MAS 961

*How To Make Something That Makes (almost) Anything*

# Active Elements
# 10 minutes Summary
# 2009/5/17

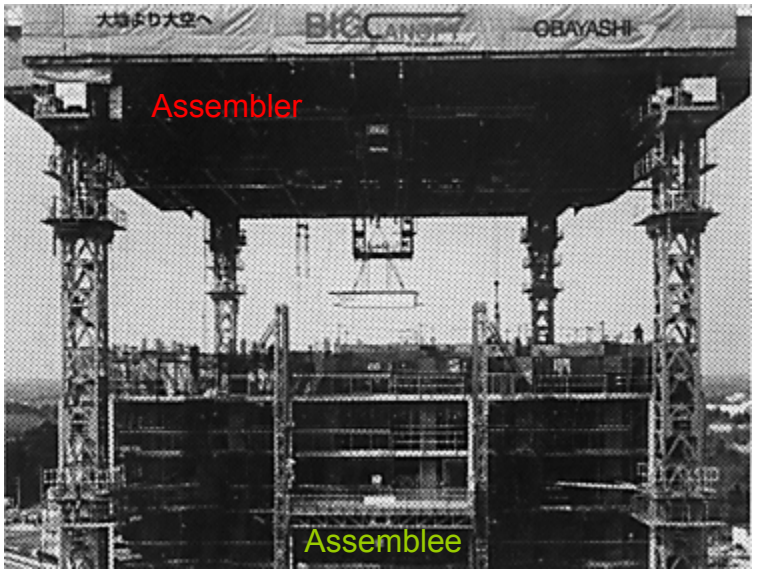
Taro Narahara

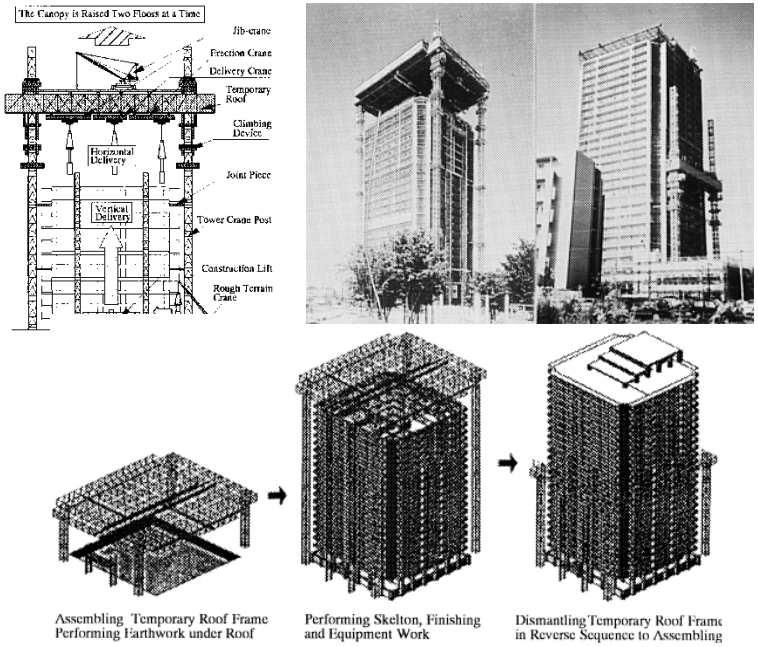
Doctoral Student

Harvard Graduate School of Design

# Precedents from Architecture:

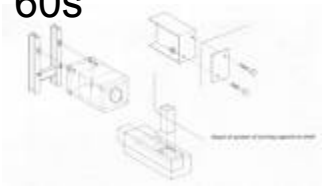
## Construction Automation in 80's

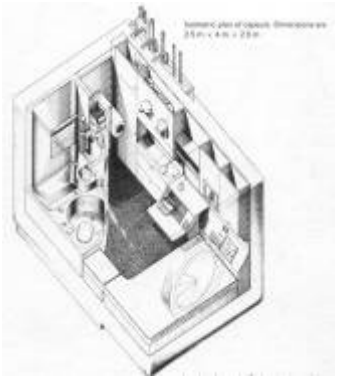


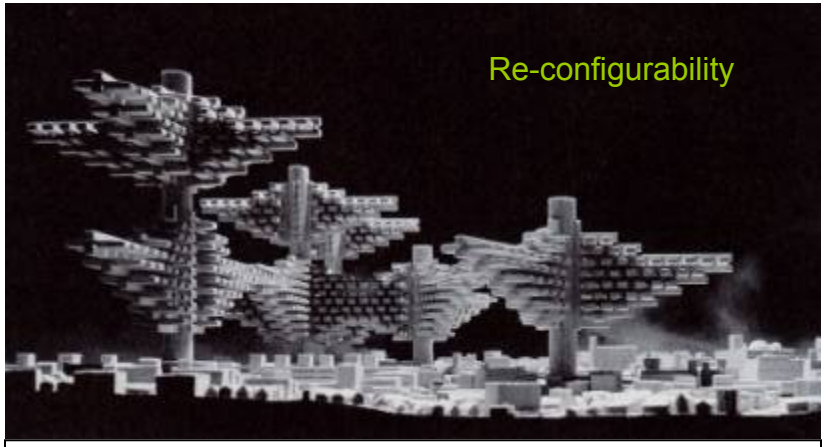




Shiokawa, Takashi. et al. (2000) **Automated construction system for high-rise reinforced concrete buildings**, Automation in Construction 9_2000.

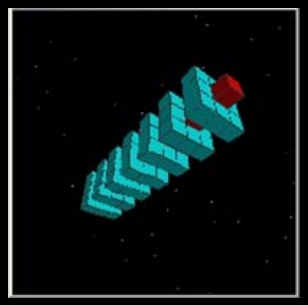
## Metabolists Movements in 60s



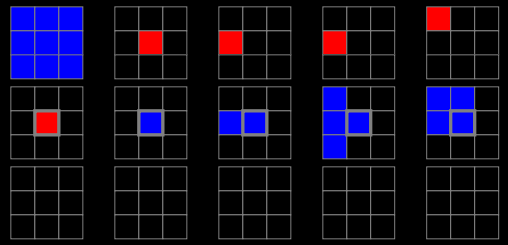




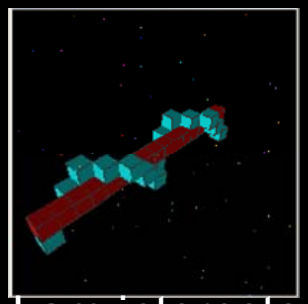

Nakagin Capsule Tower, Kurokawa, Tokyo, 1971.
The Cluster in the air, Isozaki, 1962.



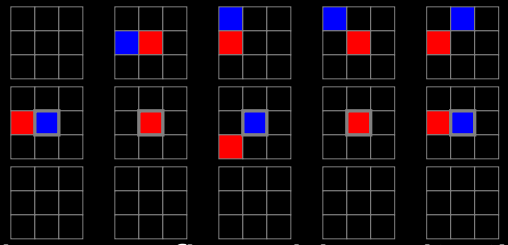
Rules produce Deterministic Structure



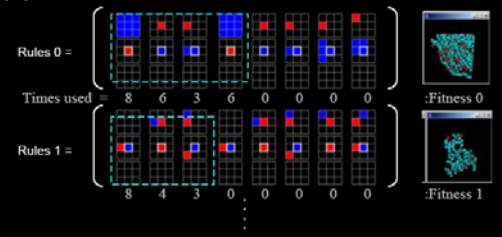
Allow 90degree Rotations



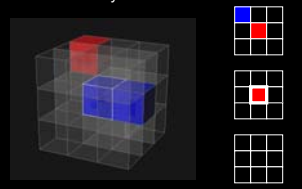
Rules produce Semi-Deterministic Structure

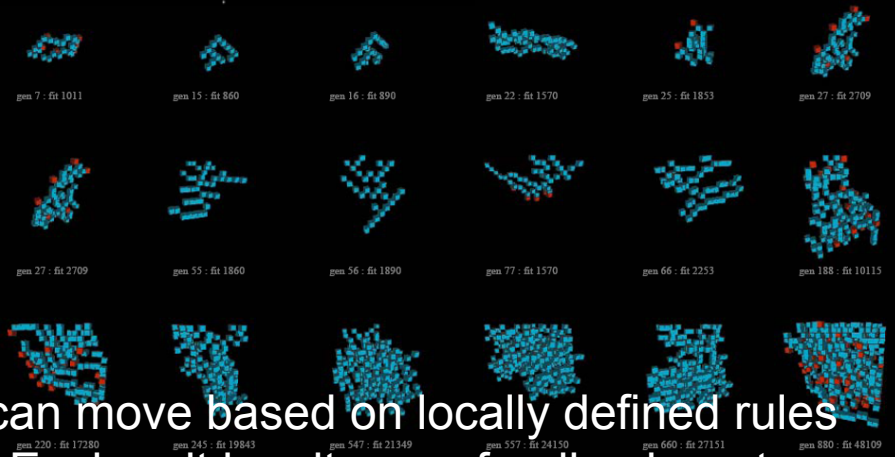


populations of rules

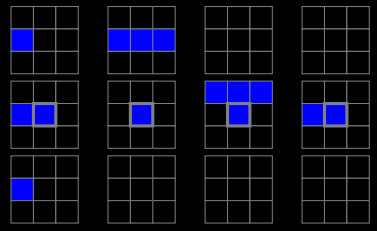



Locally defined Rules

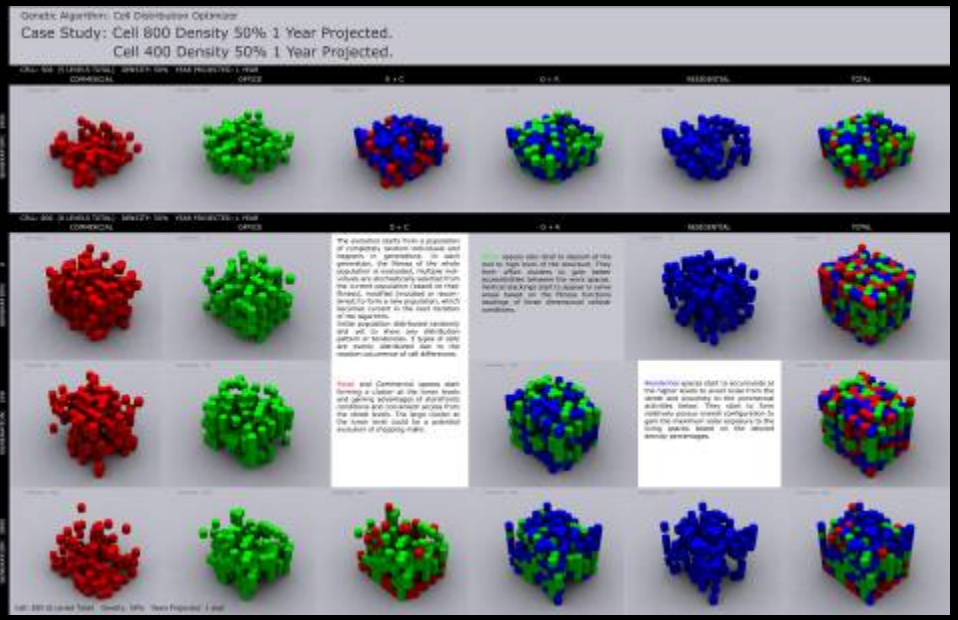




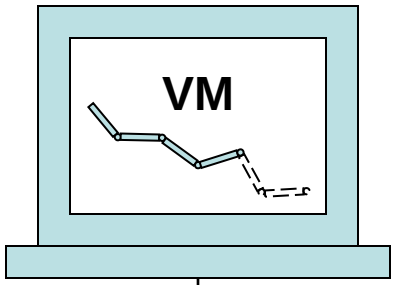

I am interested in reconfigurable units that can move based on locally defined rules rather than imposing top-down instructions. Each unit has its own feedback system.



Rules produce Non-Deterministic Structures



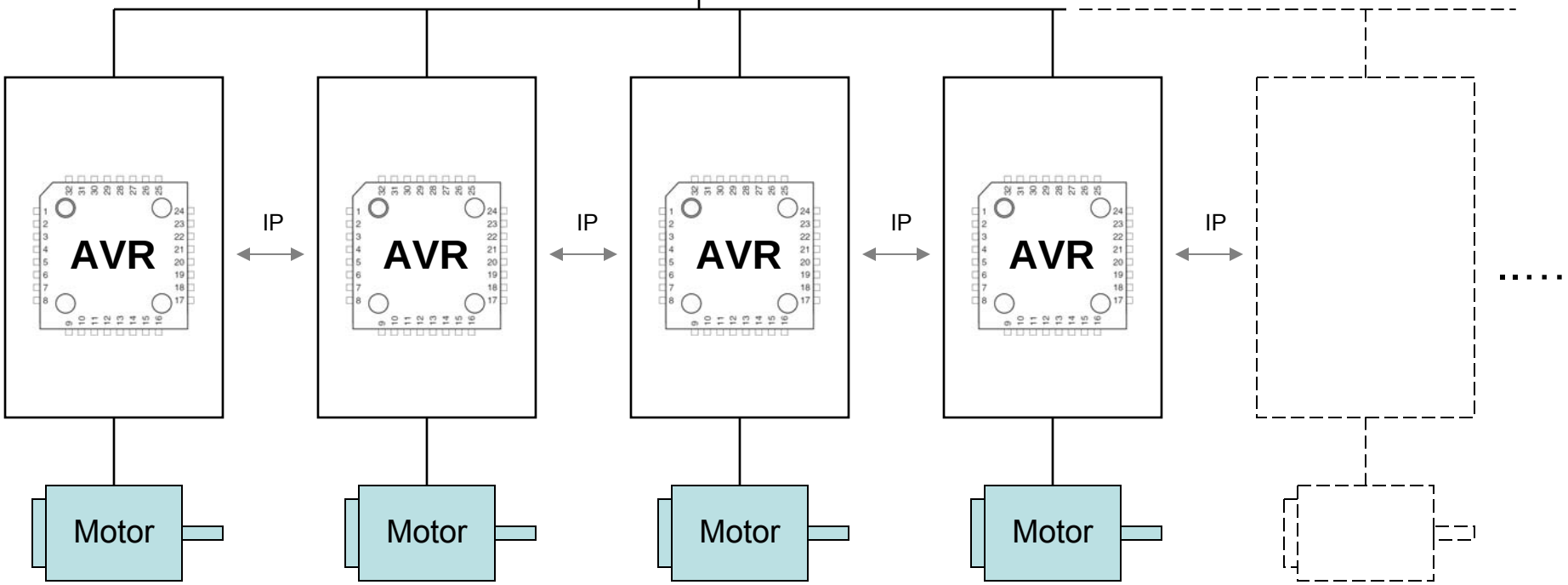
## Distributed System:



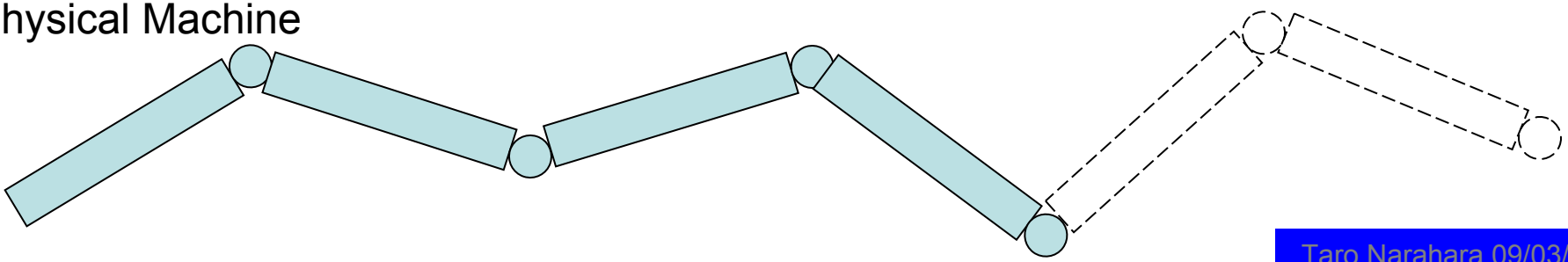

## System (Virtual Machine)

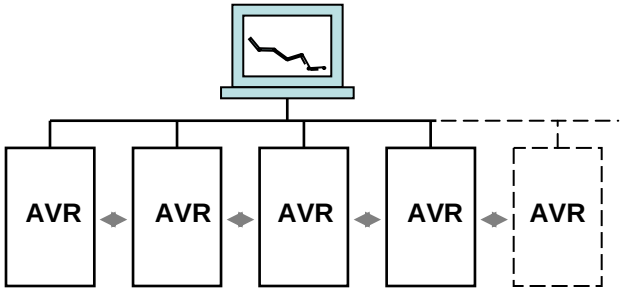
-Reconstruct **global topology** from local connectivity information from each components using Internet-0.

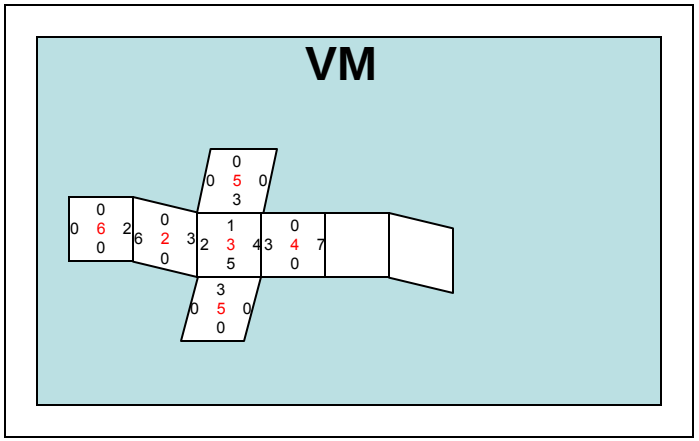
-Mental Picture inside the VM




## Physical Machine






**Physical**

Direc: TOP

IP : null;
Rotation: null

State: null
Speed: null
Duration: null

Direc: LEFT

IP address: null;
Rotation: null

State: null
Speed: null
Duration: null

Top
Left
AVR
Right
Down

Direc: RIGHT

IP address: 120;
Rotation: 0 degree

States: On/Off
Forward/Reverse
Speed: 0
Duration: 0 ms

Direc: DOWN

IP : null;
Rotation: null

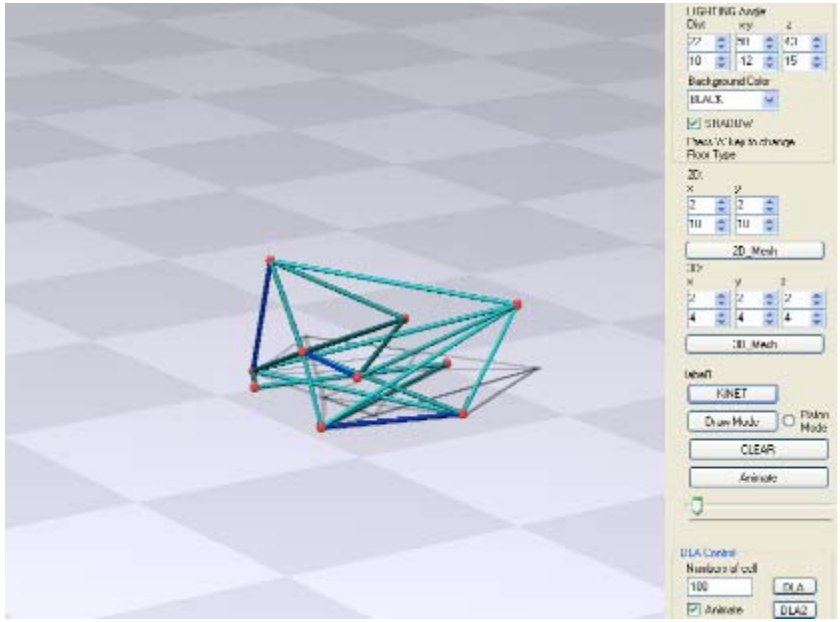
State: null
Speed: null
Duration: null

-Reconstruct Configuration

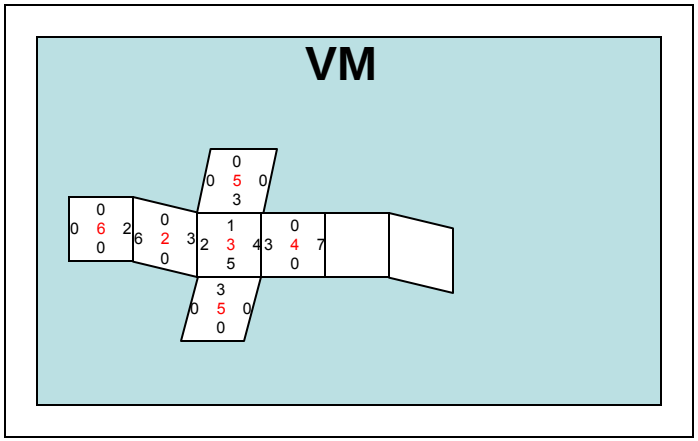
-Simulate Movements in VM

-Send Instructions to Robots

# Active Elements:



http://fab.cba.mit.edu/classes/MIT/961.09/people/Taro/assignment_2.wmv



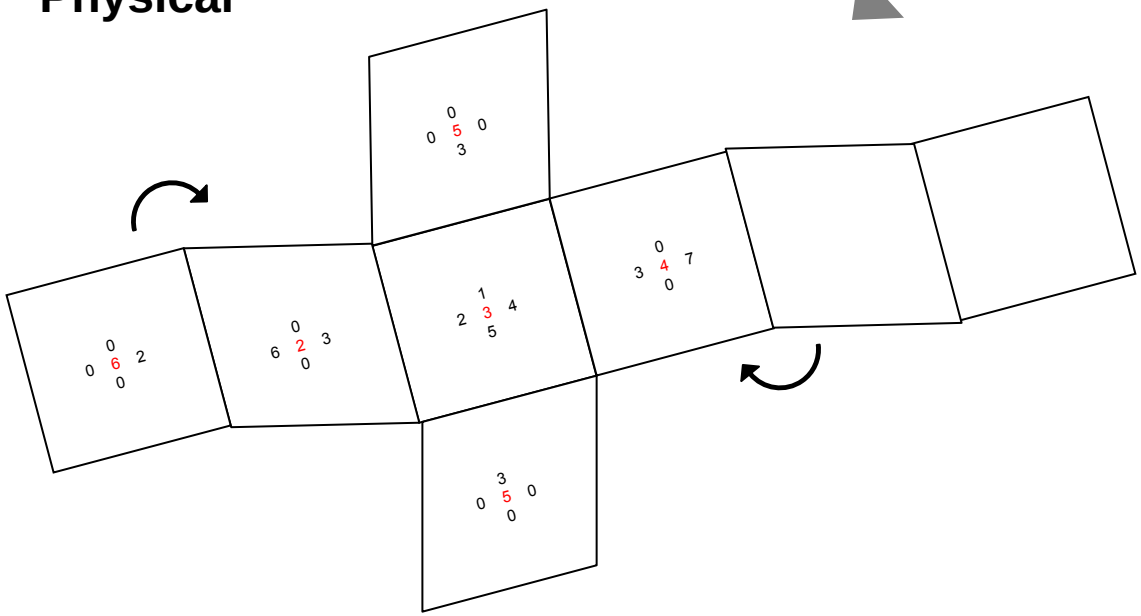

**Feedback**

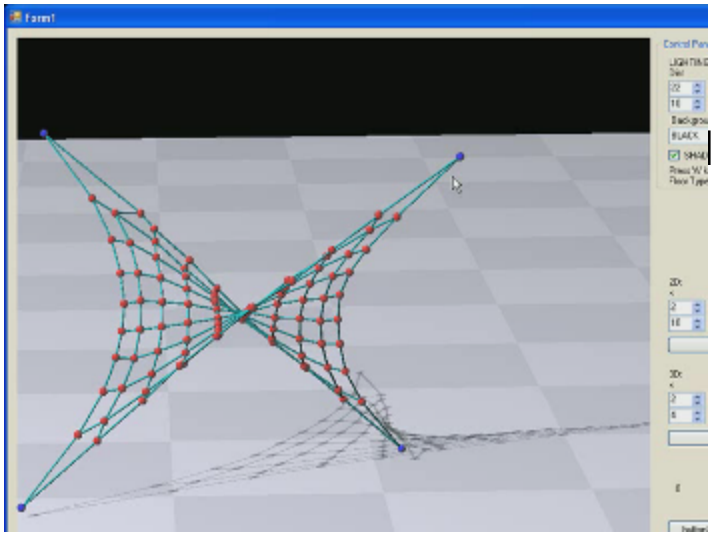
-Send Instructions to Robots

-Move Robots

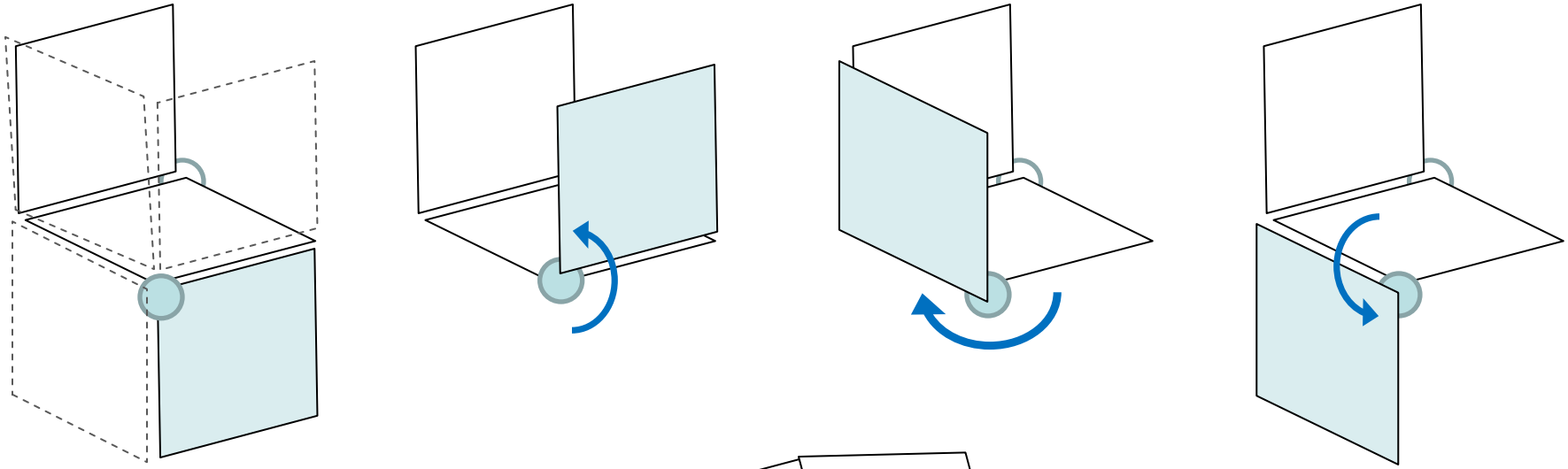
-Measure Performance
(Distance moved, Amount of Solar-exposure, etc.)

**Physical**

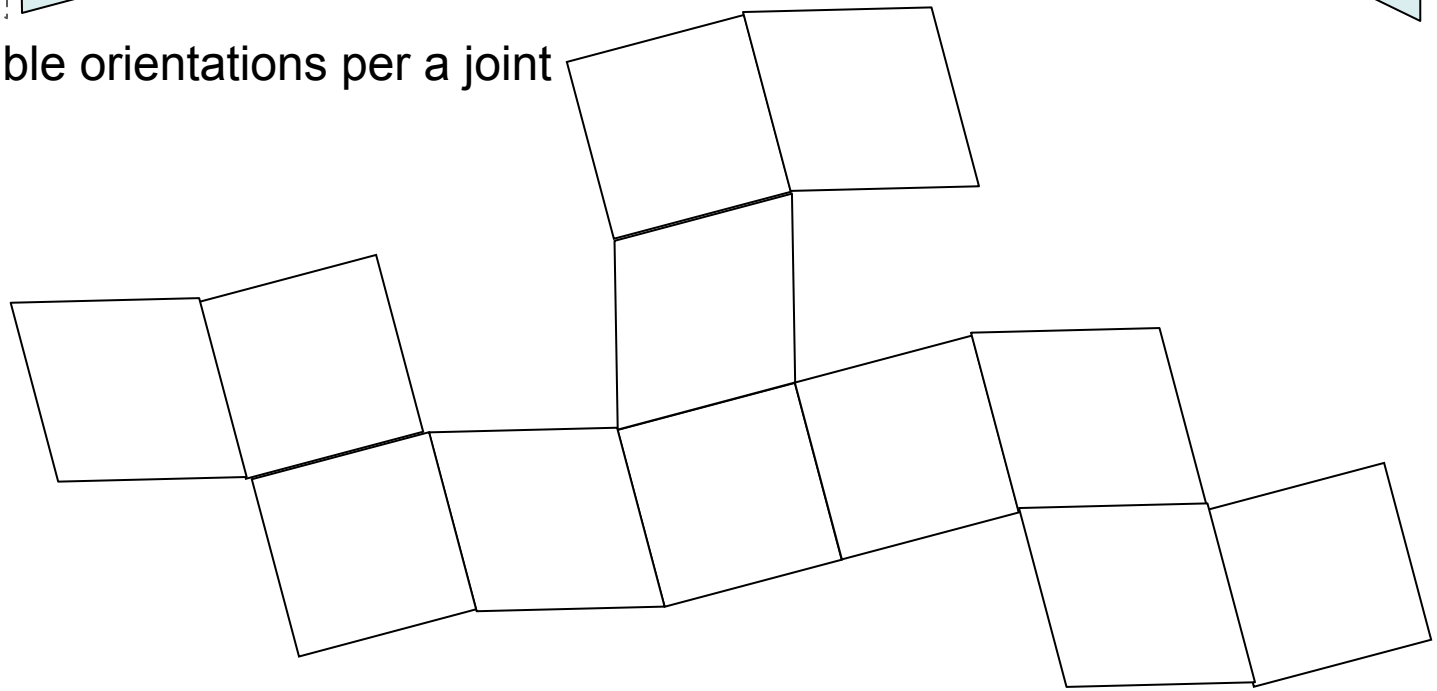




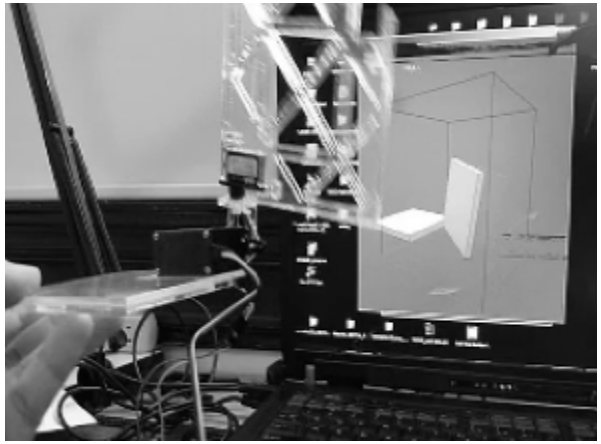
## 2-DOF at the JOINT

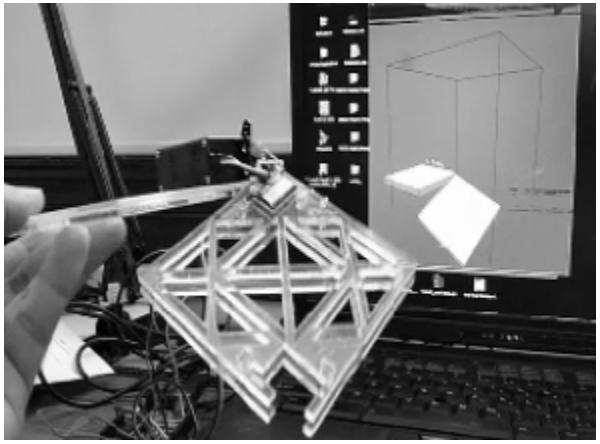


4 possible orientations per a joint

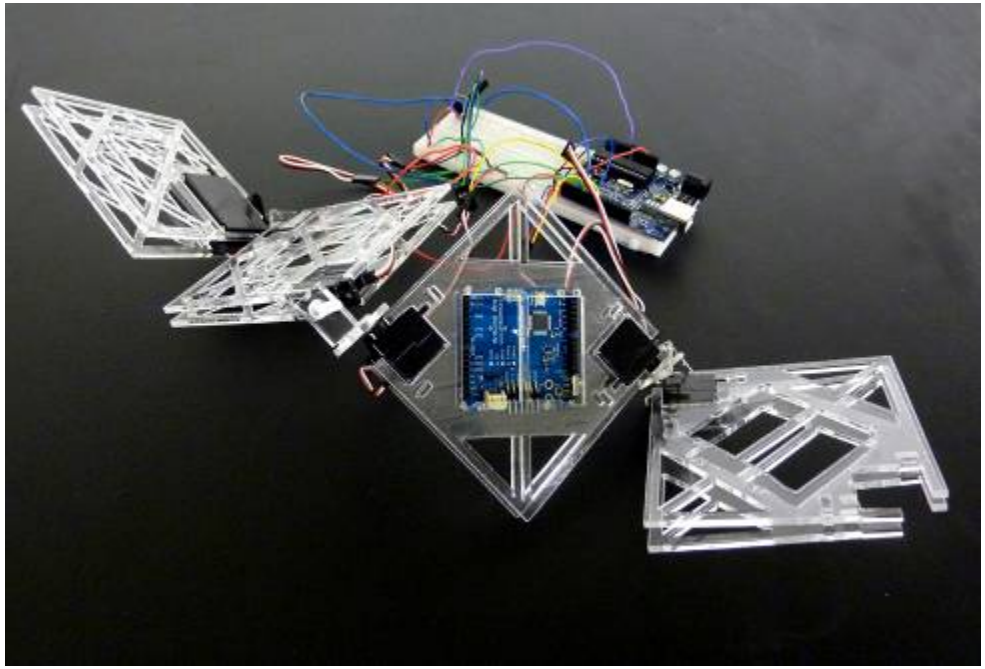


# User Interface design, Control, & Synchronization





http://fab.cba.mit.edu/classes/MIT/961.09/projects/active_elements/PS7_Taro.MOV



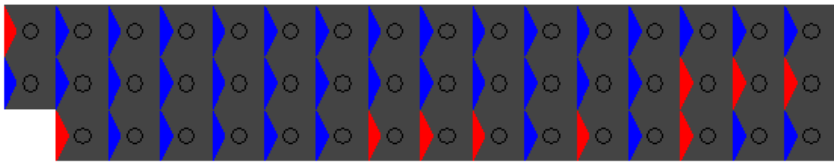
http://fab.cba.mit.edu/classes/MIT/961.09/projects/active_elements/PS9_Taro.mov

# ALA: Serial / Parallel Converter

Implementation Example:

Timing ID rotation
[ 0..10 1..10 0..11] … [ 1..11 1..00 0..10] …

00 := 0 degree
01 := 120 degree
10 :=-120 degree
:
:



Serial Inputs



Motor id:000
Motor id:001
Motor id:010
Motor id:011
Motor id:100

variable length control generator

# Serial / Parallel Converter

Outputs

1 0 0.

0 0 0

1 1 1

0 1 0

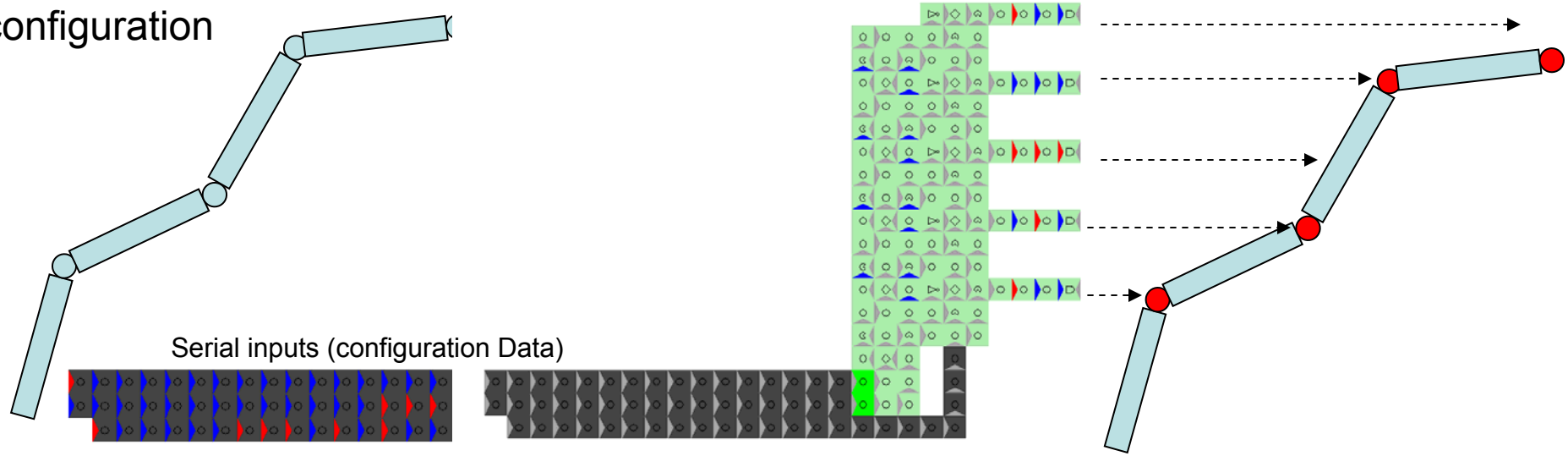
1 0 0

http://fab.cba.mit.edu/classes/MIT/961.09/projects/active_elements/RS10_ALATaro.gif

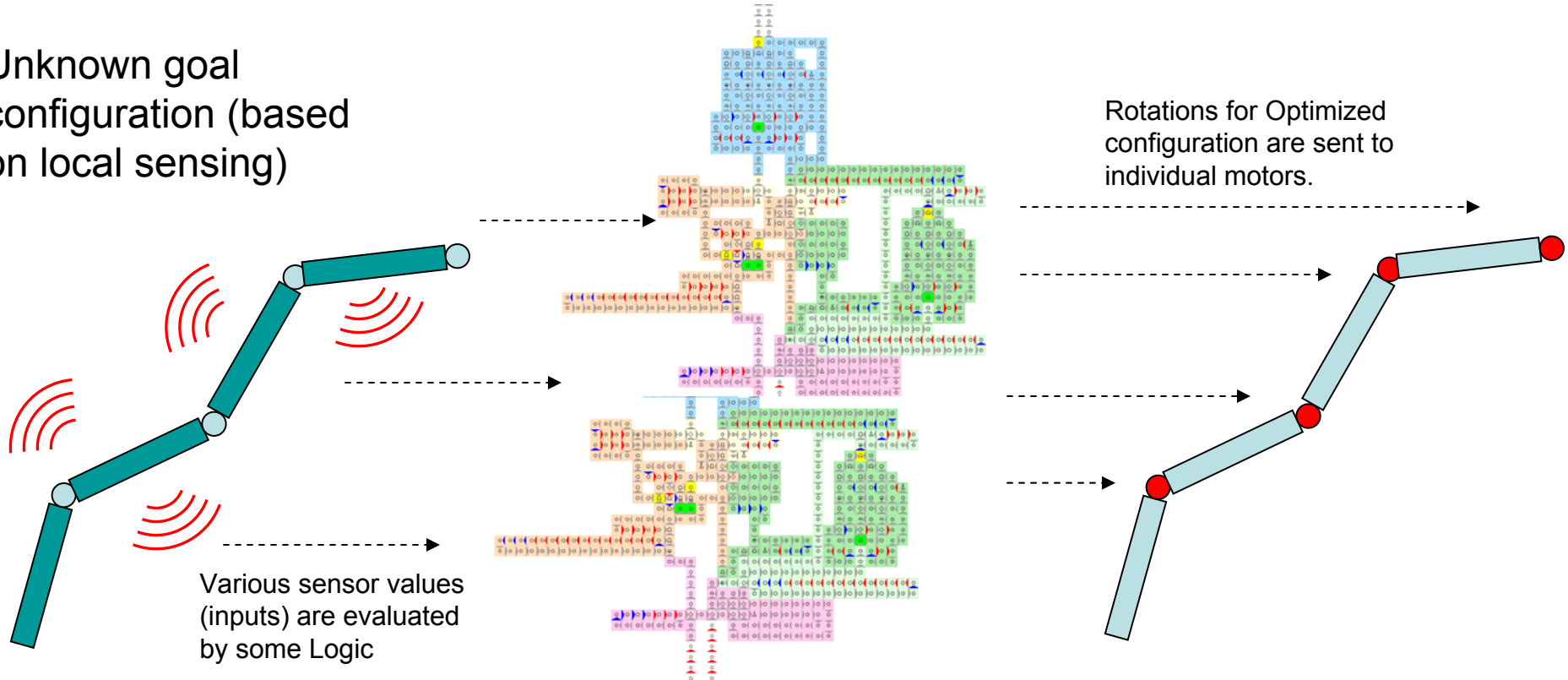
1 0 0. 0 0 0. 1 1 1. 0 1 0. 1 0 0.

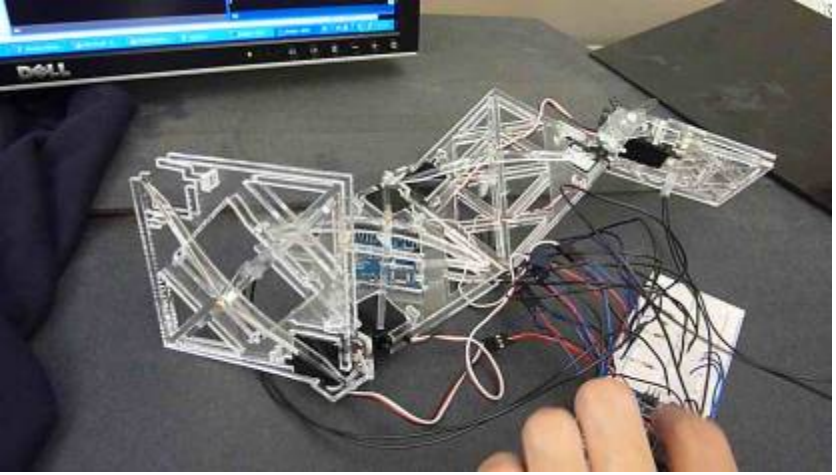
Inputs

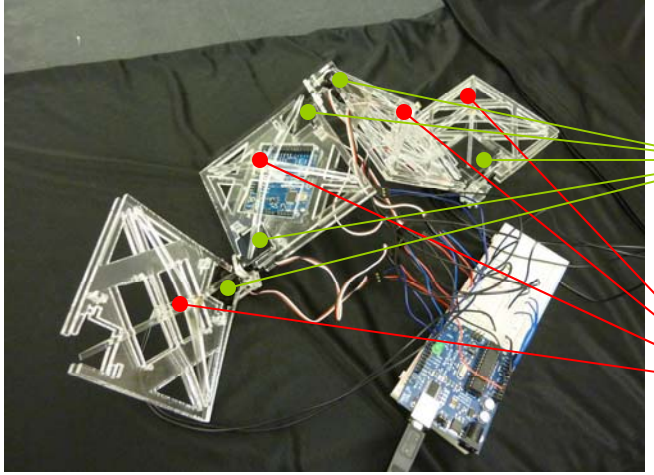
Pre-defined goal configuration




Unknown goal configuration (based on local sensing)





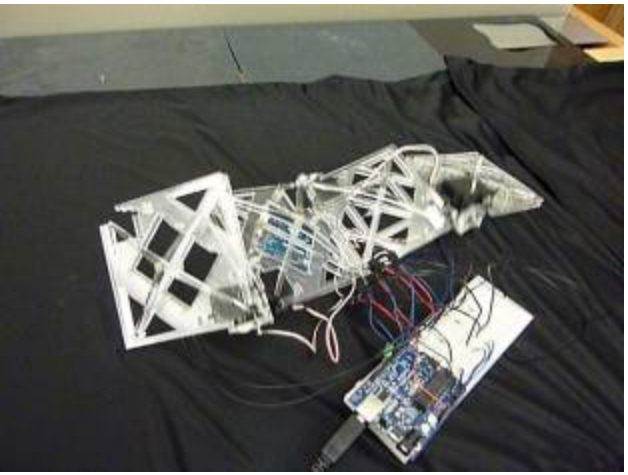


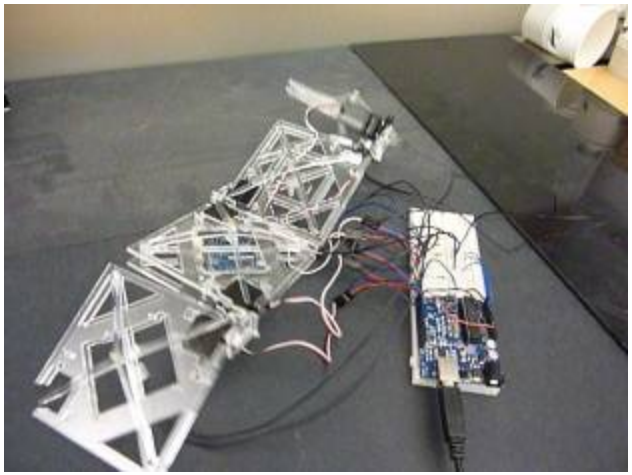
Optimize Rotations of

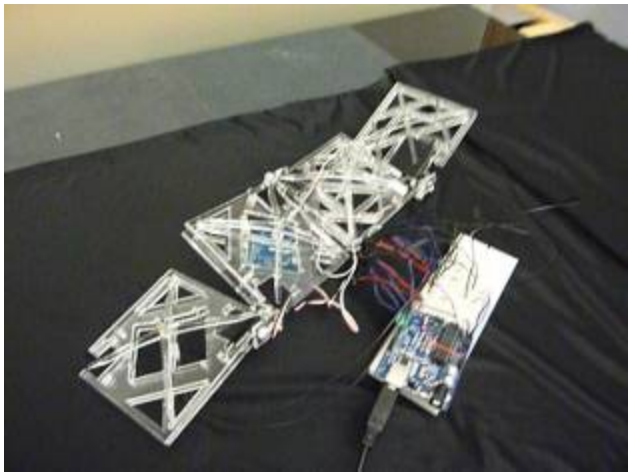
6 Motors

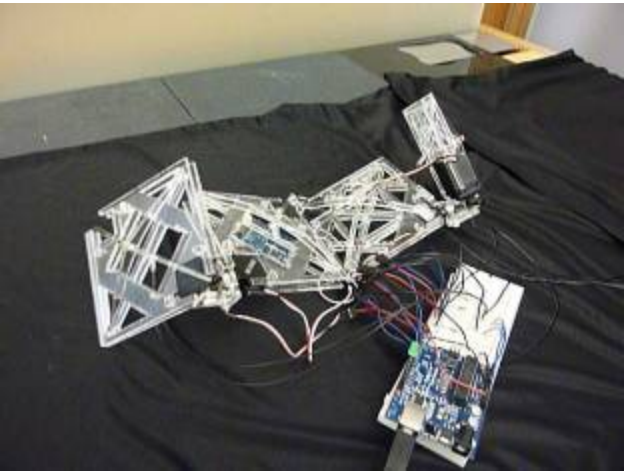
Based on Values from

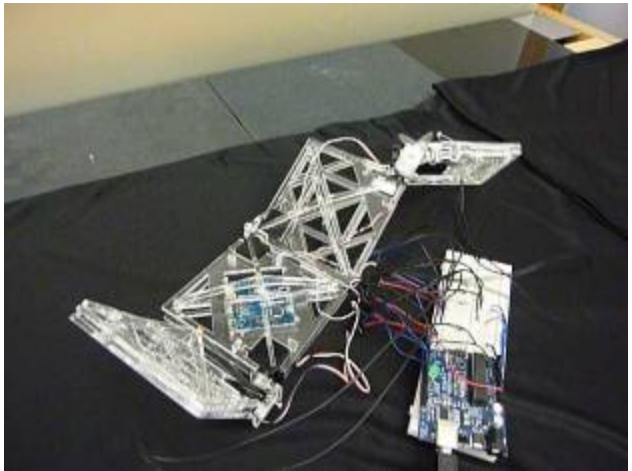
4 light Sensors

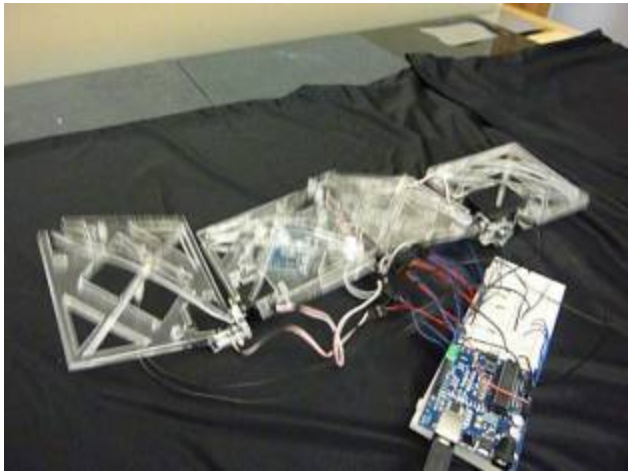








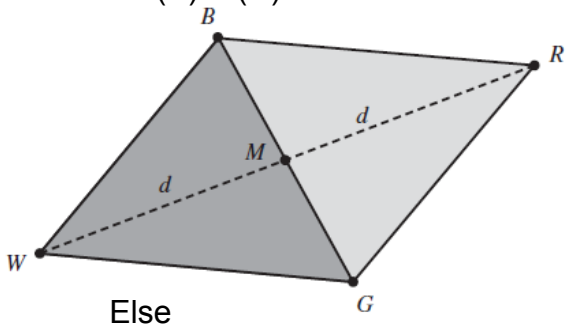




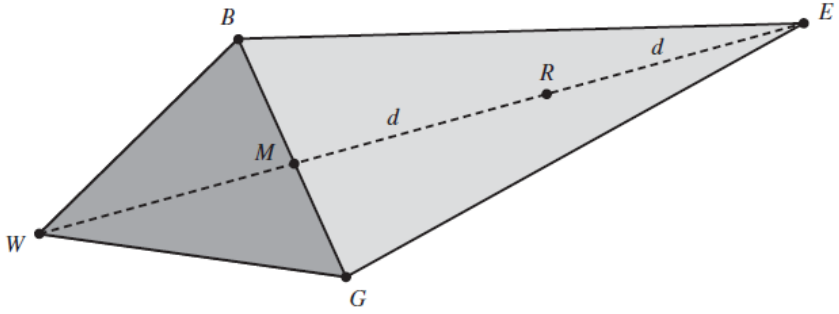
# Multi-dimensional Optimization: Nelder-Mead Method (amoeba)

6 dimensional space → find optimal lighting condition from average of 4 sensor (light diodes) values
optimize $F( \sum sensor\_i / n)$

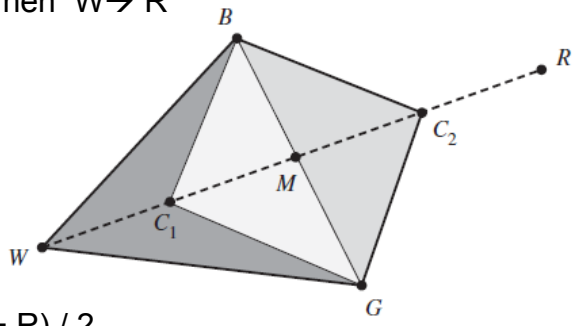
F(B)<F(G)<F(W) (Initial triangle)
R=(M - W) + M
If F(R)<F(G)
If F(B)<F(R) Then W→R



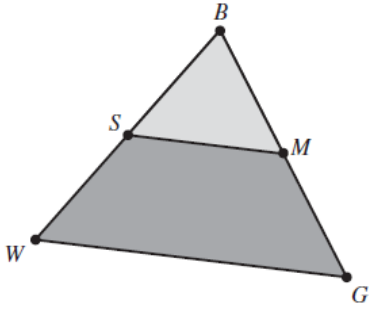

Else
E=(R - M) + R
If F(E)<F(B) Then W→ E
Else W→ R




Else
If F(R)<F(W) Then W→ R

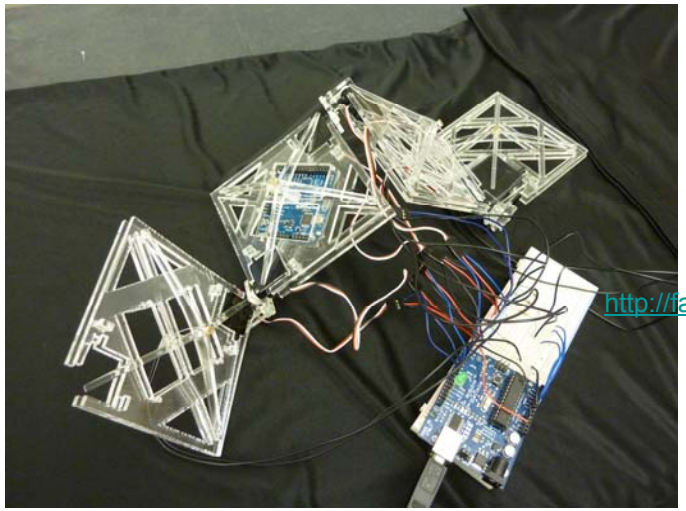



C=(M + R) / 2
If F(C)<F(W) Then W→ C
Else
S=(B + M) / 2 W→ S

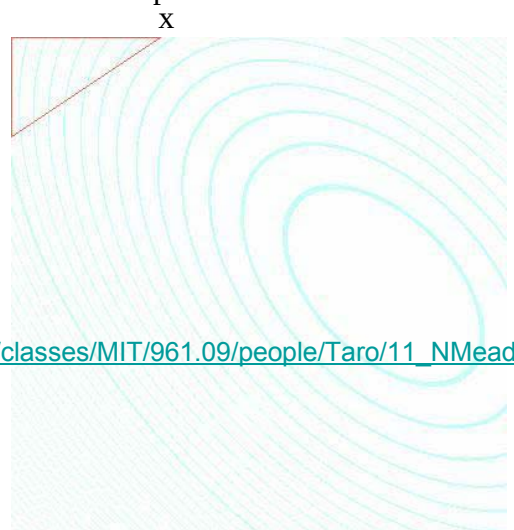


# Multi-dimensional Optimization:

Best: Servo1: 110˚
Servo2: 110°
Servo3: 135°
Servo4: 45°
Servo5: 45°
Servo6: 110°



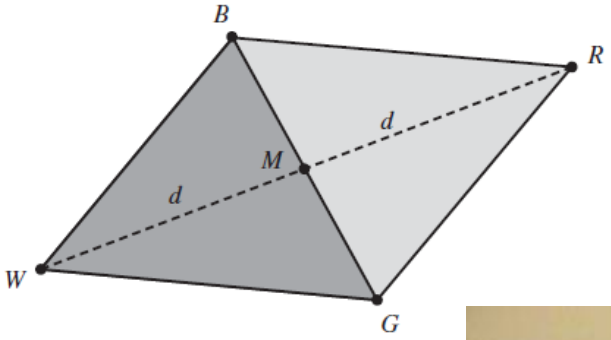
Ameoba walk process



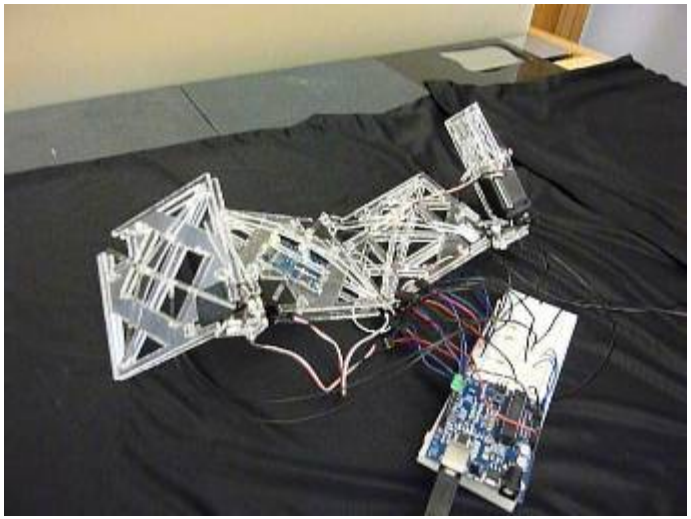

http://fab.cba.mit.edu/classes/MIT/961.09/people/Taro/11_NMead.gif

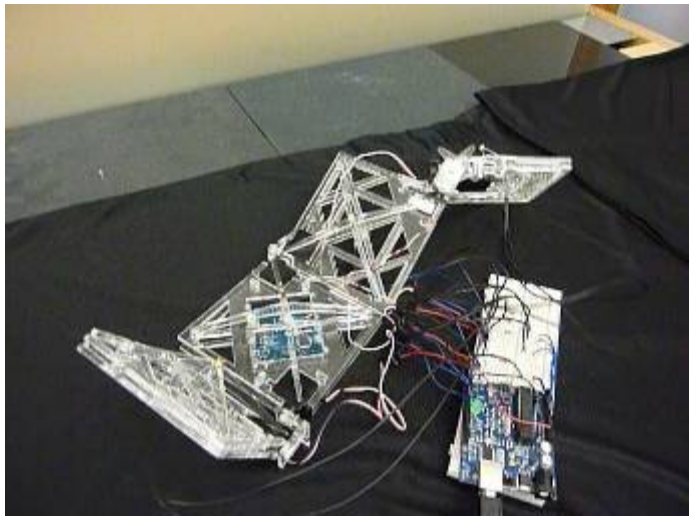
Example: $f(x, y) = x^2 - 4x + y^2 - y - xy$

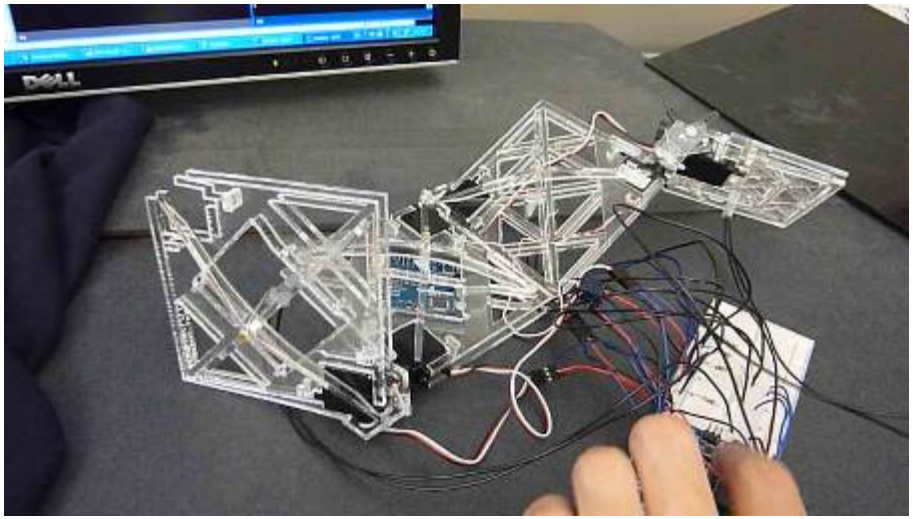
Worst: Servo1: 90˚
Servo2: 110°
Servo3: 90°
Servo4: 90°
Servo5: 110°
Servo6: 140°




Good: Servo1: 45˚
Servo2: 130°
Servo3: 180°
Servo4: 75°
Servo5: 95°
Servo6: 25°







## Random Search Example

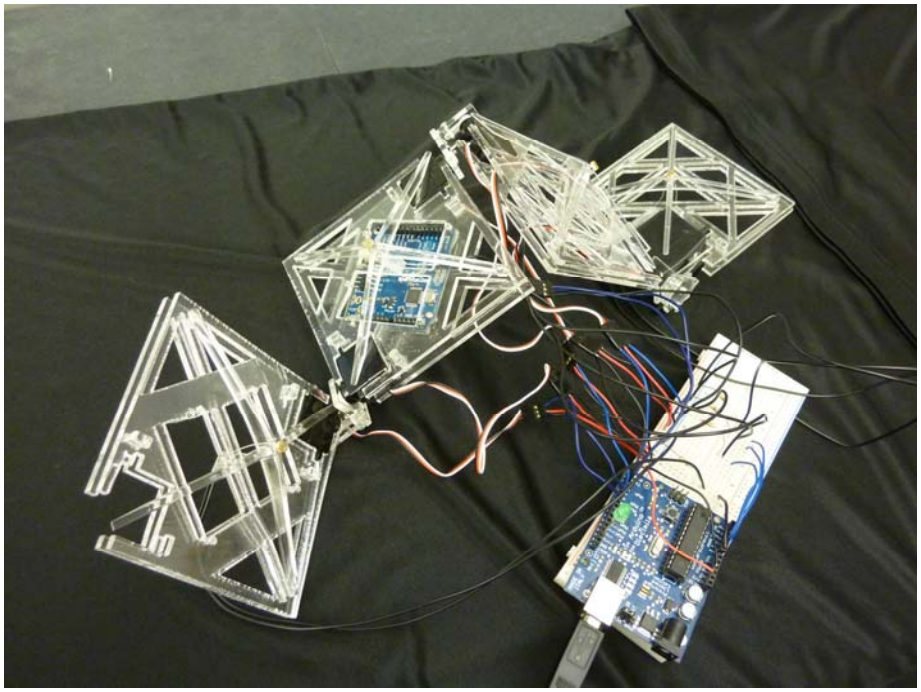
Foreach sensors

If (Sensor value improved)
Then continue to rotate in the same direction

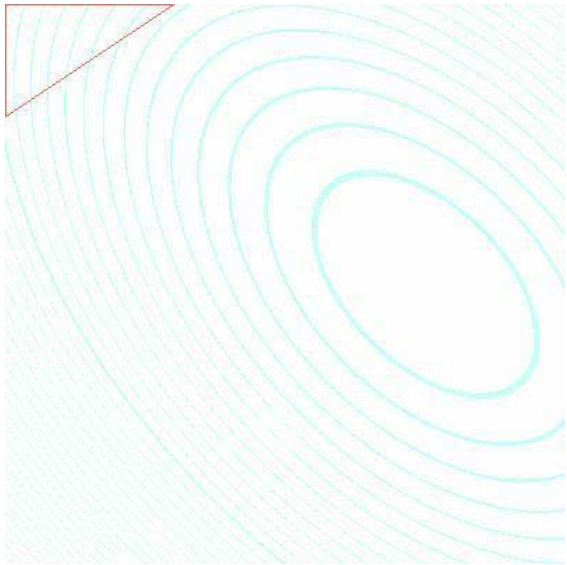
Else Turn the other way

| | | | |
|---|---|---|---|
| Sensor1 | Sensor2 | Sensor3 | Sensor4 |
| Motor1 | Motor2,3 | Motor4,5 | Motor6 |

http://fab.cba.mit.edu/classes/MIT/961.09/people/Taro/11_Taro_0.mov



## Nelder-Mead Method Implementation



http://fab.cba.mit.edu/classes/MIT/961.09/people/Taro/11_Taro1b_Li.mov
http://fab.cba.mit.edu/classes/MIT/961.09/people/Taro/11_Taro_2_Li.mov

## Possible Future Explorations:

-Look at other multidimensional optimization algorithms:
Levenberg-Marguardt, and etc.

- Further investigations on appropriate Geometries of elements for architectural and spatial purposes. (I only used simple square for my own experiments)

- Use architectural program types, occupancies, or social issues as a feedback values to optimize configurations. (not only environmental such as light value)