

# TRIBAL KNOWLEDGE

## DASL + HUBO LAB

---

Alex Alspach • Roy Gross



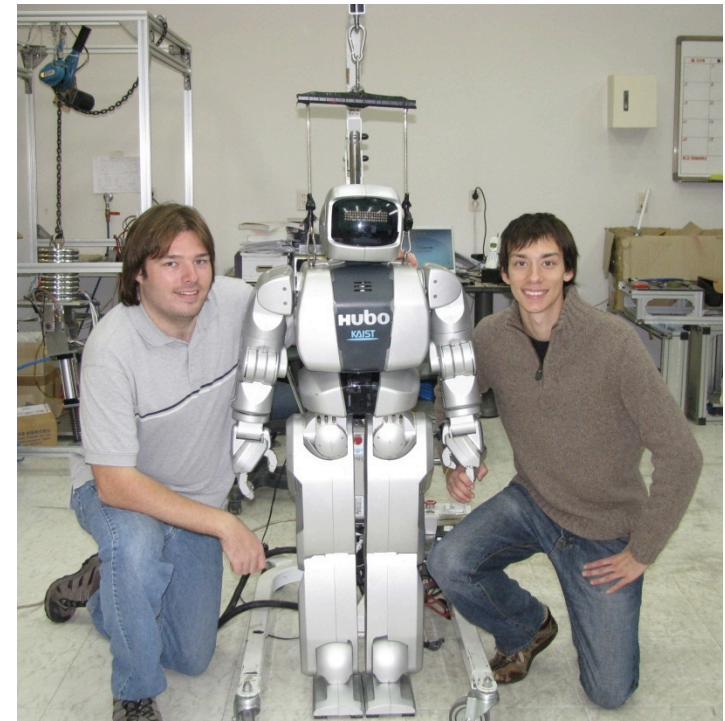
**Welcome to DASL**

# 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

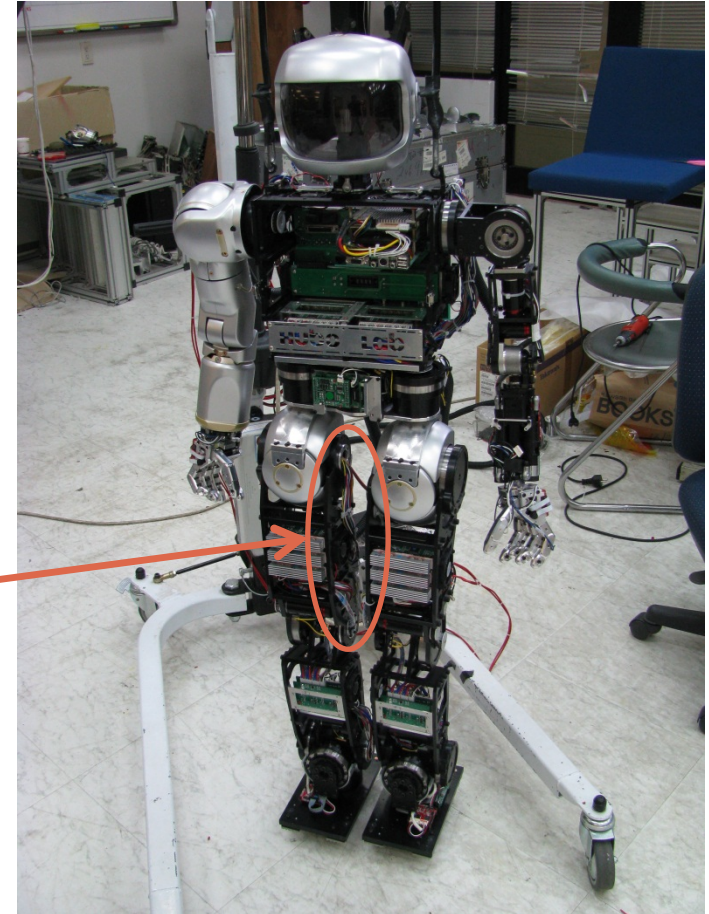
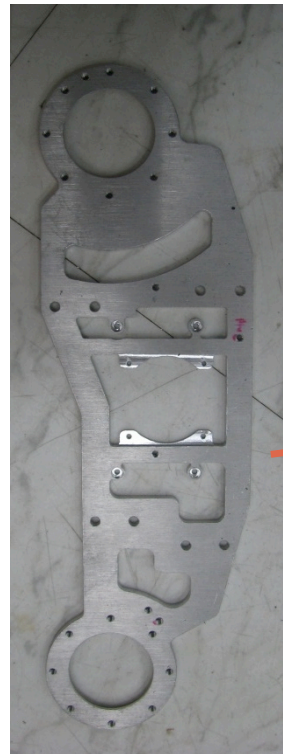
- Goal 1: Transfer data to versatile 3D solid model
- Goal 2: Learn and document manufacturing processes
- Goal 3: Generate approximate physical model (mass, inertial prop.)
- Goal 4: Lay ground-work for HUBO manual

# CAD Transfer



HUBO upper-  
inside leg  
AutoCAD  
drawing

Identify  
actual part

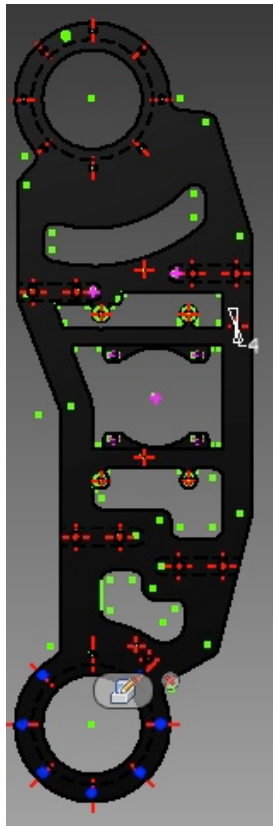


Identify part on assembly



# CAD Transfer

Transfer  
AutoCAD to  
Autodesk  
Inventor



Finished  
3D part

## Manufacturing Notes

- M4 X 0.7\_1
- 3.3 Tap Drill/ M4 Pattern
- 3.3mm Tap Drill #8\_2
- M4 X 0.7\_2
- Circular Pattern2
- 3.3mm Tap Drill #4
- M4 X 0.7\_3
- M4 X 0.7\_4
- M4 X 0.7\_5
- M4 X 0.7\_6
- 3.3mm Tap Drill #2
- M4 X 0.7\_7
- M4 X 0.7\_8
- 5mm Hole # 8
- C5 3mm Hole #4
- 2.5mm Tap Drill
- M3 X 0.5\_1
- M3 X 0.5\_2
- M3 X 0.5\_3
- M3 X 0.5\_4

P - 10.ipt iProperties

General Summary Project Status Custom Save Physical

Solids

The Part Update

Material Aluminum-6061 Clipboard

Density 2.710 g/cm<sup>3</sup> Requested Accuracy Low

General Properties

Center of Gravity

|        |                             |   |                    |
|--------|-----------------------------|---|--------------------|
| Mass   | 0.304 kg (Relative I        | X | 34.295 mm (Relativ |
| Area   | 56737.098 mm <sup>2</sup> ( | Y | 183.412 mm (Relati |
| Volume | 112156.809 mm <sup>3</sup>  | Z | -2.935 mm (Relativ |

Inertial Properties

Principal Global Center of Gravity

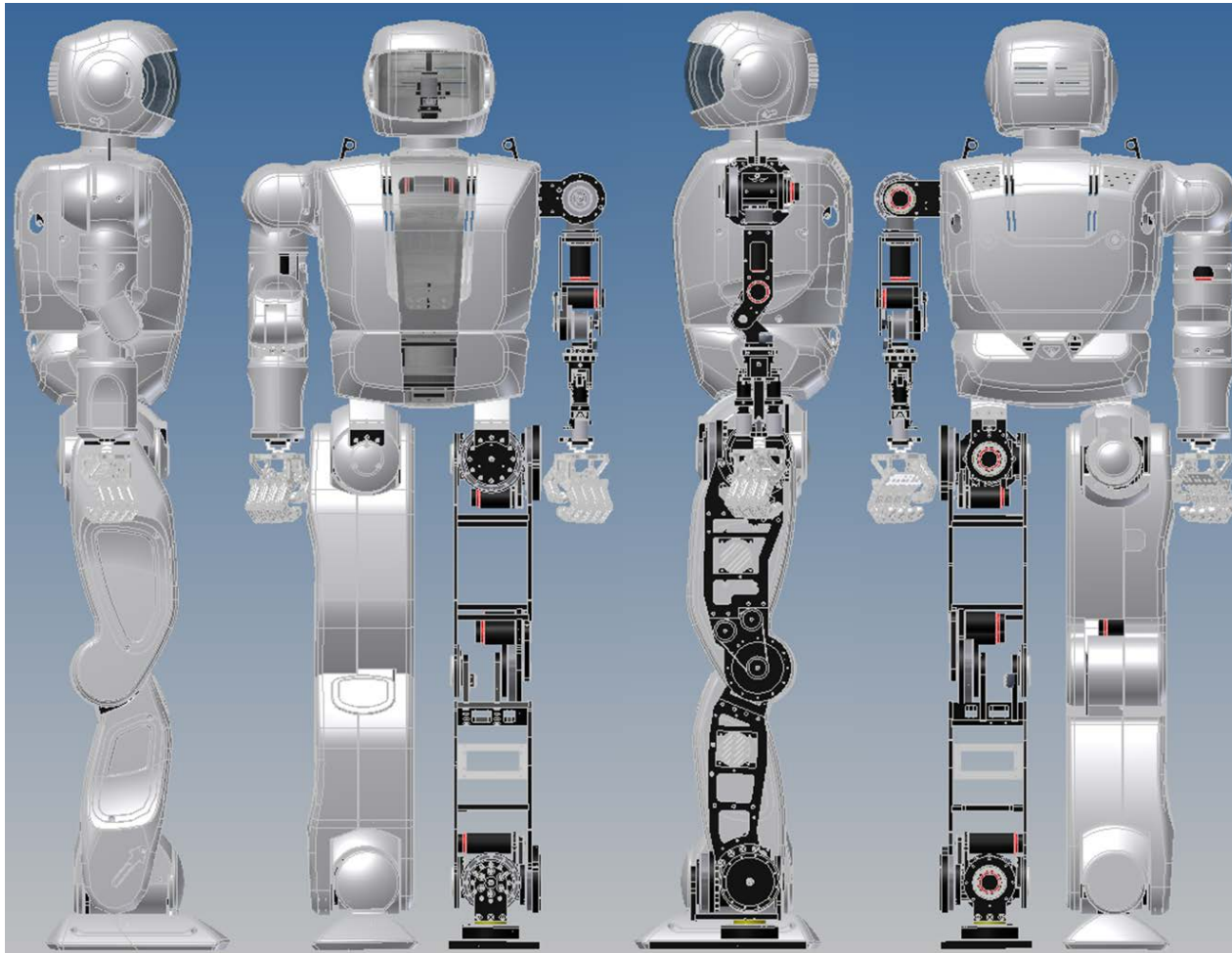
Mass Moments

|     |                             |                                     |                             |
|-----|-----------------------------|-------------------------------------|-----------------------------|
| Ixx | 2841.518 kg mm <sup>2</sup> | Calculated using negative integral. |                             |
| Ixy | 12.042 kg mm <sup>2</sup>   | Iyy                                 | 308.945 kg mm <sup>2</sup>  |
| Ixz | 0.386 kg mm <sup>2</sup>    | Iyz                                 | 0.264 kg mm <sup>2</sup>    |
|     |                             | Izz                                 | 3148.631 kg mm <sup>2</sup> |

## Physical Properties

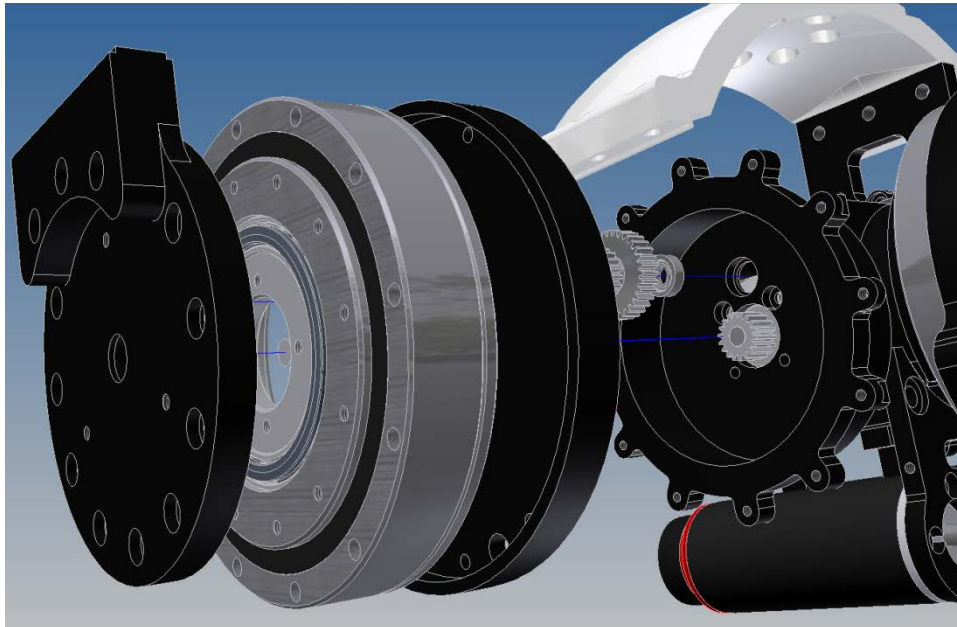
# CAD Transfer

**Repeat 200 Times...**

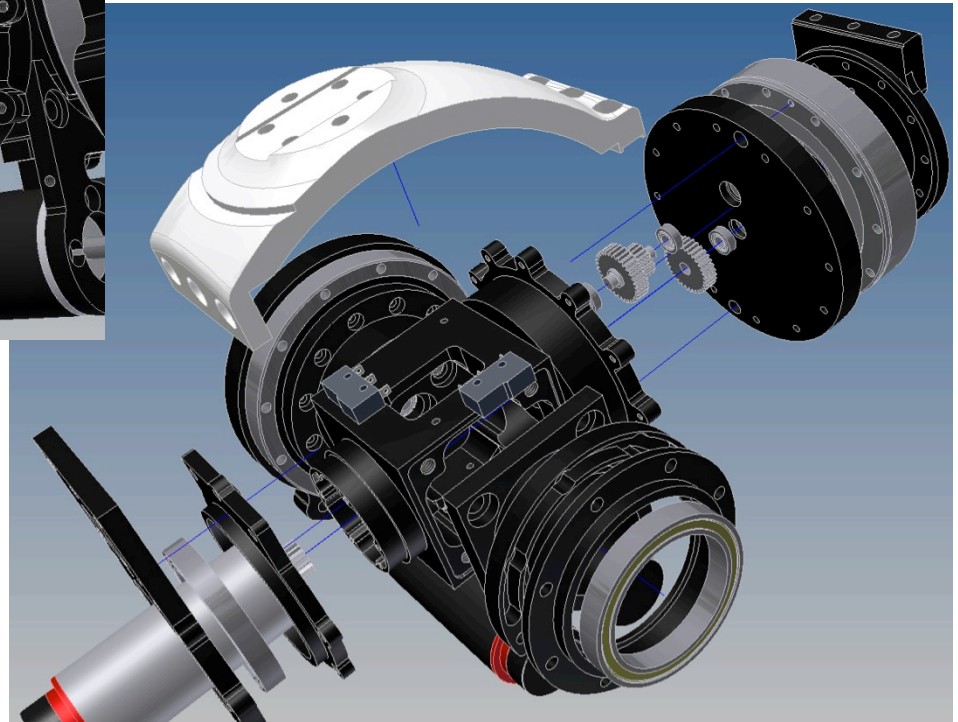




# CAD Transfer

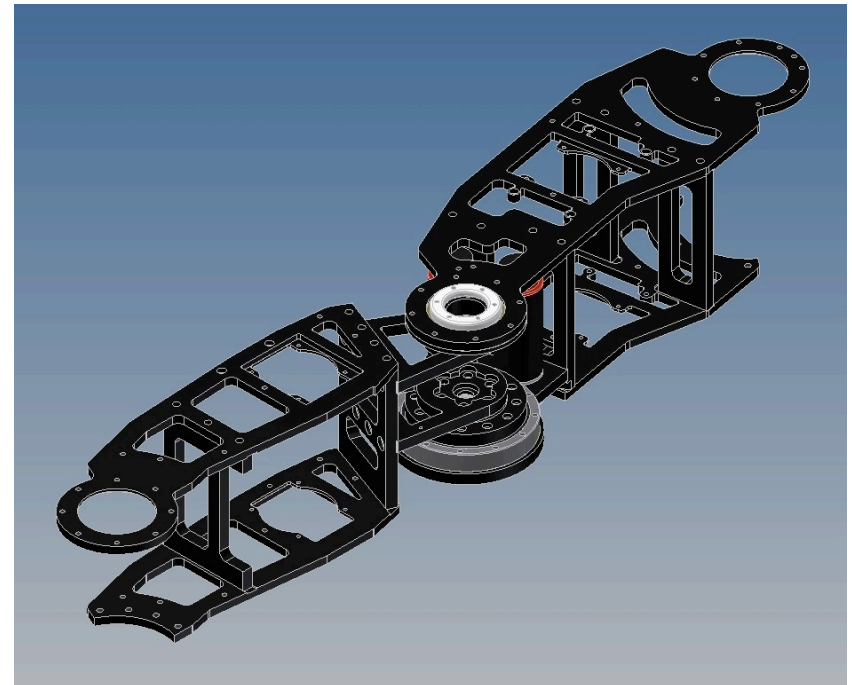


Internal components & Exploded views



# 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



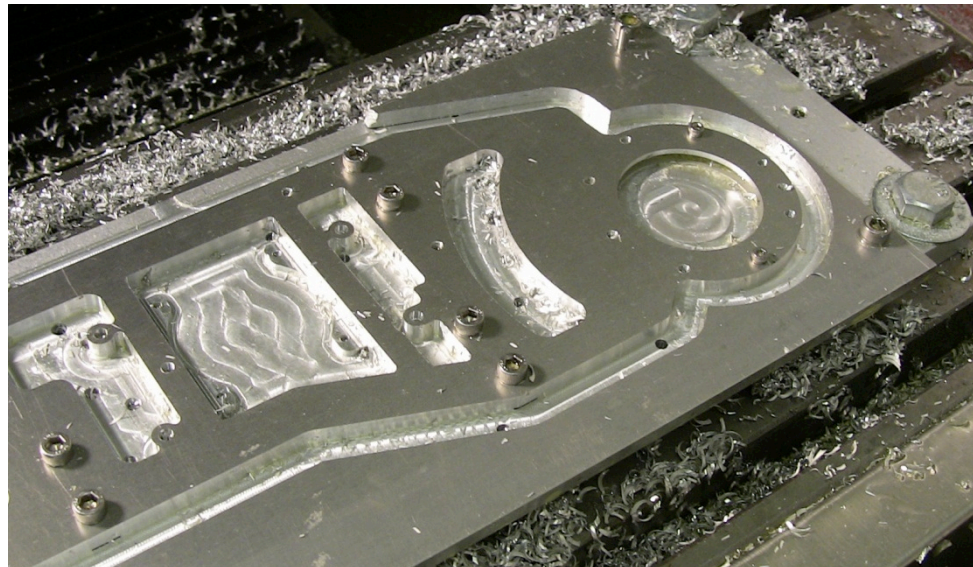
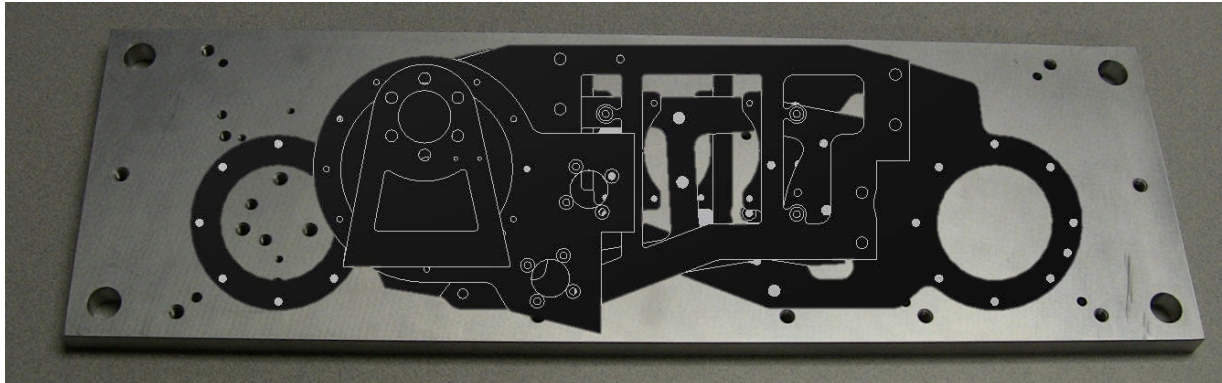
HUBO Lab CNC



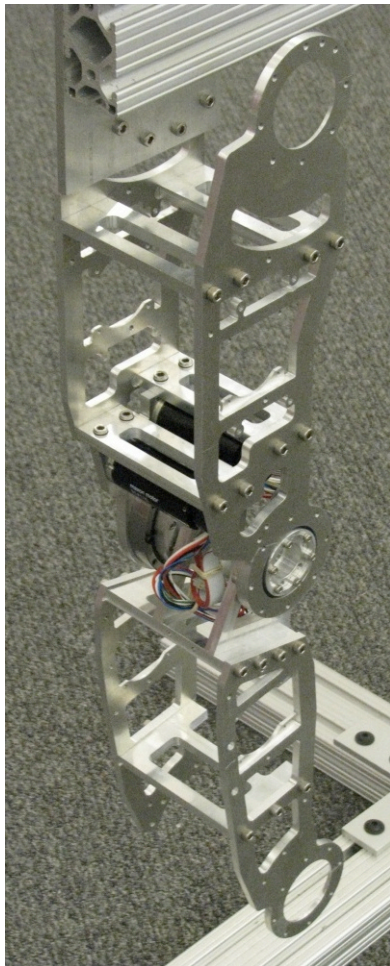
DASL CNC



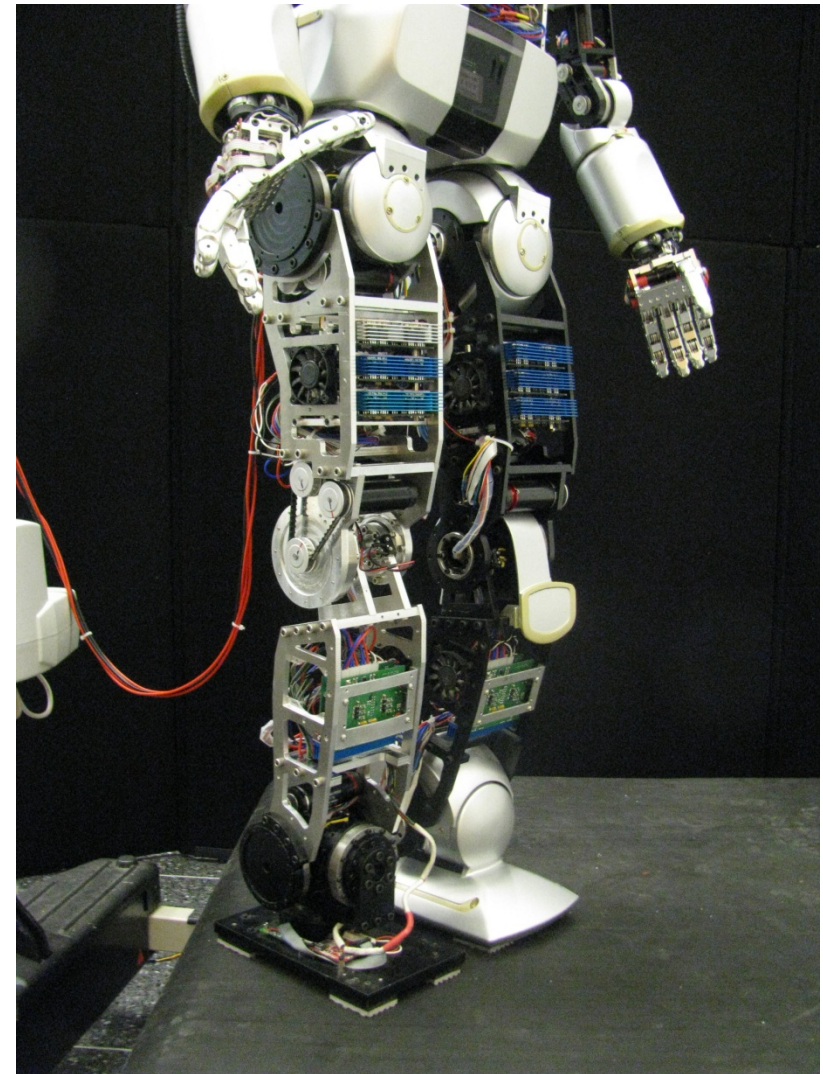
# Model Verification



# Model Verification

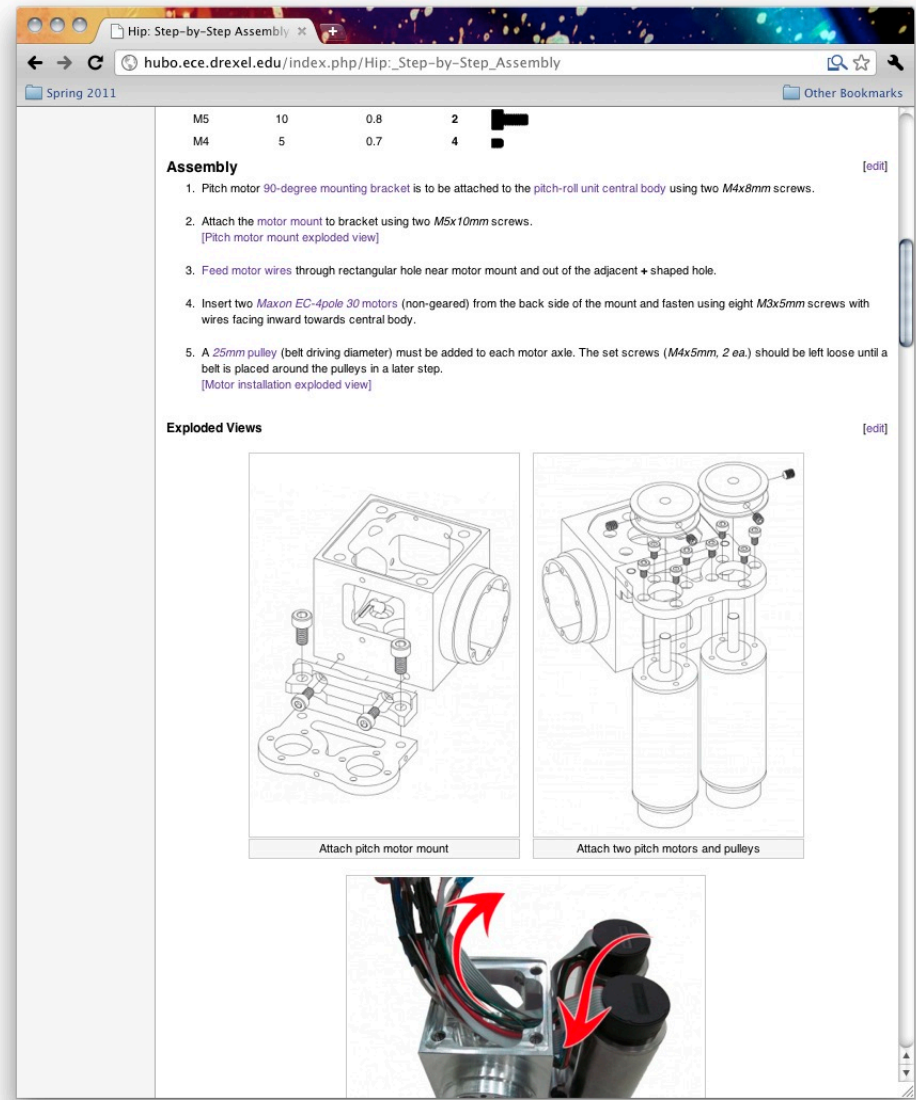
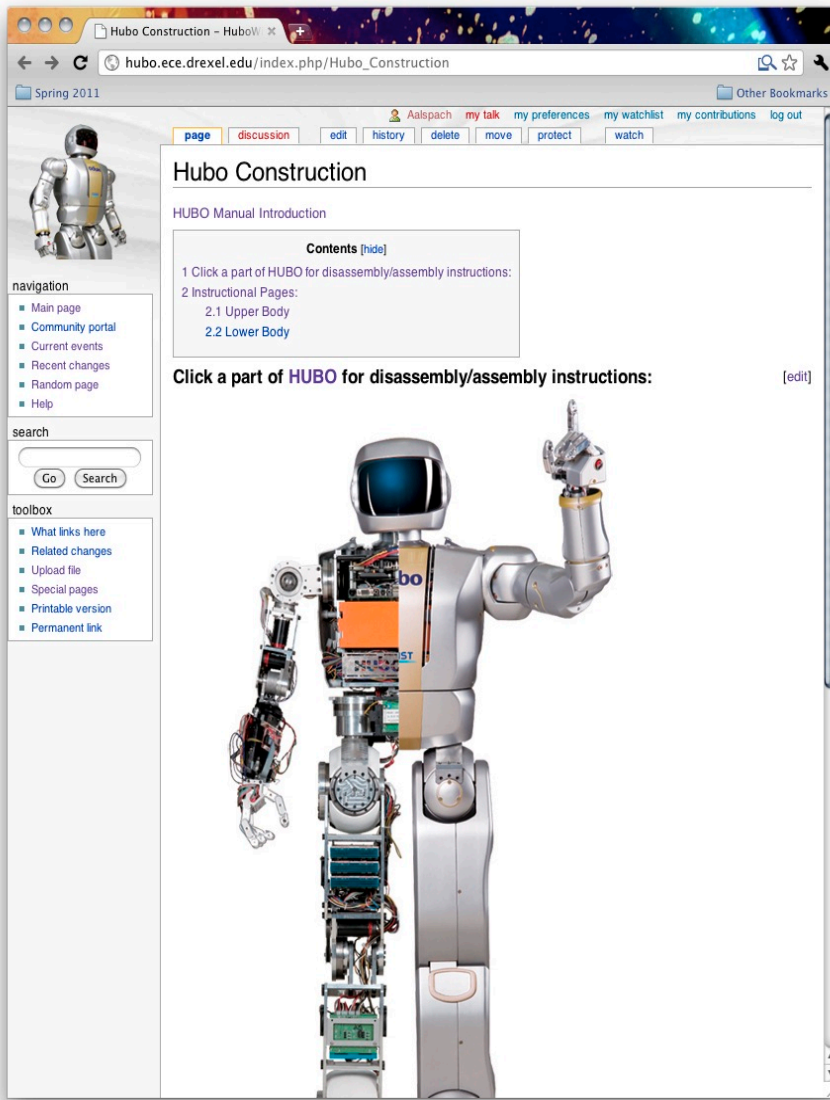


DASL-Machined Leg Replica





# HUBO Assembly Manual



# HUBO Assembly Manual

Waist: Before you start - Tools, Hardware and Components

Contents (hide)

- 1 Tools
- 2 Screws
  - 2.1 Socket Head Cap Screws
  - 2.2 Pan Head Screws
  - 2.3 Set Screws
- 3 Components
  - 3.1 Bearings
  - 3.2 Pulleys
  - 3.3 Belts
  - 3.4 Motors
  - 3.5 Harmonic Drive
  - 3.6 Frame

**Tools**

- 2.0mm hex wrench
- 2.5mm hex wrench
- 3.0mm hex wrench
- 4.0mm hex wrench
- Phillips Screwdriver (#1)

**Screws**

**Socket Head Cap Screws**

| Size | Length(mm) | Pitch(mm) | Quantity | 1:1                                                                                 |
|------|------------|-----------|----------|-------------------------------------------------------------------------------------|
| M3   | 6          | 0.5       | 48       |   |
| M3   | 8          | 0.5       | 12       |  |
| M3   | 25         | 0.5       | 36       |  |
| M4   | 8          | 0.7       | 24       |  |
| M5   | 8          | 0.8       | 12       |  |
| M5   | 10         | 0.8       | 6        |  |

**NOTE: M3x8s must be Steel (not stainless)**

**Pan Head Screws**

| Size | Length(mm) | Pitch(mm) | Quantity | 1:1                                                                                 |
|------|------------|-----------|----------|-------------------------------------------------------------------------------------|
| M2.3 | 10         | 0.4       | 6        |  |

**Set Screws**

| Size | Length(mm) | Pitch(mm) | Quantity | 1:1                                                                                 |
|------|------------|-----------|----------|-------------------------------------------------------------------------------------|
| M4   | 4          | 0.7       | 6        |  |
| M4   | 6          | 0.7       | 6        |  |

**Components**

**Bearings**

**Motors**

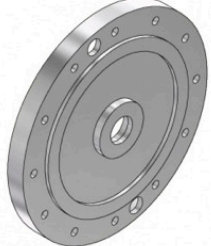


Maxon EC-4pole 30  
(EC-PWERM30 200W 48V)  
**HUBO-01-401G-01**  
3x

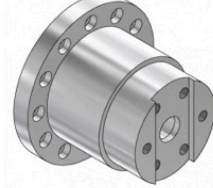
**Harmonic Drive**



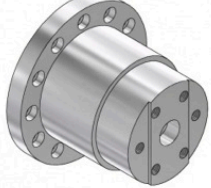
Harmonic Drive Large Cover  
**HUBO-05-102C-01**  
3x



Harmonic Drive Small Cover  
**HUBO-05-103C-01**  
3x

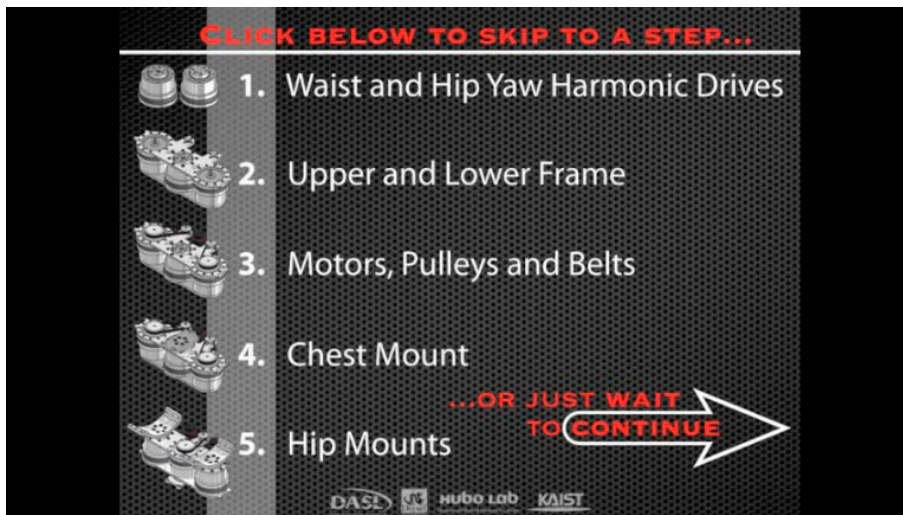
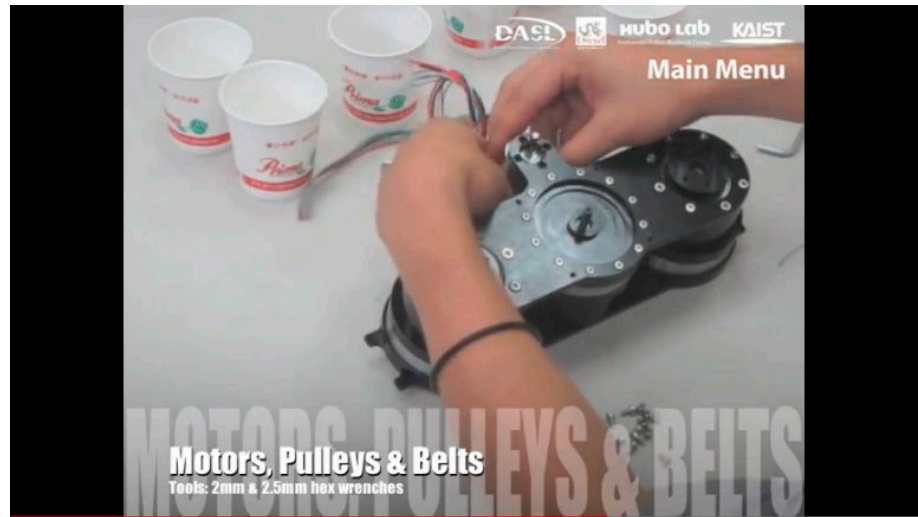


Chest Drive Shaft  
**HUBO-05-104M-01**  
1x



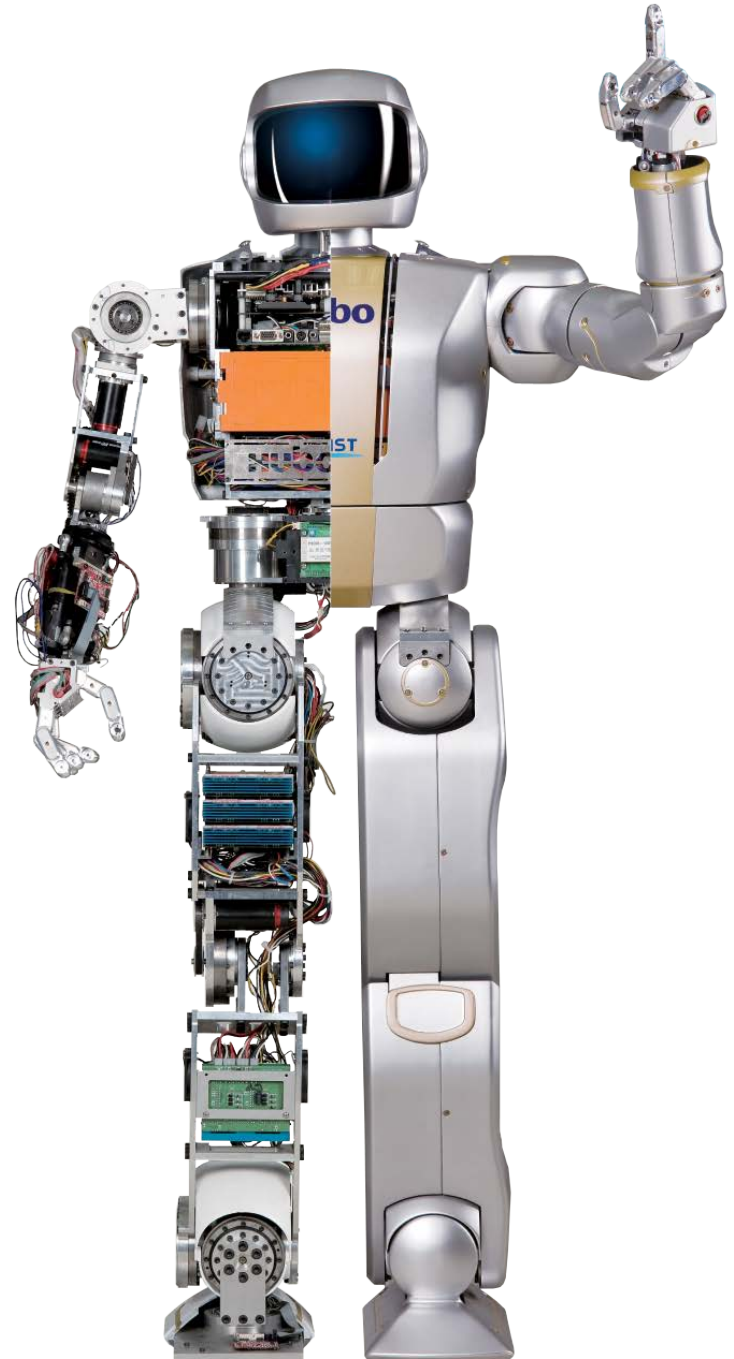
Hip Drive Shaft  
**HUBO-05-105C-01**  
2x

# 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

Line Following with Camera Sensor (Lego NXT Robot) !Under Construction!

dasl.mem.drexel.edu/~seanMason/?page\_id=289

Spring 2011

Sean Mason

HOME DASL 130 Course Website Résumé South Korea Status Reports

**OpenRAVE: Line Following with Camera Sensor (Lego NXT)**

Keywords: openrave, installation, matlab, helloworld, startup, robot, simulation

The video below shows the simulation of a line following robot that has been created in openRAVE. This simulation utilizes the Lego NXT REM robot model and a simulated camera sensor that is available in the openRAVE program for beginner users. OpenRAVE has an extensive list of real-world robot that can be simulated including the PR2, Barrett Arm, Puma, Segway and more. The problem for many users is that this simulation can only act as a simulation because the capitol involved in acquiring one of the mentioned robots is too steep. However, the Lego NXT robot kit is affordable to any college or laboratory and offers a good starting platform for roboticist interested in exploring the capabilities openRAVE. The line following problem was chosen because it is a standard starting problem for a mobile robot platform with either camera or light sensing sensor feedback. This tutorial will show the read how to create a line following workspace, attach a camera sensor, and execute a line following control algorithm in python.

**Motivation and Audience**

This tutorial's motivation is to show how a line following problem can be simulated in OpenRAVE by using the Lego NXT robot built in my previous tutorial, a custom environment that includes a line path, and a camera sensor that is available in openRAVE. Before starting this tutorial the reader should :

- Have openRAVE installed

INFORMATION  
Change this site's admin Theme options

Search

RECENT ENTRY  
2011-02-28 Robotic Conference  
2011-02-22 The Home Stretch  
2011-02-14 Protected: Business  
2011-02-07 Lunar New Year  
2011-01-31 My Spontaneous V

ARCHIVE  
February 2011  
January 2011  
December 2010  
November 2010  
October 2010  
September 2010  
July 2010  
June 2010  
May 2010

CATEGORY  
DASL WORK  
Résumé  
South Korea : M  
South Korea S  
Tutorials

rLab3: Mobile Robot Line Following

dasl.mem.drexel.edu/~alexAlspach/?page\_id=512

Spring 2011

Alex Alspach

If you build it, it will run. // alexalspach@gmail.com

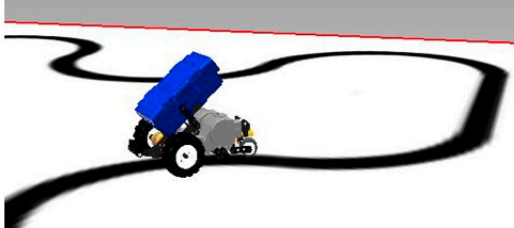
Home Content RSS Log in

Home Korea! Résumé Contact/About MiniHUBO Photos DASL Links DASL140 Tutorials

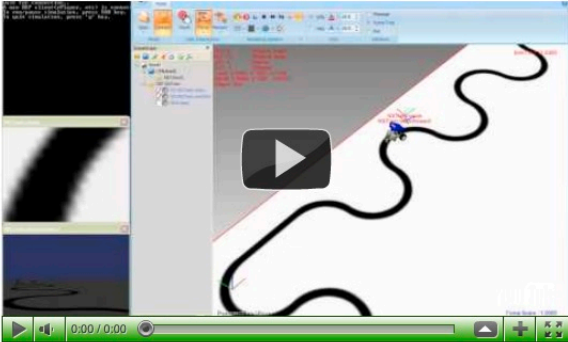
rLab1: 3DOF Manipulator Model | rLab2: Model Importing and Devices | **rLab3: Mobile Robot Line Following** | rLab4: Porting Code to Real Robot

**rLab3: Mobile Robot Line Following**

Keywords: RoboticsLab, Robotics, Simulation, Model, Modeling, Tutorial, SimLab, Rlab, Dynamic, LEGO Mindstorms, Mobile, Line following, Computer vision



[Click for VIDEO](#)



The photo and video depict a model of a LEGO Mindstorms Tribot following a line using computer vision. A model with physical properties, visual representation, free joints (caster), motorized wheels, and sensors (camera) allows you to dynamically simulate robot-environment interaction. The big picture problem is creating a program to load the environment and robot with a control plugin to handle input and command execution. This tutorial provided you with the mobile robot model and a line following code walk-through to familiarize you with RoboticsLab application coding techniques. This tutorial will take about one hour to complete.

► CNC Machining (2)  
► Colleagues (1)  
► DASL (8)  
► HUBO (7)  
► Korea (6)  
► MiniHUBO (3)  
► PRE (3)

Contact/About  
DASL  
DASL140  
Korea!  
Links  
MiniHUBO Photos  
Photography  
TiTVue  
Résumé  
Tutorials  
rLab1: 3DOF Manipulator Model  
rLab2: Model Importing and  
Devices  
rLab3: Mobile Robot Line  
Following  
rLab4: Porting Code to Real  
Robot

April 2011

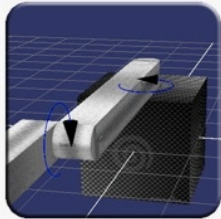
| M  | T  | W  | T  | F  | S  | S  |
|----|----|----|----|----|----|----|
|    |    |    | 1  | 2  | 3  |    |
| 4  | 5  | 6  | 7  | 8  | 9  | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |    |

► Feb

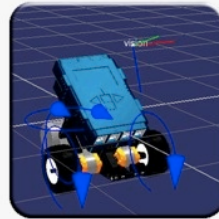
MiniHUBO Fabrication (10)  
Foot (6)  
Hp (4)  
Leg (2)  
Files (2)  
Uncategorized (9)  
AI (2)  
Humanoid (8)  
Koreamazing (11)

# 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



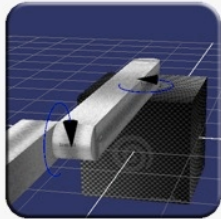
Tutorial 4:  
**Porting Code to  
Real Robot**  
January 27, 2011

|                                                                                  |                                                                                      |
|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|                                                                                  |                                                                                      |
| <p><b>LEGO NXT platform x2</b><br/>Programmed in C++ with OpenCV &amp; NXT++</p> | <p><b>RoboticsLab simulation x4</b><br/>Programmed with C++ and SimLab libraries</p> |

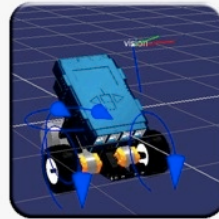
**DASL** **UC** **Hubo Lab** **KAIST**

# Simulations and Tutorials

# RoboticsLab



Tutorial 1:  
**3DOF Manipulator  
Model**  
November 3, 2010



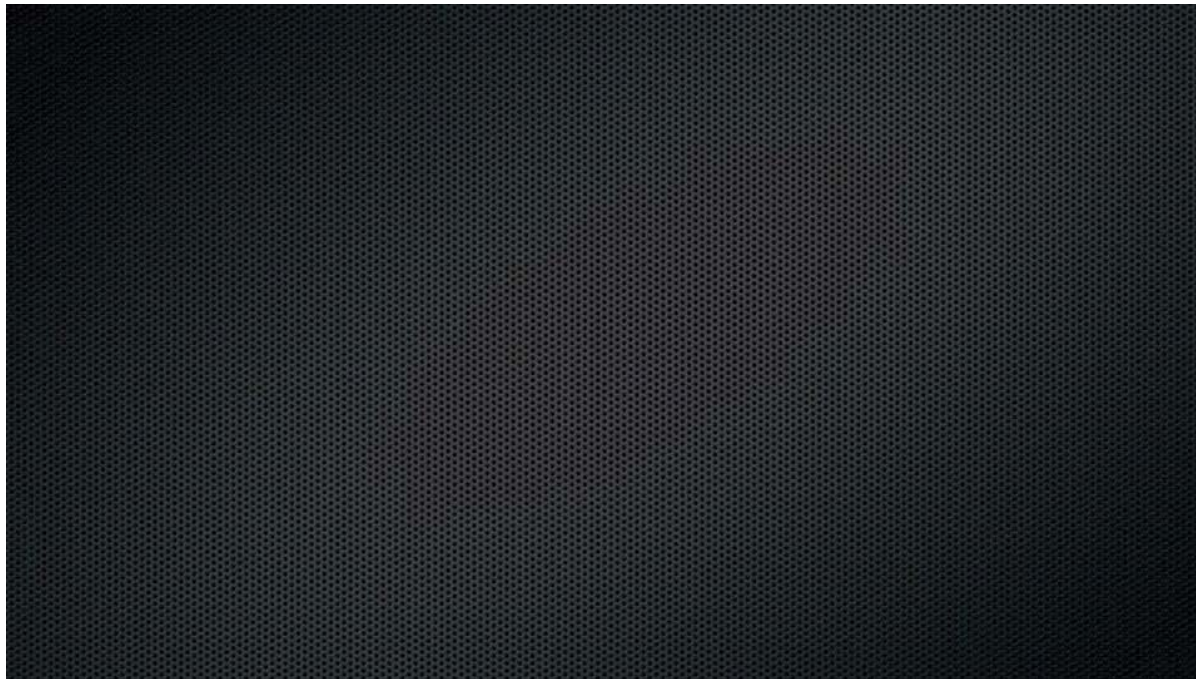
Tutorial 2:  
**Importing  
Geometry and  
Devices**  
December 6, 2010



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.am](#)]

## 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 diagram 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 Vision 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 algorithm 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.

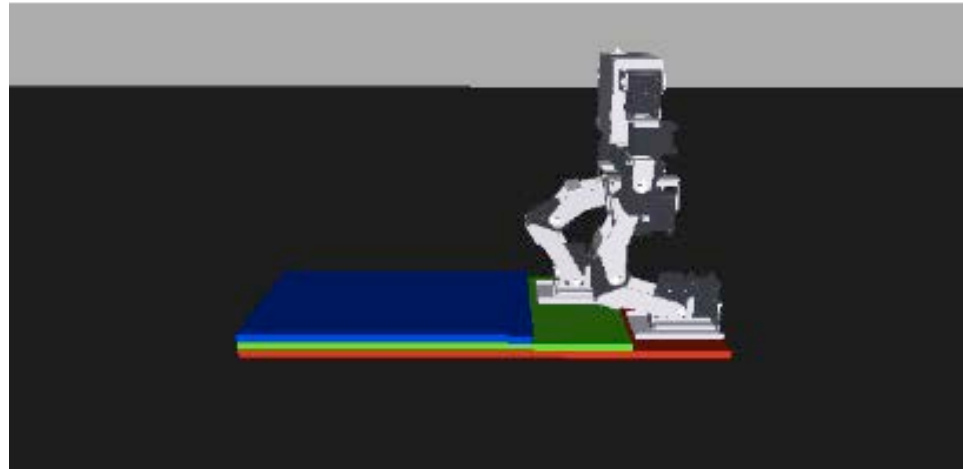
### CONTROL\_LINETRACER.CPP:

```

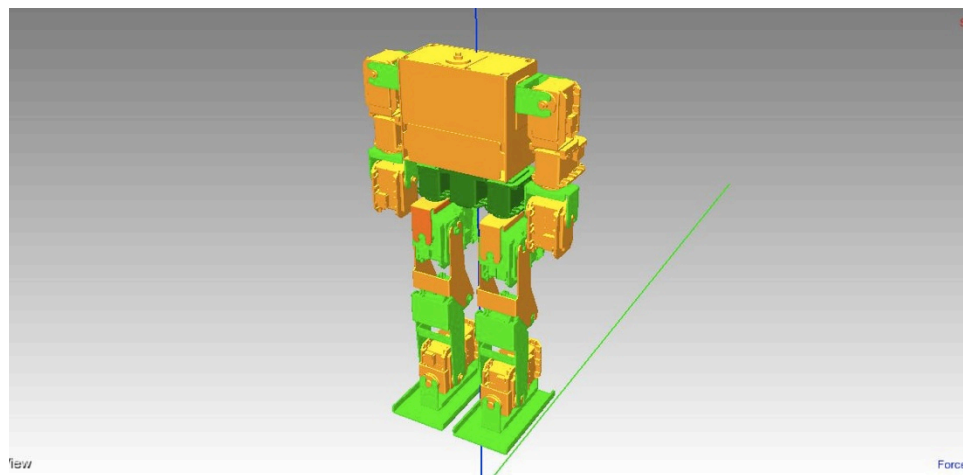
001 // control_linetracer.cpp
002 // Simulation control algorithm
003
004 /* RoboticsLab, Copyright 2008-2010 SimLab Co., Ltd. All rights reserved.
005 * This library is commercial and cannot be redistributed, and/or modified
006 * WITHOUT ANY ALLOWANCE OR PERMISSION OF SimLab Co., LTD.
007 */
008
009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

```

# Simulations and Tutorials



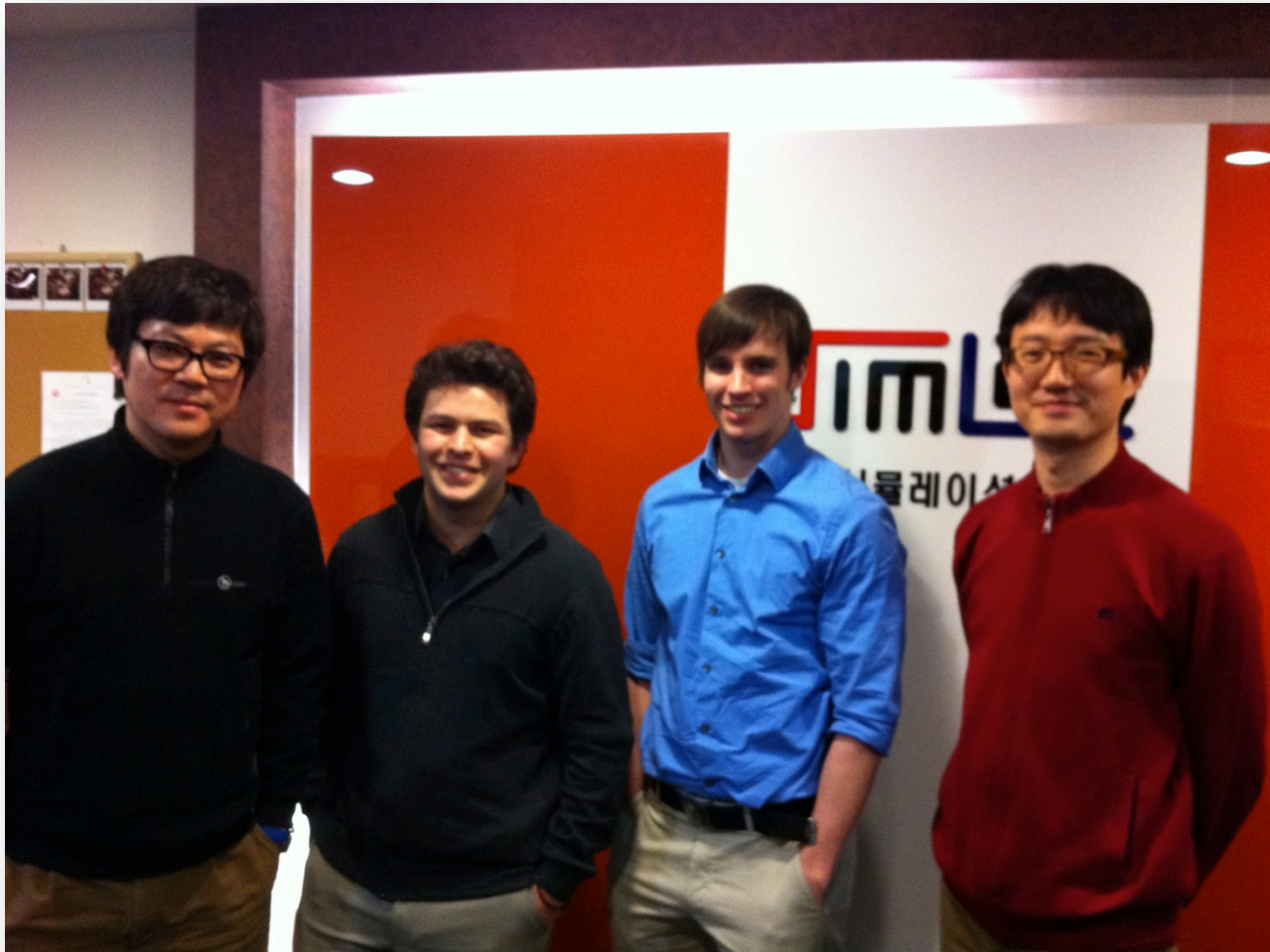
OpenRAVE



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.

