLALGORITHM_EX_
022 :rControlAlgorithmEx(rdc)
023 #else
024 :rControlAlgorithm(rdc)
025 #endif
026 // comment this out otherwise problems will be encountered when building. Used for arms with
    many degrees of freedom
027 //, _jdof(0)
028 {
029 }
030
031 control_linetracer::~control_linetracer()
032 {
033 }
034
035 ////////////////// ADDED //////////////////////////
036 // tracking function called from control function
037 void control_linetracer::tracking()
038 {
039     int i = 0;
040     int begin = -1, end = -1, from = -1, to = -1;
041
042     // the tracking function uses a play and white version of the camera image created using
    PaintItBlack()
```

Simulations and Tutorials



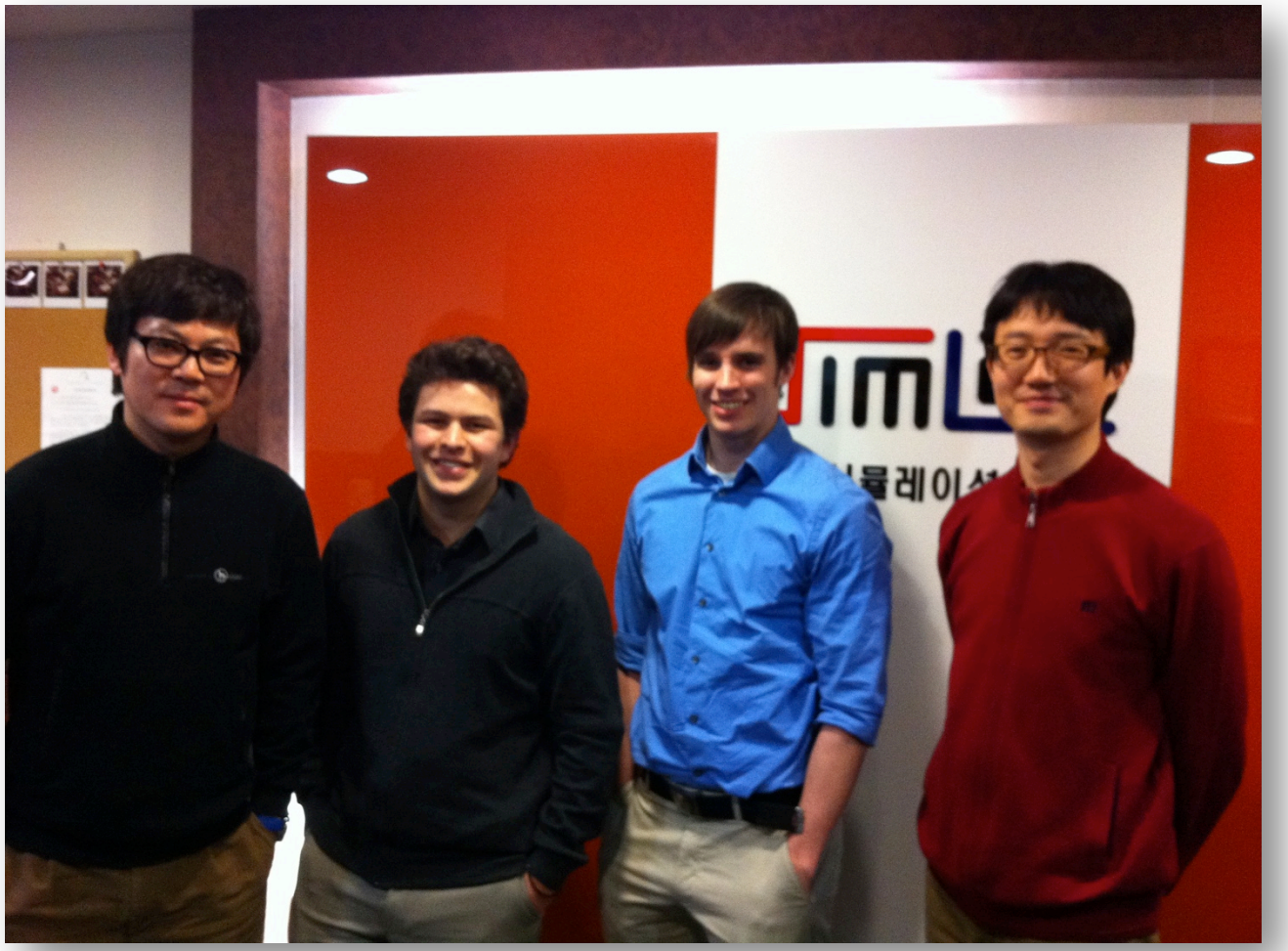
OpenRAVE



RoboticsLab

Simulations and Tutorials

RoboticsLab



Dr. Jonghoon (Miles) Park, Alex Alspach, Sean Mason & Mr. Sang-Yup (Sean) Yi

HUBO Lab Brothers

Outreach

Ms. Yuna Choi and family

Questions?

Thank you.