

# TRIBAL KNOWLEDGE

## DASL + HUBO LAB

---

Alex Alspach • Roy Gross

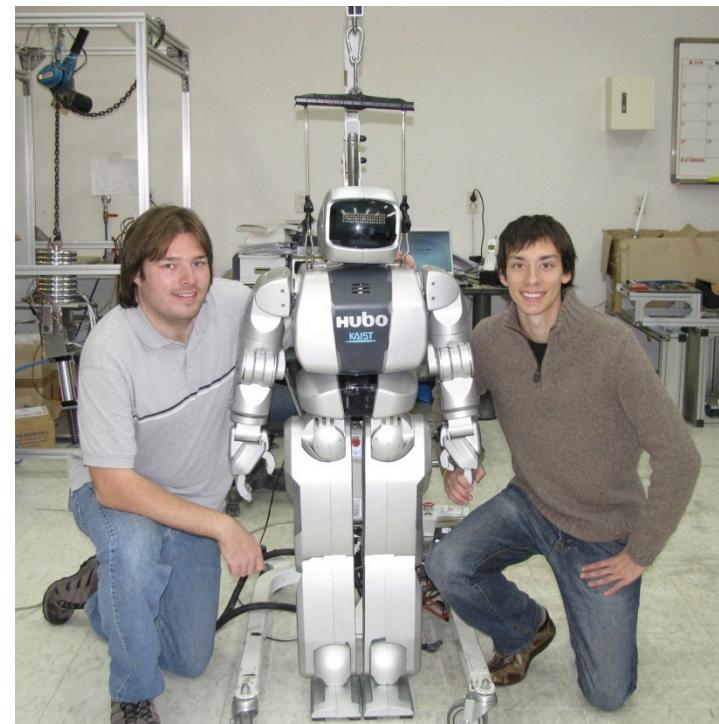


# Topics of Discussion

- 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

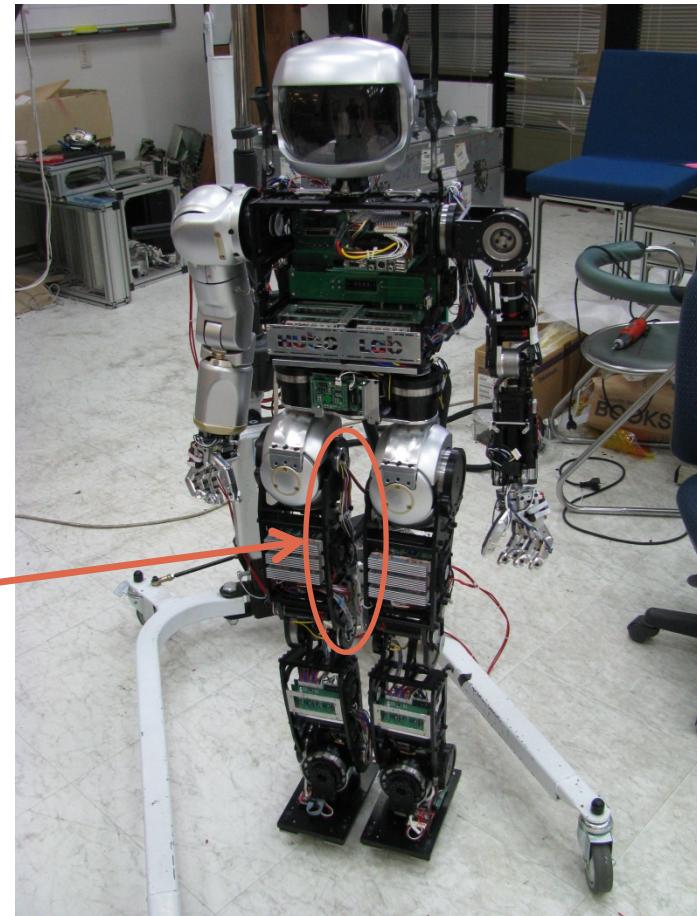
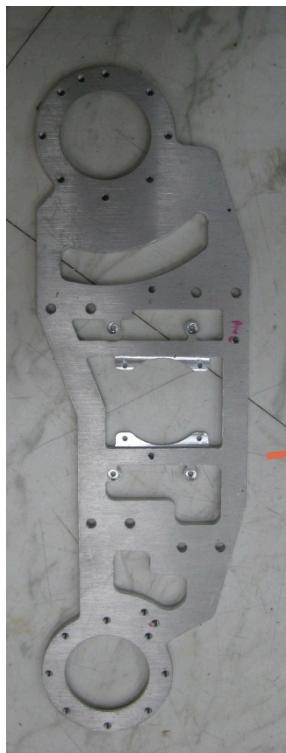
- Goal 1: Transfer data to versatile 3D solid model
- Goal 2: Learn and document manufacturing processes
- Goal 3: Generate approximate physical model (mass, inertial prop.)
- Goal 4: Lay ground-work for HUBO manual

# CAD Transfer



HUBO upper-  
inside leg  
AutoCAD  
drawing

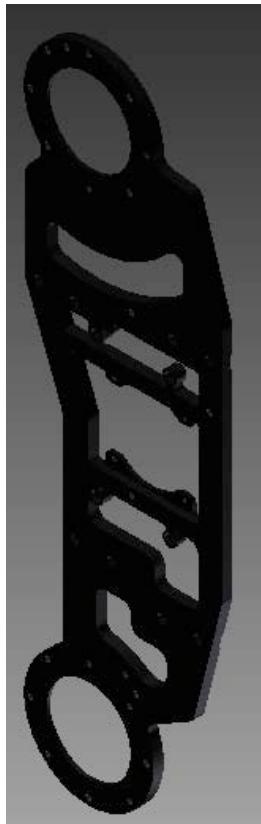
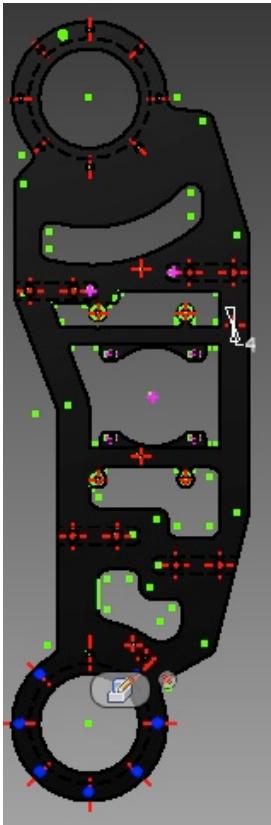
Identify  
actual part



Identify part on assembly

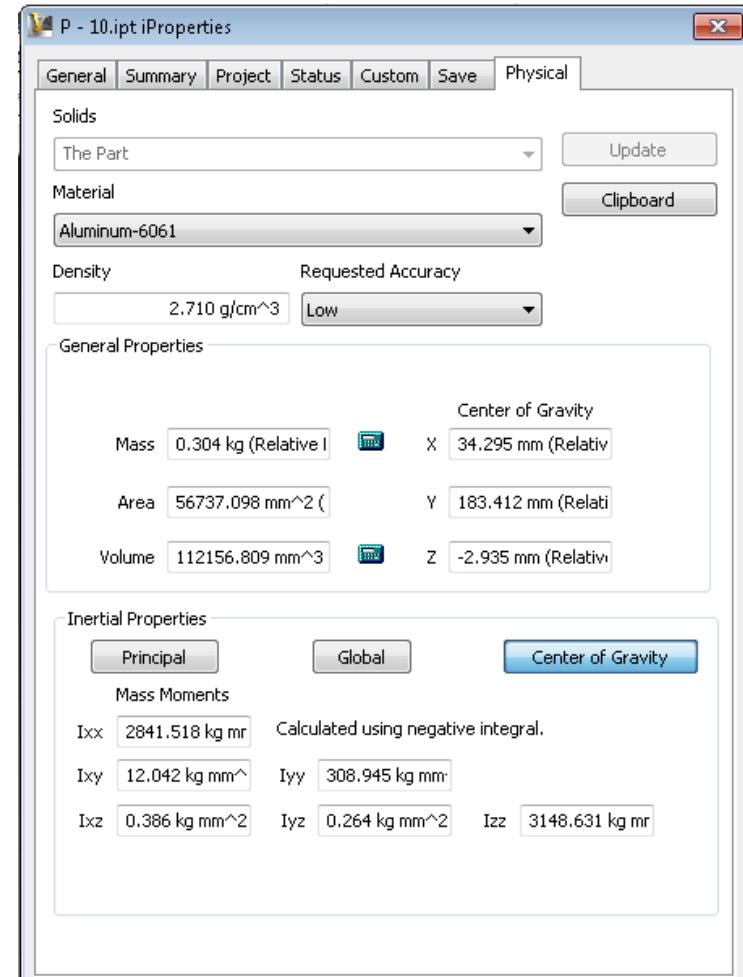
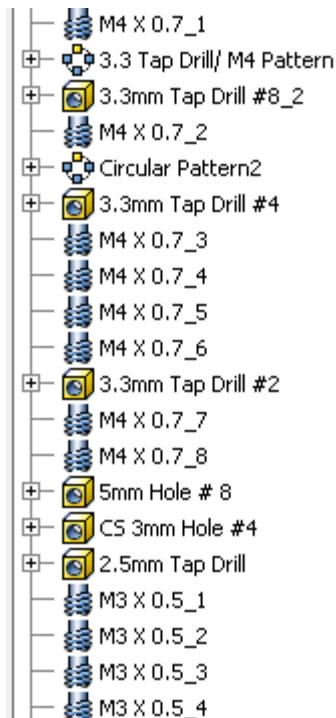
# CAD Transfer

Transfer  
AutoCAD to  
Autodesk  
Inventor



Finished  
3D part

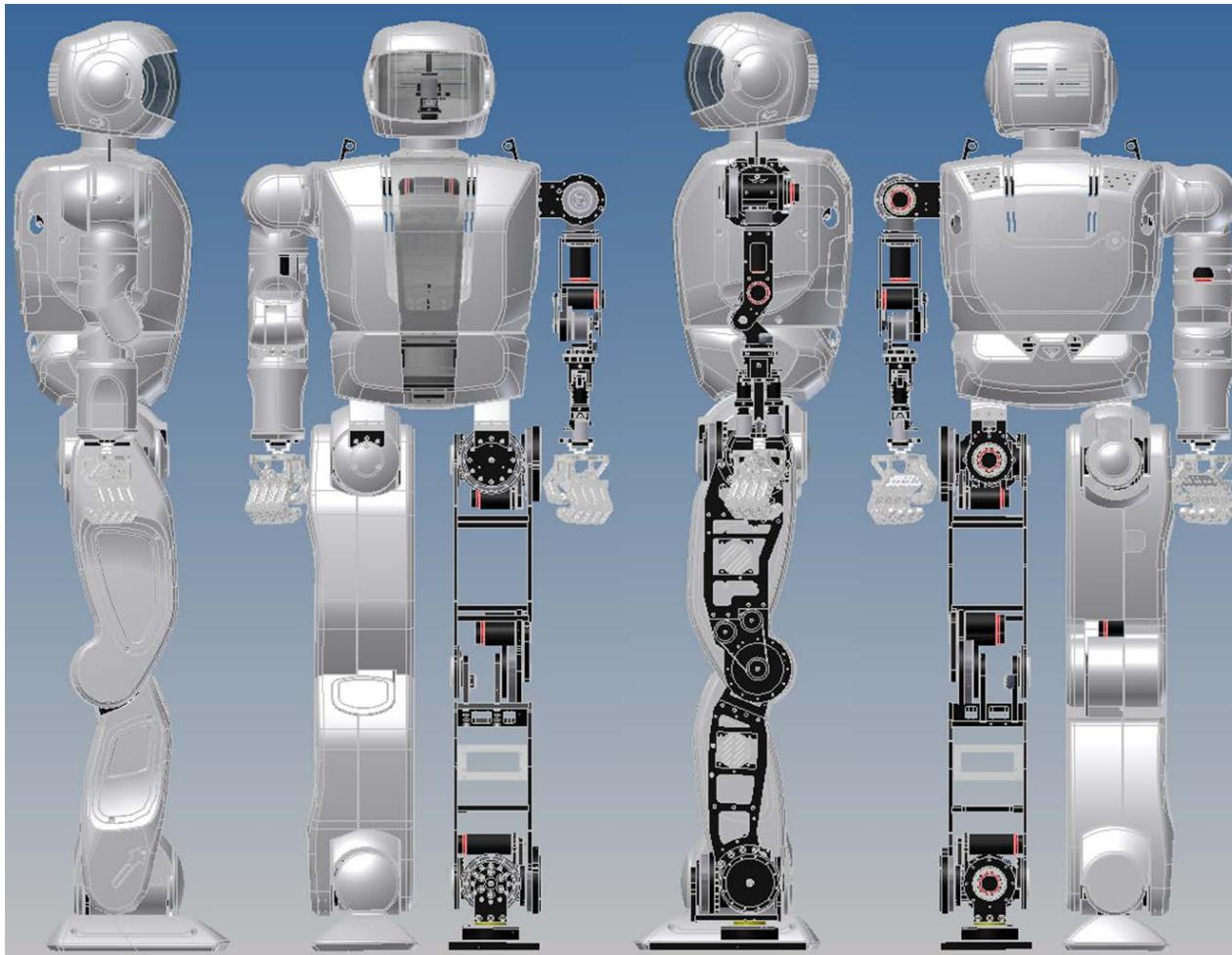
## Manufacturing Notes



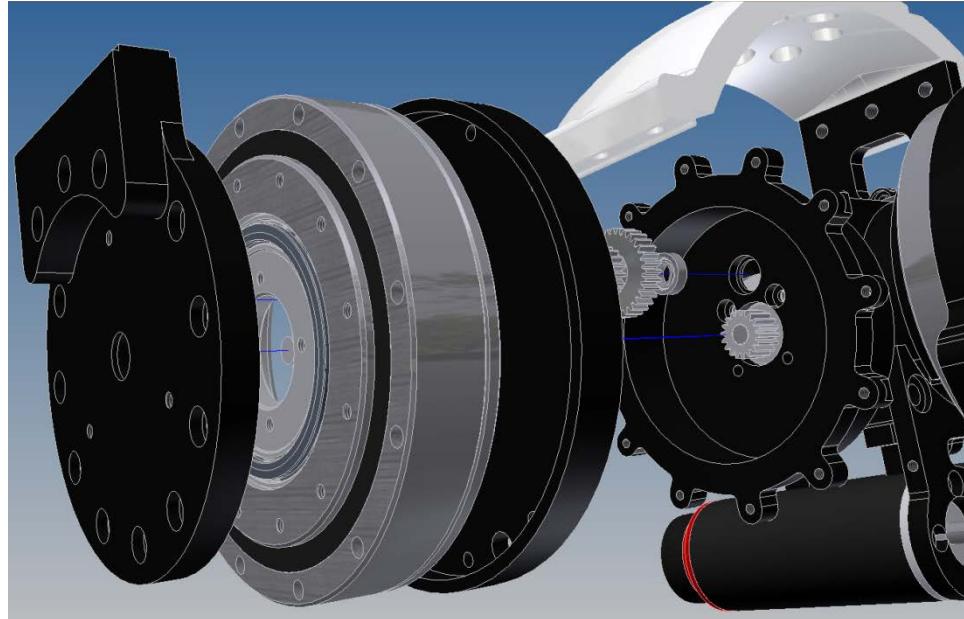
Physical Properties

# CAD Transfer

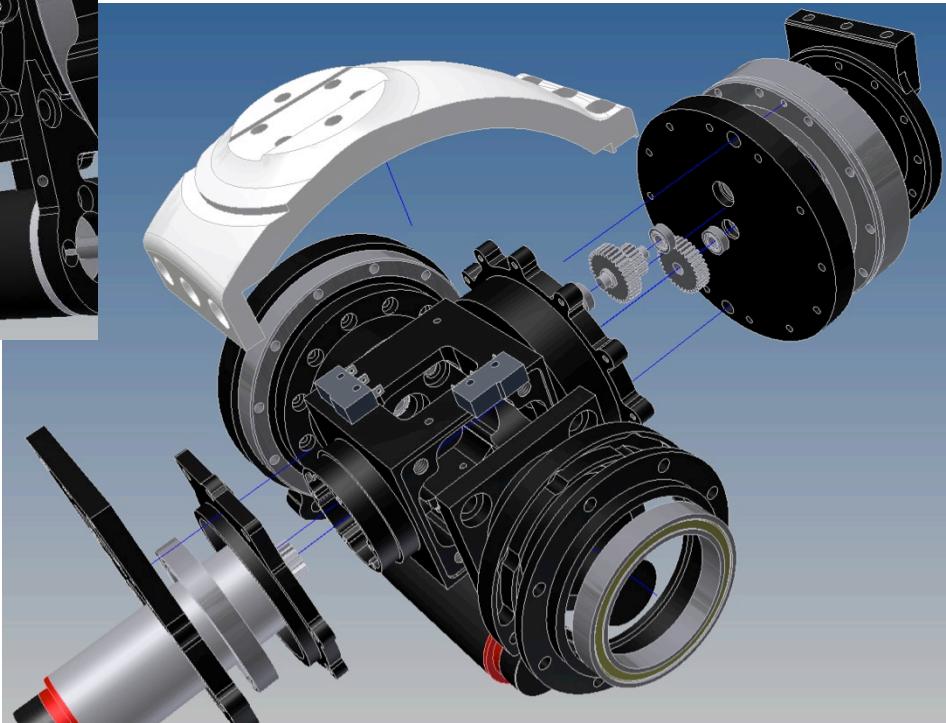
Repeat 200 Times...



# CAD Transfer

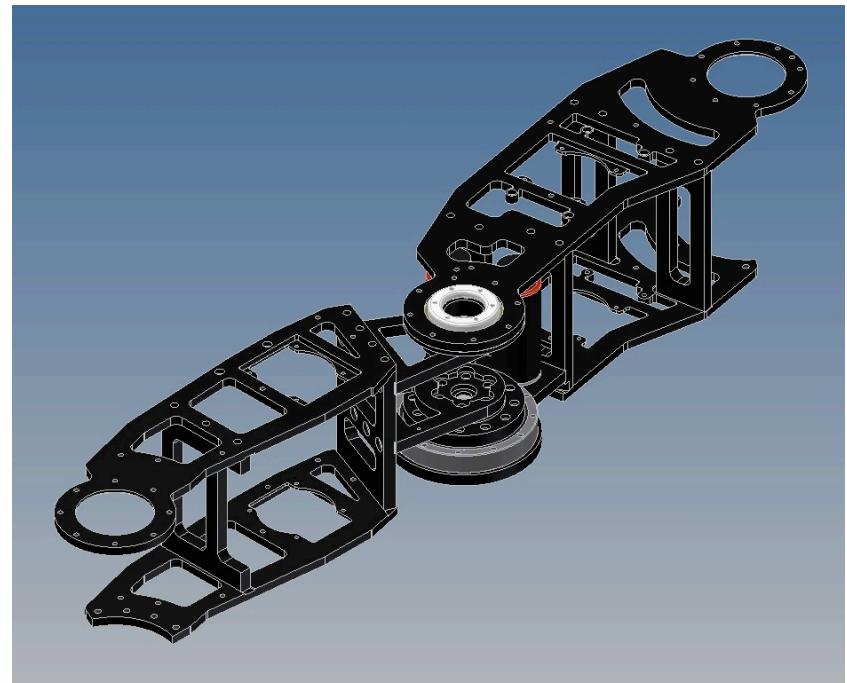


Internal components & Exploded views



# 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

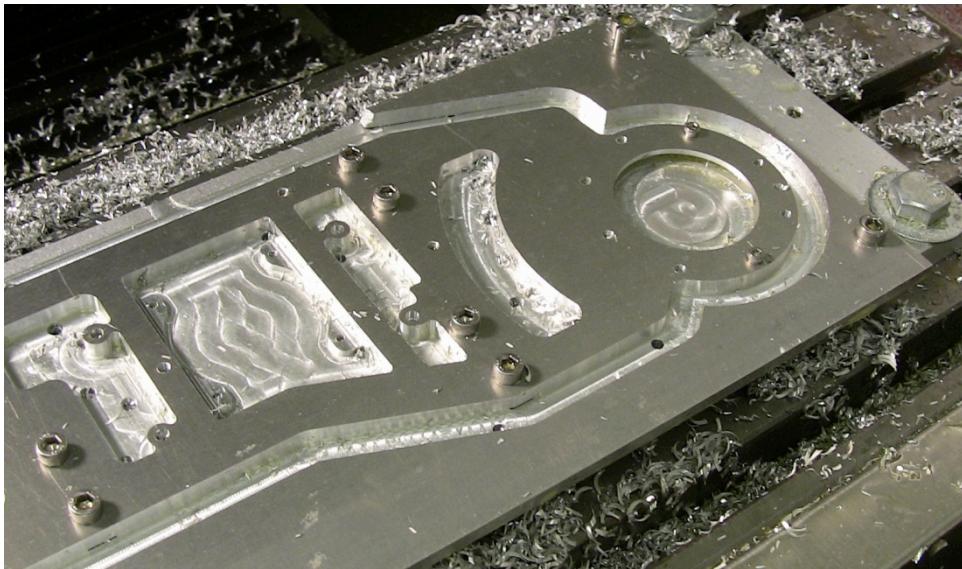
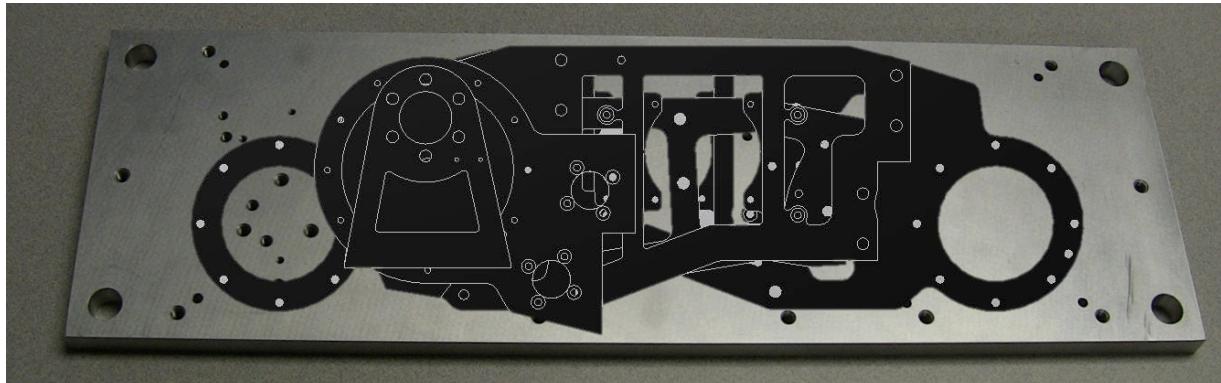


HUBO Lab CNC

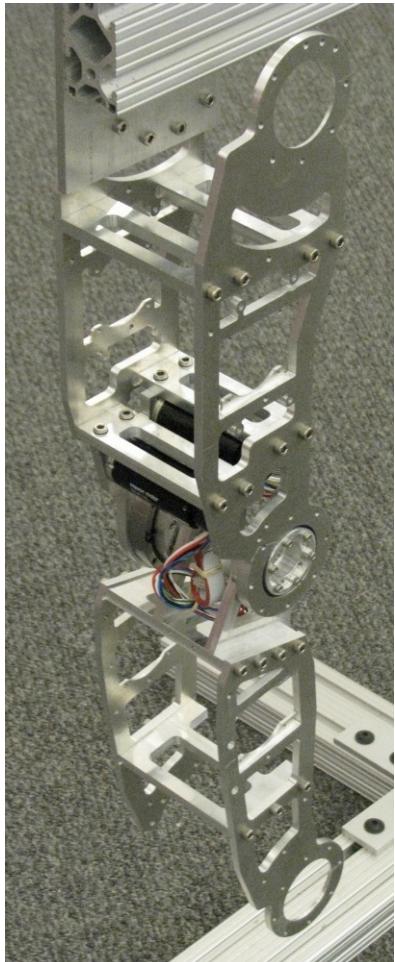


DASL CNC

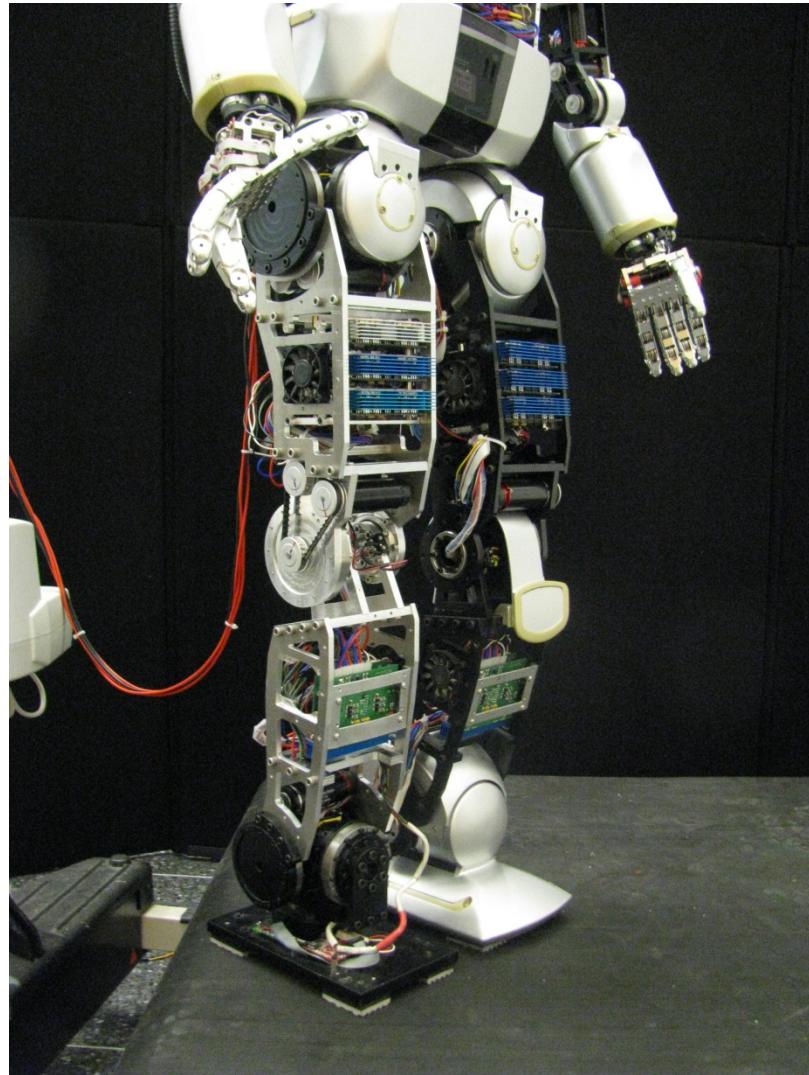
# Model Verification



# Model Verification



DASL-Machined Leg Replica



# HUBO Assembly Manual

Hubo Construction - HuboW □

hubo.ece.drexel.edu/index.php/Hubo\_Construction

Spring 2011 Other Bookmarks

Aalspach my talk my preferences my watchlist my contributions log out

page discussion edit history delete move protect watch

## Hubo Construction

### HUBO Manual Introduction

Contents [hide]

1 Click a part of HUBO for disassembly/assembly instructions:  
2 Instructional Pages:  
    2.1 Upper Body  
    2.2 Lower Body

Click a part of HUBO for disassembly/assembly instructions: [edit]

Hip: Step-by-Step Assembly □

hubo.ece.drexel.edu/index.php/Hip:\_Step-by-Step\_Assembly

Spring 2011 Other Bookmarks

M5 10 0.8 2  
M4 5 0.7 4

### Assembly

[edit]

- Pitch motor 90-degree mounting bracket is to be attached to the pitch-roll unit central body using two M4x8mm screws.
- Attach the motor mount to bracket using two M5x10mm screws.  
[Pitch motor mount exploded view]
- Feed motor wires through rectangular hole near motor mount and out of the adjacent + shaped hole.
- Insert two Maxon EC-4pole 30 motors (non-gearied) from the back side of the mount and fasten using eight M3x5mm screws with wires facing inward towards central body.
- A 25mm pulley (belt driving diameter) must be added to each motor axle. The set screws (M4x5mm, 2 ea.) should be left loose until a belt is placed around the pulleys in a later step.  
[Motor installation exploded view]

### Exploded Views

[edit]

Attach pitch motor mount

Attach two pitch motors and pulleys

# 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

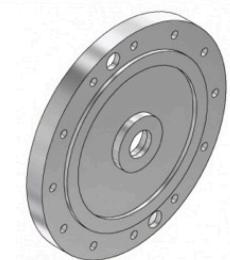
Bearings

Motors



Maxon EC-4pole 30  
(EC-PWERMAX30 200W 48V)  
**HUBO-01-401G-01**  
3x

Harmonic Drive

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

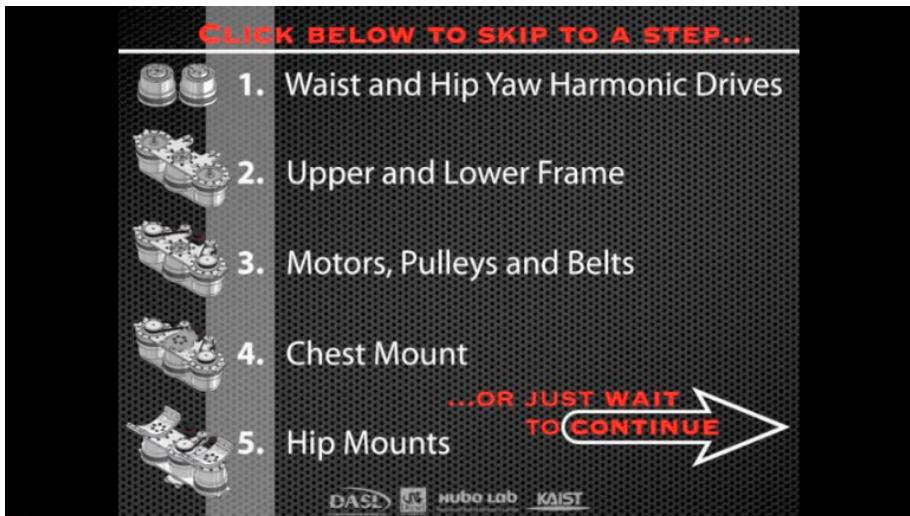
Harmonic Drive Small Cover  
**HUBO-05-103C-01**  
3x



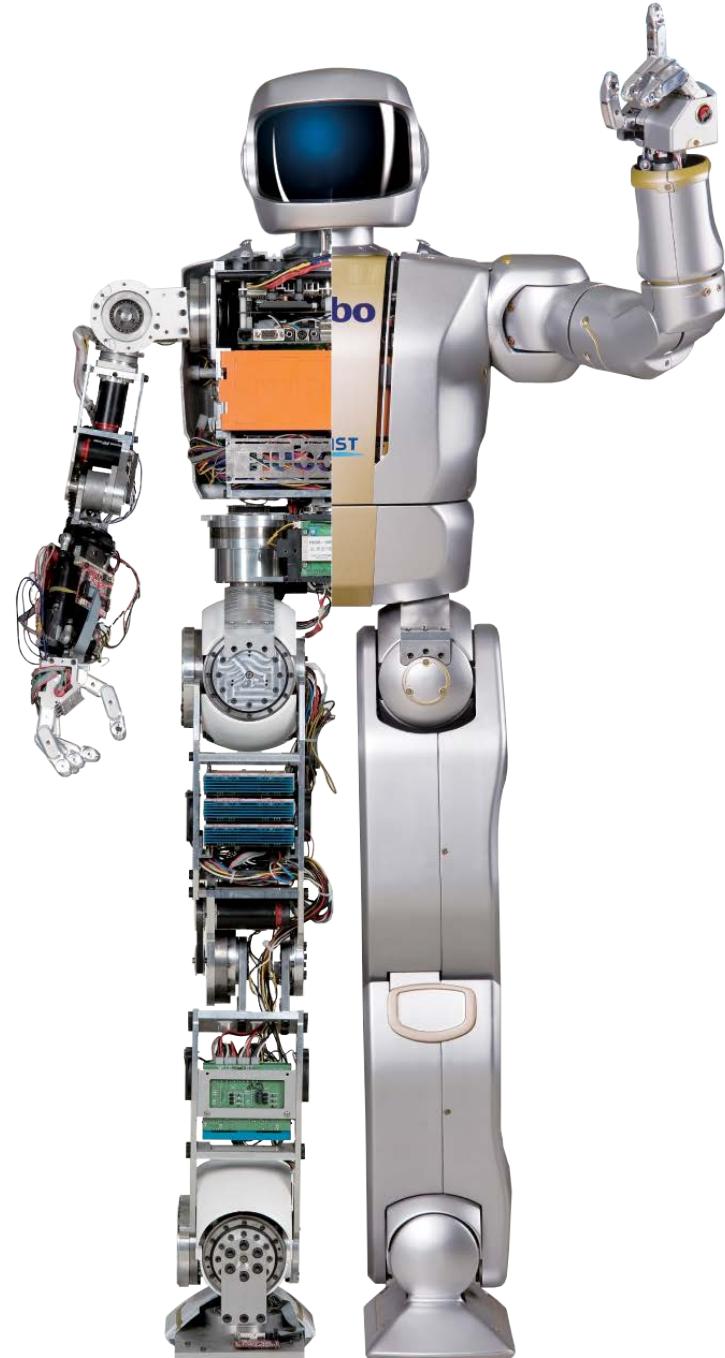

Chest Drive Shaft  
**HUBO-05-104M-01**  
1x

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

**D Line Following with Camera** dasl.mem.drexel.edu/~seanMason/?page\_id=289

Spring 2011 Other Bookmarks

Sean Mason

HOME DASL 130 Course Website Résumé South Korea Status Reports

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

**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 openRave. The reason this tutorial has been written for both line following and using the Lego NXT robot is to act as an introduction to the openRave program for beginner users. OpenRave has an extensive list of real-world robots 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 capital 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 robotists interested in exploring the capabilities of 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 reader 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

## RoboticsLab

rLab3: Mobile Robot Line Fo dasl.mem.drexel.edu/~alexAlspach/?page\_id=512

Spring 2011 Other Bookmarks

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

Home Korean 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

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

**rLab3: Mobile Robot Line Following**

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

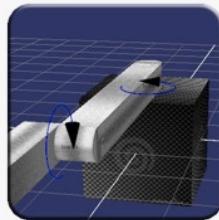
Contact/About  
DASL  
DASL140  
Korean  
Links  
MiniHUBO Photos  
Photography  
TiltView  
Résumé  
Tutorials  
rLab1: 3DOF Manipulator Model  
rLab2: Model Importing and Devices  
rLab3: Mobile Robot Line Following  
rLab4: Porting Code to Real Robot

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	21	22
23	24	25	26	27	28	29
30						

MiniHUBO Fabrication (10)  
Foot (5)  
Hip (4)  
Leg (2)  
Files (2)  
Uncategorized (9)  
AI (2)  
Humanoid (8)  
Koreamazing (11)

# Simulations and Tutorials

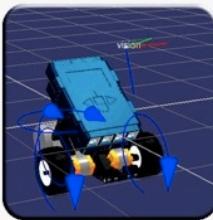
# RoboticsLab



Tutorial 1:

**3DOF Manipulator Model**

November 3, 2010



Tutorial 2:

**Importing Geometry and Devices**

December 6, 2010



Tutorial 3:

**Mobile Robot Line Following**

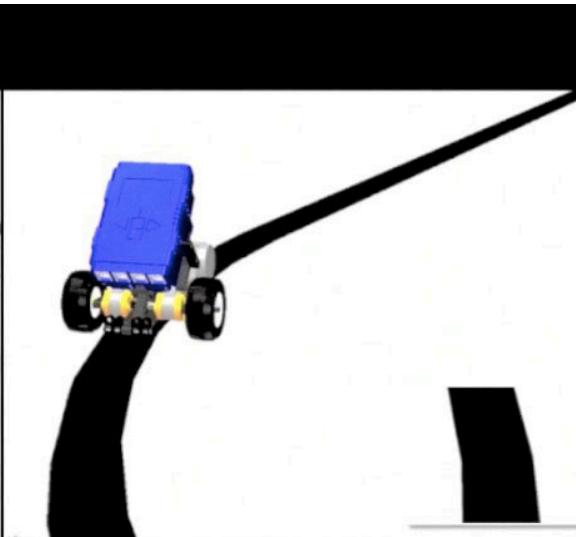
November 24, 2010



Tutorial 4:

**Porting Code to Real Robot**

January 27, 2011

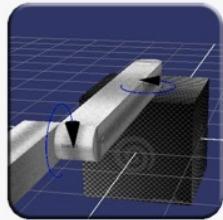


**LEGO NXT platform x2**  
Programmed in C++ with OpenCV & NXT++

**RoboticsLab simulation x4**  
Programmed with C++ and SimLab libraries

# 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**

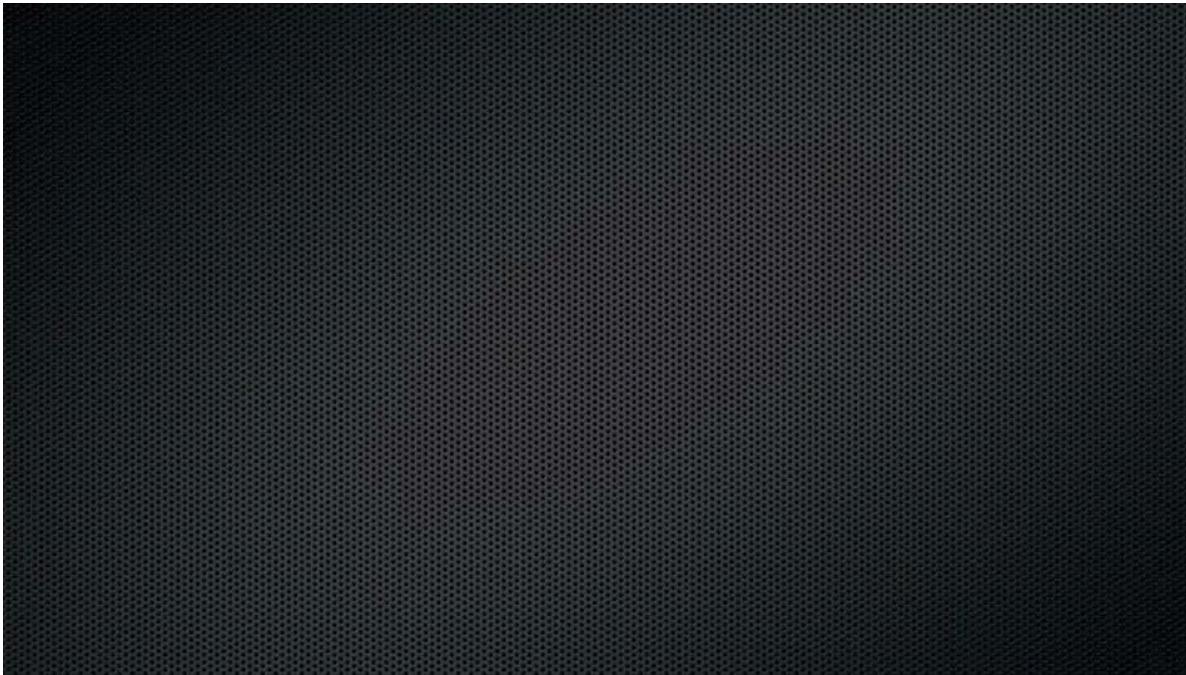
November 24, 2010



Tutorial 4:

**Porting Code to Real Robot**

January 27, 2011



# Simulations and Tutorials

# RoboticsLab

[rLab2. Model Importing and ...](#)

Welcome to Open Ro... Spring 2011 Kinect Point Cloud Orosco... 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 NXTram.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 Vison Sensors

[rLab3: Mobile Robot Line Fo...](#)

Welcome to Open Ro... Spring 2011 Kinect Point Cloud Orosco... 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.

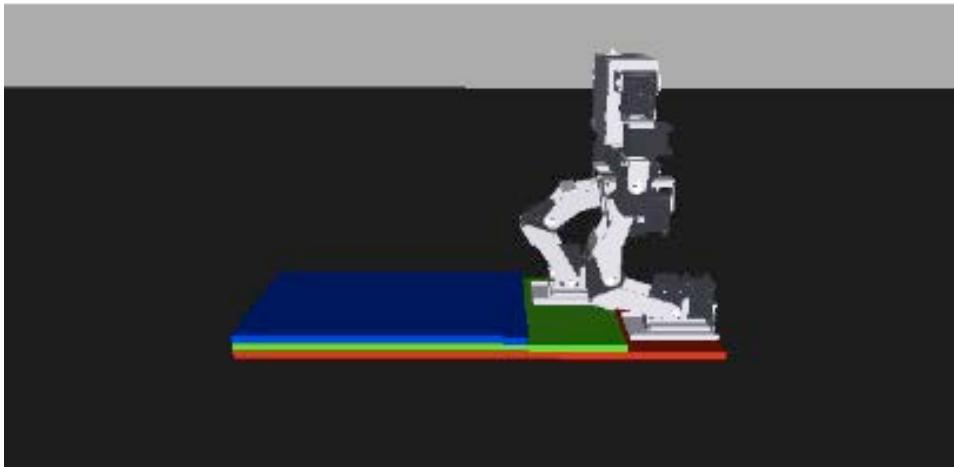
```
CONTROL_LINETRACER.CPP:
```

```

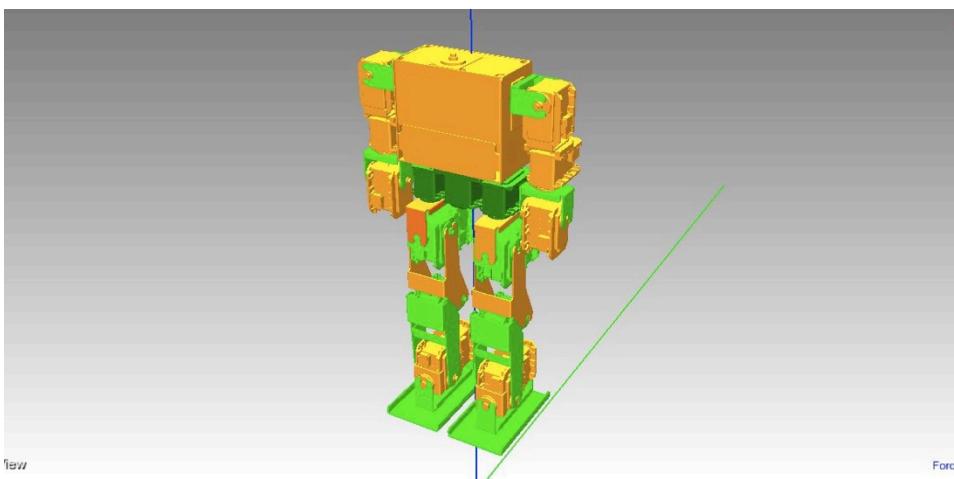
001 // control_linetracer.cpp
002 // Simulation control algorithm
003
004 /* RoboticsLab, Copyright 2008-2010 SimLab Co., Ltd. All rights reserved.
005 *
006 * This library is commercial and cannot be redistributed, and/or modified
007 * WITHOUT ANY ALLOWANCE OR PERMISSION OF SimLab Co., LTD.
008 */
009
010
011 #include "control_linetracer.h"
012 #include "control_linetracerCmd.h"
013
014 ////////////////////////////////////////////////////////////////// ADDED //////////////////////////////////////////////////////////////////
015 // for use in line tracking()
016 #define GRAYTHR 20 // threshold of gray intensity
017 #define WDTHR 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
027 // many degrees of freedom
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
043     PaintItBlack()

```

# Simulations and Tutorials

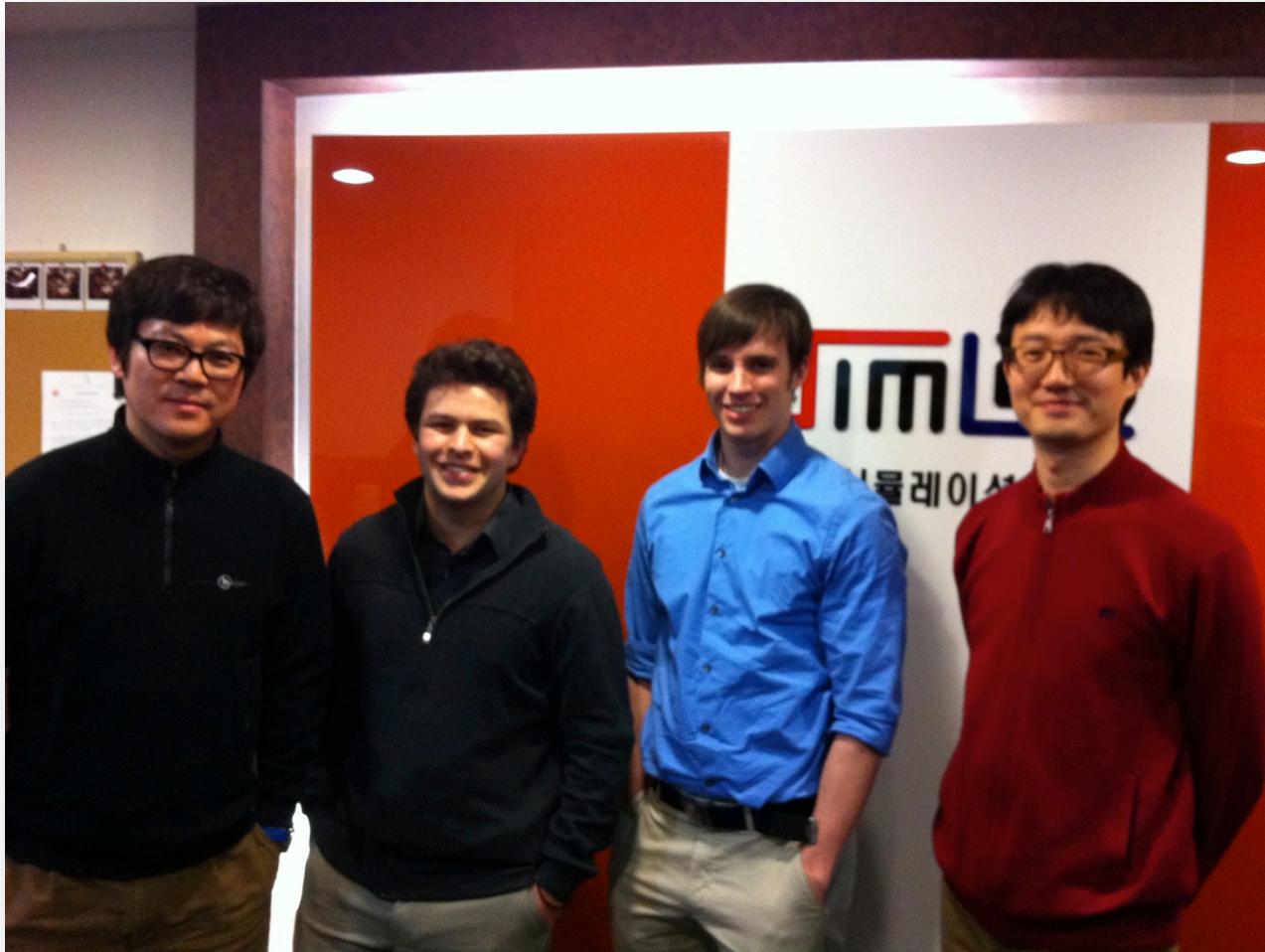


OpenRAVE  
RoboticsLab



# 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.

