# Structure and Synthesis of Robot Motion

## Motion Synthesis under Sensorimotor Uncertainty I

Subramanian Ramamoorthy
School of Informatics

6 February, 2012

## How do we tell robots what to do?

## Exercise: What do we need to tell robots?

- You are given a few robots (automated car, robot arm that can bat a ball)
- We want these robots to perform dexterous skills
  - Drive around and park themselves, being (commonly) sensible
  - Be able to catch and throw a ball from many different conditions (baseball/cricket)
  - Watch out for other robots or people and act accordingly

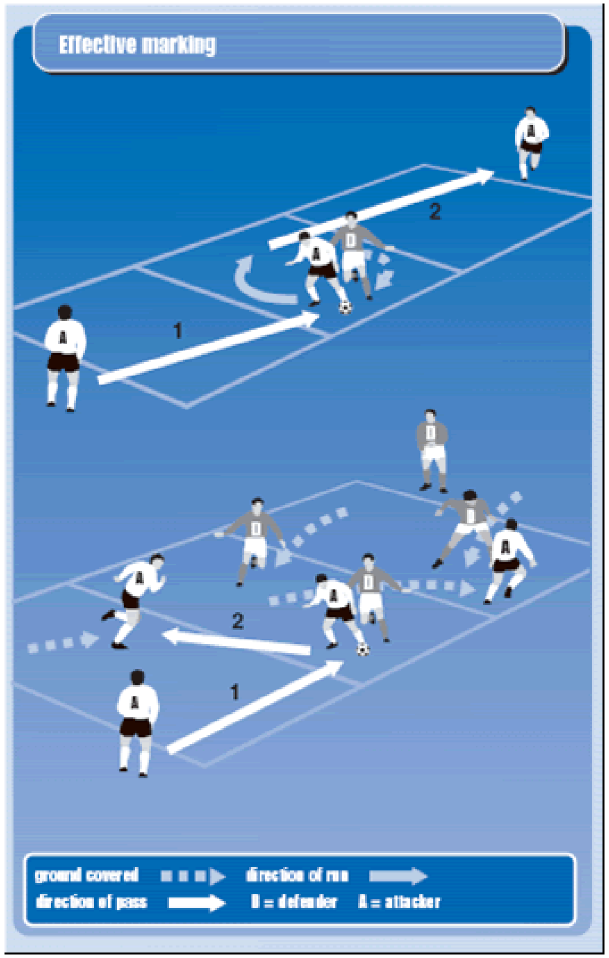
- Concisely describe the major elements you need to tell your robot about
  - What exactly is the task? How will you specify/encode the above tasks?
  - What types of models should it use? What information channels does it have?
  - How do the above interact with each other?

Discussion: How will you do this?

## Situational Awareness & Motion Synthesis

- Common sense is hard to automate!
- An aspect of common sense is our ability to take into account many different sources of information and make coarse judgments regarding actions
- What exactly does this mean for your motion synthesis procedures?

# Exercise: How will you tell a humanoid robot to mark another player?



## Basic Solution Strategy: Reactive Control

On the face of it, all of these tasks are not incompatible with the optimal control approach mentioned in an earlier lecture

- What aspects are ‘easy’ to encode that way? What is hard to encode?

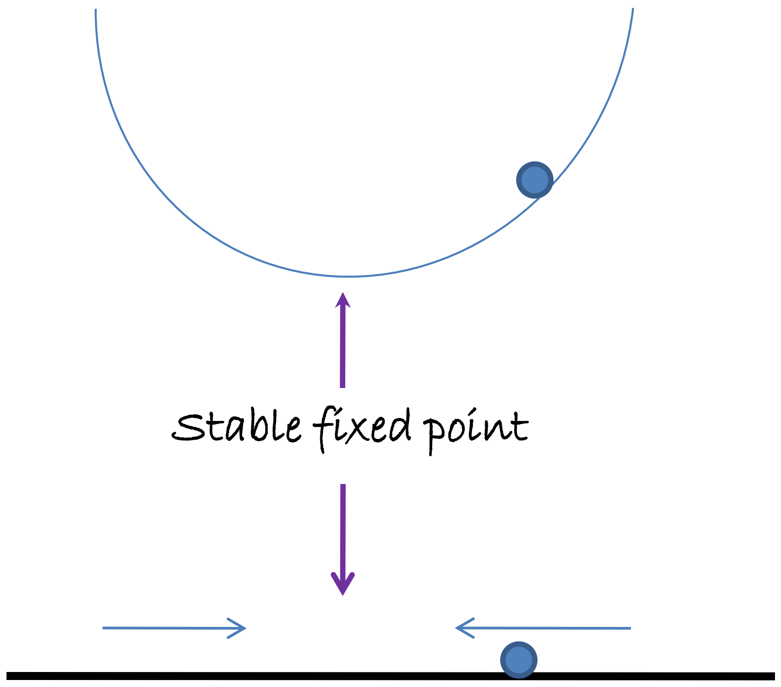
One general purpose strategy that people have obtained significant mileage out of – potential functions.

- In a sense, the value function in RL encodes something similar

# What is a Potential Function?

- Imagine a *1*-dim ball rolling within a flat bowl
- Where will it eventually end up after long time interval?
- What happens if you push the ball around with your finger?

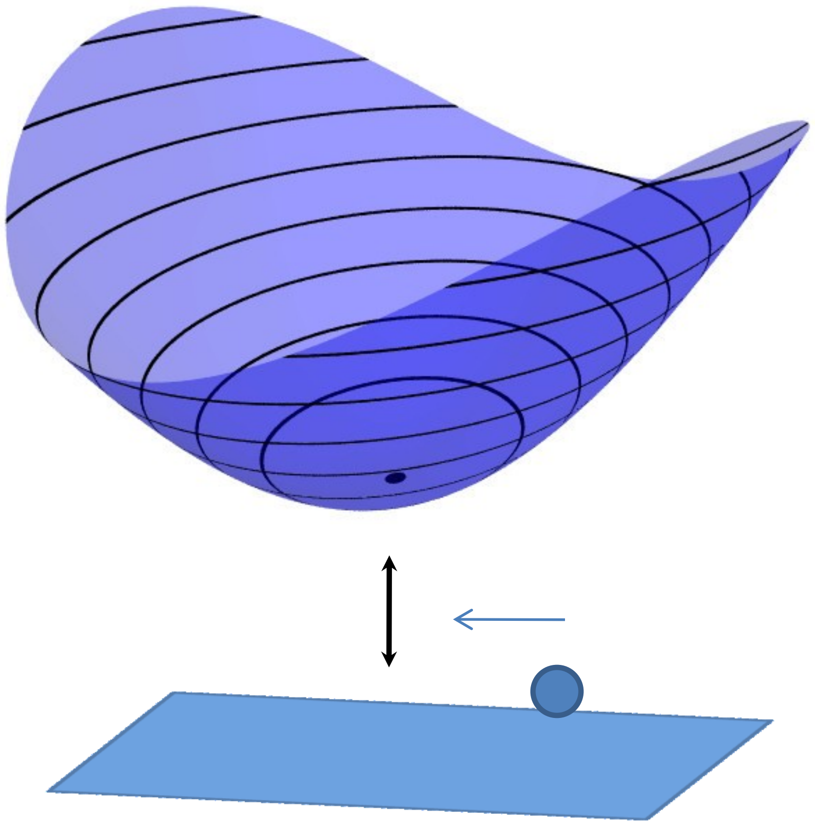
- If you "create" such a field on your *1*-dim flat world, where will the ball go?



# Higher-dim bowl

- You could play the same game in higher dimensions
- With contours that shrink down to a point, the ball will move in the direction that decreases a measure of height

- The effect on a *2*-dim workspace is that the ball will converge to a fixed point



# How to Encode a Bowl? Potential Function

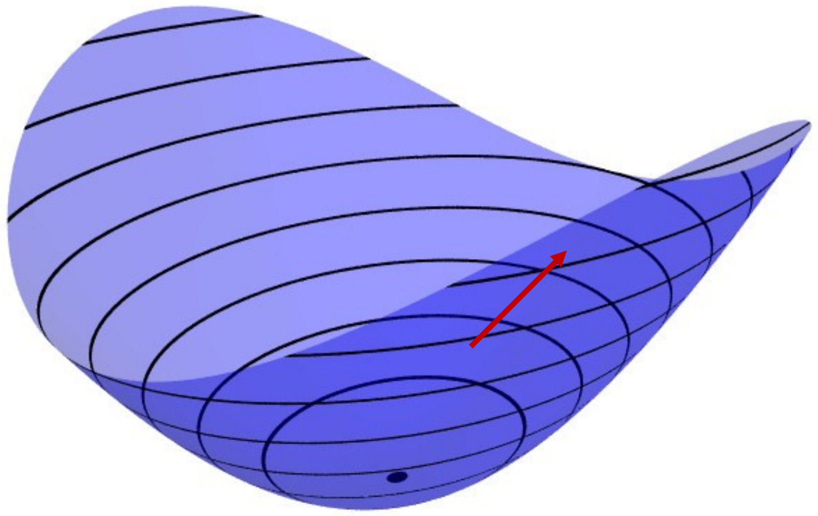
- Differentiable real-valued function,

$$U : \mathbb{R}^m \mapsto \mathbb{R}$$

- Treat the value as 'energy'
- Then, gradient is the vector,

$$\nabla U(q) = DU(q)' = [\frac{\partial U}{\partial q_1}(q), \ldots, \frac{\partial U}{\partial q_m}(q)]'$$

- The gradient points in the direction that locally maximally increases $U$

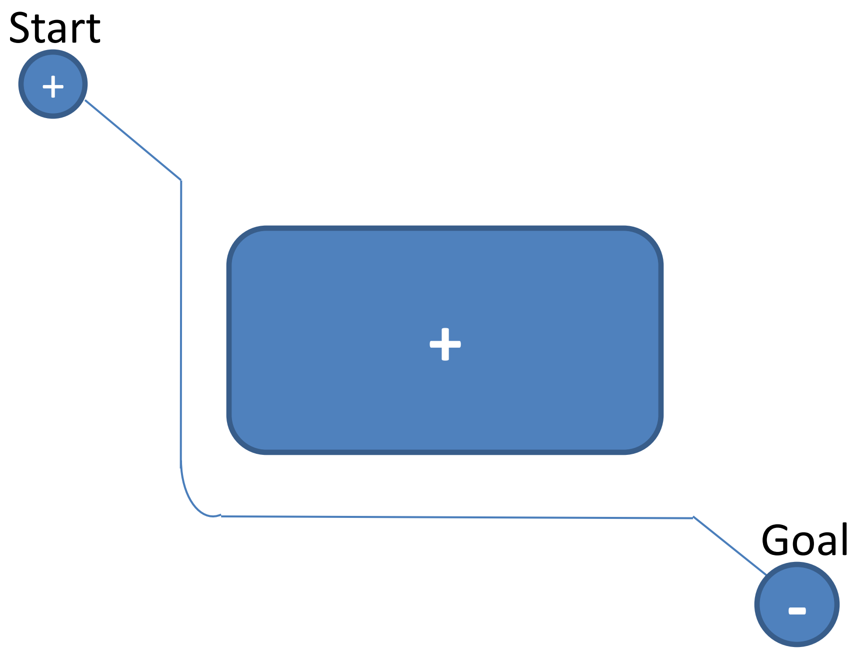


Property: Work done along a closed path is zero

# Point Robot with Potential Function

- Moves along a vector field induced by gradients
- We can think of gradient as force on a charged particle attracted to opposite charge
- This way, one can have attraction/repulsion for navigation purposes

<u>Note</u>: We are currently ignoring "real dynamics"



Path taken by a +ve charged particle

## Using a Potential Function

The robot can use a potential function for gradient descent,

$$\dot{c}(t) = -\nabla U(c(t))$$

The robot's motion would terminate at a critical point, $q^*$,

$$\nabla U(q^*) = 0$$

It is possible to check max/min/saddle point by looking at the Hessian,

$$\begin{pmatrix} \frac{\partial^2 U}{\partial q_1^2} & \cdots & \frac{\partial^2 U}{\partial q_1 \partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 U}{\partial q_1 \partial q_n} & \cdots & \frac{\partial^2 U}{\partial q_n^2} \end{pmatrix}$$

- Positive-definite Hessian: local minimum
- Negative-definite Hessian: local maximum

<u>Assumption</u>:
No flat patches in $U$
(Non-degenerate Hessian)

# Representing a Potential Function

Potential function can include attractive and repulsive terms,

$$U(q) = U_{att}(q) + U_{rep}(q)$$

*Attractive* Potential choices:

Conic: $U(q) = \zeta \| q - q_{goal} \|$

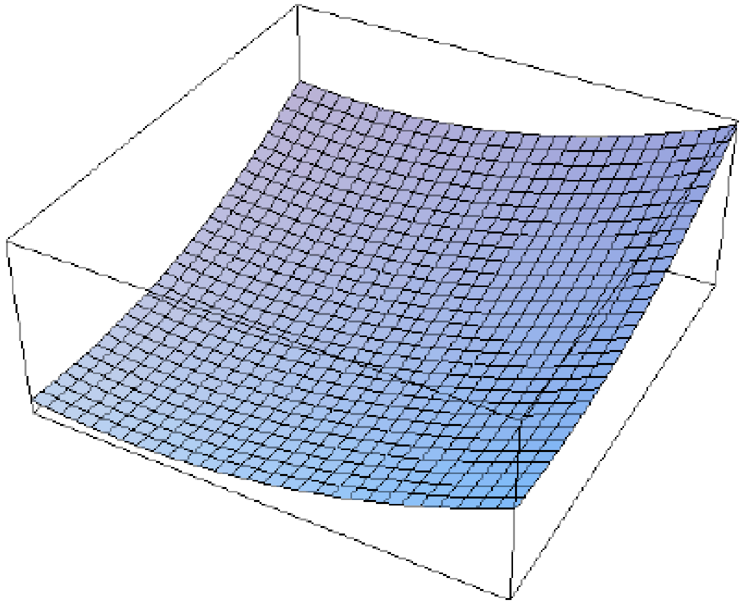
Quadratic: $U_{att}(q) = \frac{1}{2}\zeta d^2(q, q_{goal})$

*Repulsive* Potential choices:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^*})^2, & D(q) \leq Q^* \\ 0, & D(q) > Q^* \end{cases}$$

## Attractive Potential

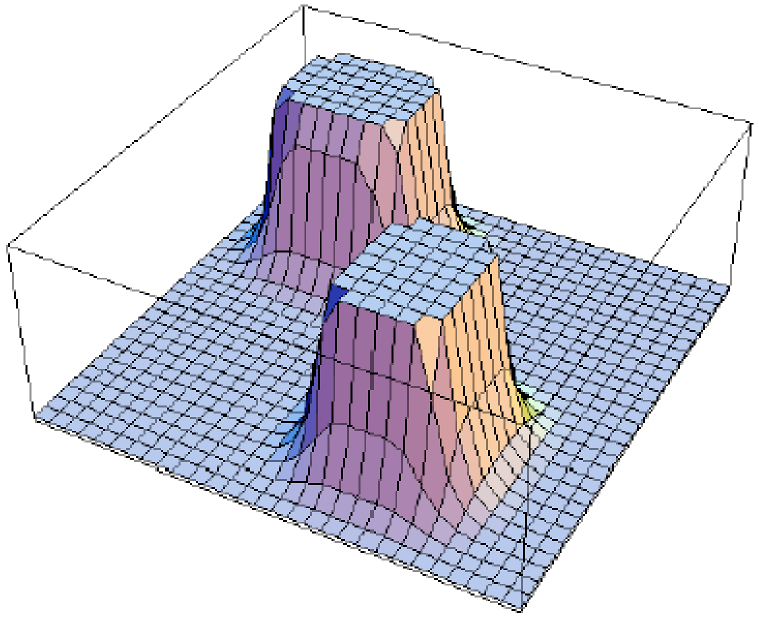
$$U_{\text{att}}(q) = \frac{1}{2} k \, \delta^2_{\text{goal}}(q)$$



$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(q)$$
$$= -k \, \delta_{\text{goal}}(q)$$

# Repulsive Potential

$U_{rep}(q) = 1/2\ \eta\ (1/D(q) - 1/Q^*)^2,\ D(q) \leq Q^*$, else 0

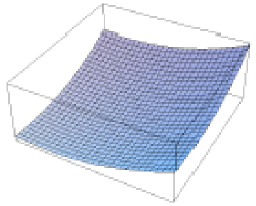


For convex obstacles:
Potentials can be
superimposed (summed)

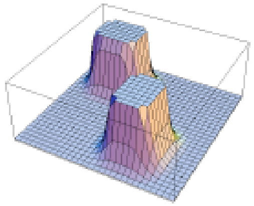
## Combined Potential Field

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

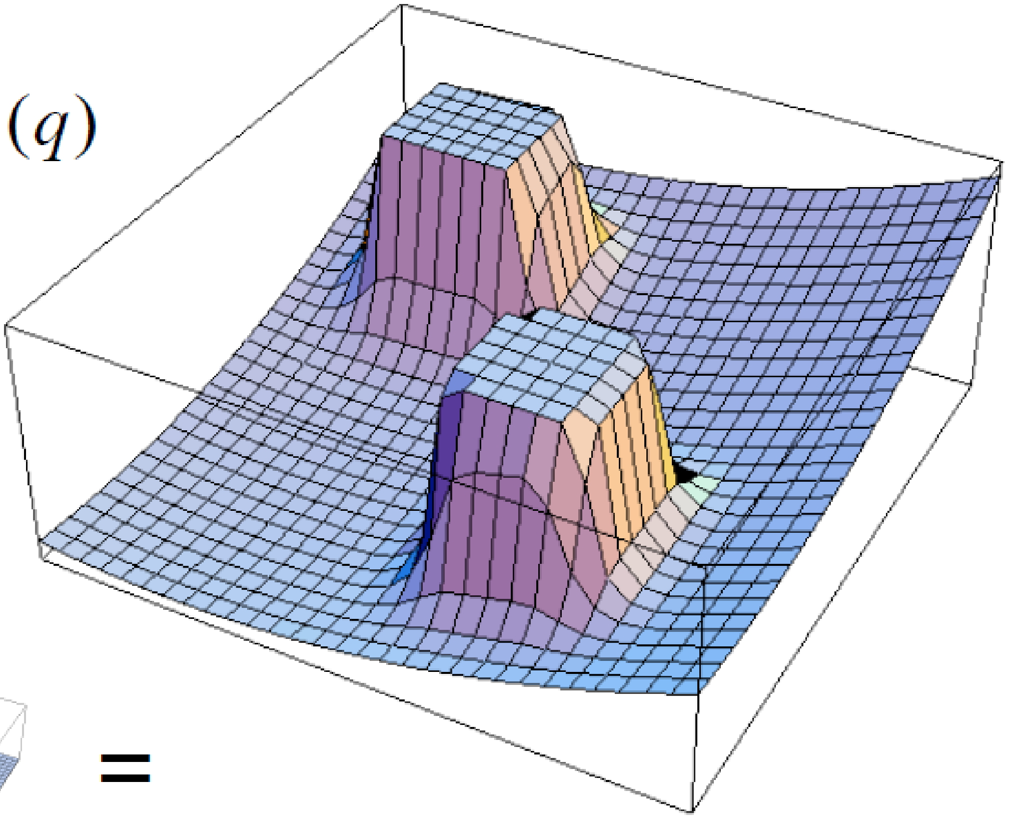
$$F(q) = -\nabla U(q)$$



+



=



# On-line Motion Planning with PF

**Input**: Function, $\nabla U(q)$
**Output**: Sequence $[q(0), q(1), ...q(i)]$

```
1. $q(0) = q_{start}$
2. i = 0
3. while $\nabla U(q(i)) \neq 0$
4.     $q(i+1) = q(i) + \alpha(i)\nabla U(q(i))$
5.     $i = i + 1$
6. end while
```

Many possible improvements,
draw on numerical optimization methods

## How do we Know PF will 'Work' ?

- It is possible to use ideas from dynamical systems theory
- One approach: can you find a Lyapunov function that is monotonically decreasing?
- An instance of stability over time,

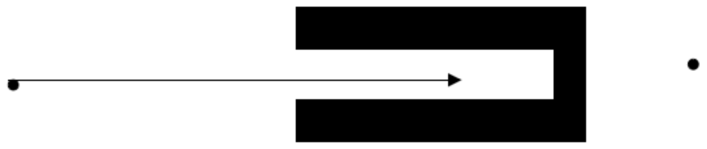
$$L = \sum_i \left( \frac{1}{2} \mathbf{v}_i^2 + U^S(\mathbf{x}_i) \right)$$

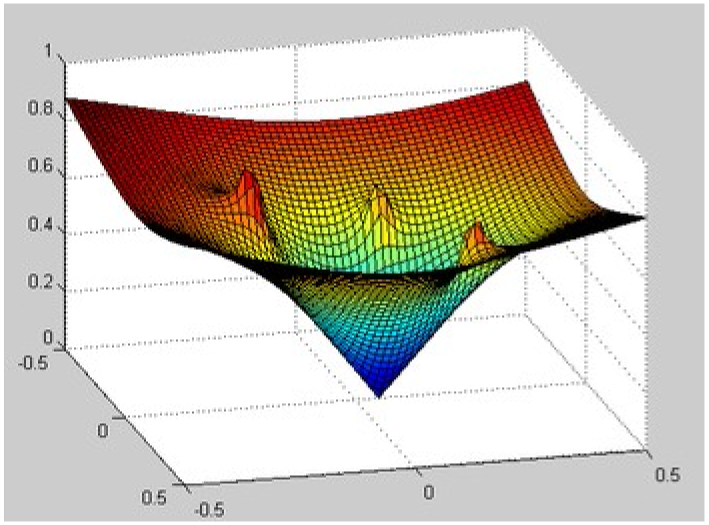
$\Rightarrow$ 'Stability'

$$\frac{dL}{dt} = -\sigma \sum_i \mathbf{v}_i^2 \leq 0$$

# Problem with Potential Function Approach

Local Minima!





As we create potential surfaces by superposition, it is not always easy to ensure that there are no spurious minima that are not at the goal.

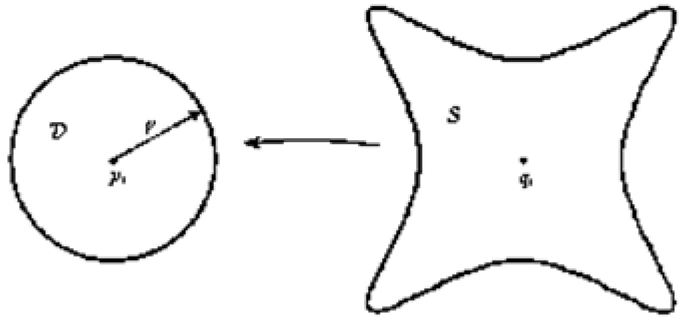
## Navigation Functions

- A function $\phi$: $Q_{free} \rightarrow [0,1]$ is called a *navigation function if it*
  - is smooth (or at least $C^2$)
  - has a unique minimum at $q_{goal}$
  - is uniformly maximal on the boundary of free space
  - is Morse
- A function is Morse if every critical point (a point where the gradient is zero) is isolated.

# Key Conceptual Move

- Define provably correct strategy for a canonical "sphere" world
- Define a ***diffeomorphism*** that maps to the actual workspace
- Make sure mapping preserves the correctness property

*Solve the basic problem here*



*"Lift" the solution to more complex space*

## Navigation Function in Sphere World

In sphere world, define obstacle distance functions of the form,

$$\beta_0(q) = -d^2(q, q_0) + r_0^2$$

$$\beta_i(q) = -d^2(q, q_i) - r_i^2$$

Instead of working with closest distance, define

$$\beta(q) = \prod \beta_i(q)$$

This allows you to define navigation functions of the form,

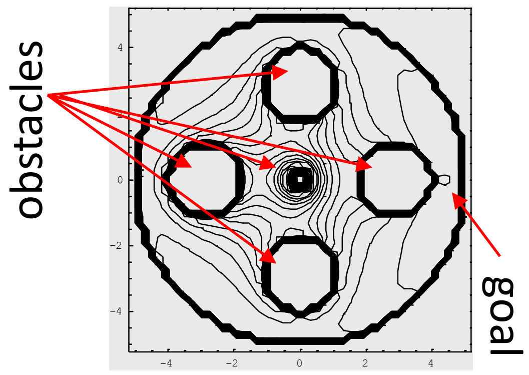
$$\phi(q) = \frac{d^2(q, q_{goal}}{[(d(q, q_{goal})^{2\kappa} + \beta(q)]^{\frac{1}{\kappa}}}$$

which is guaranteed to reach $q_{goal}$ for a large $\kappa$.

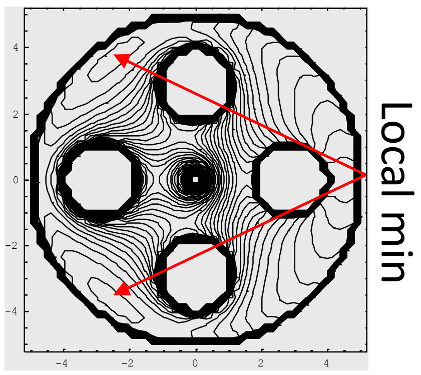
# Navigation Function for the Sphere World

$$\varphi_k(\mathbf{q}) = \left(\rho_k \circ \sigma_1 \circ \frac{\gamma_k}{\beta}\right)(\mathbf{q}) = \frac{\|\mathbf{q}-\mathbf{q}_\mathrm{d}\|^2}{\left(\|\mathbf{q}-\mathbf{q}_\mathrm{d}\|^{2k} + \beta(\mathbf{q})\right)^{1/k}}$$

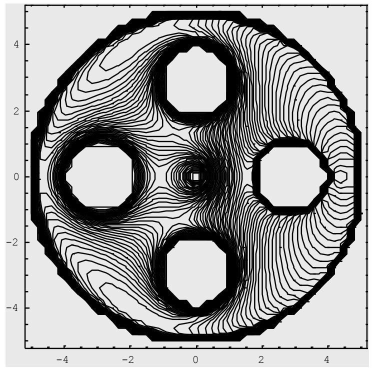
For sufficiently large k, this is a navigation function



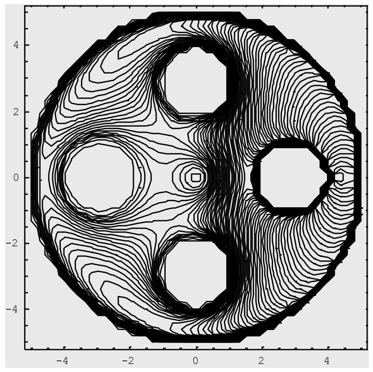
k=3



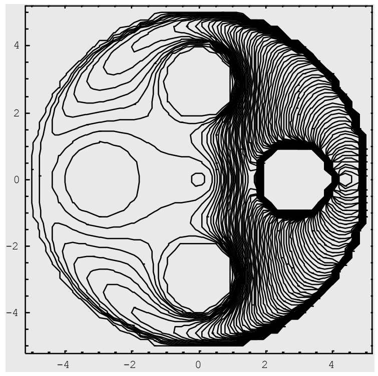
k=4



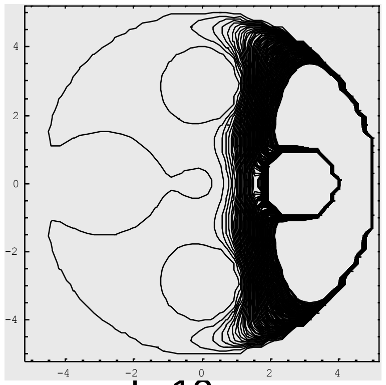
k=6



k=7

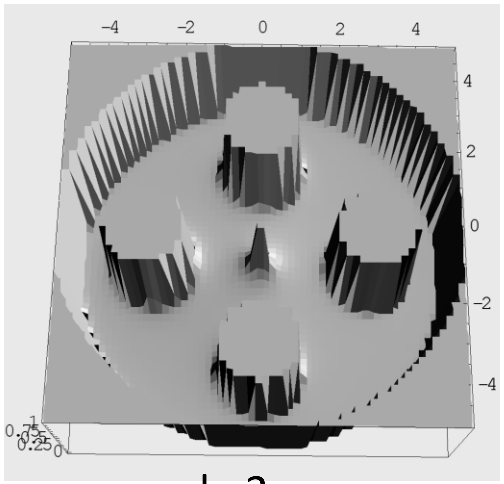


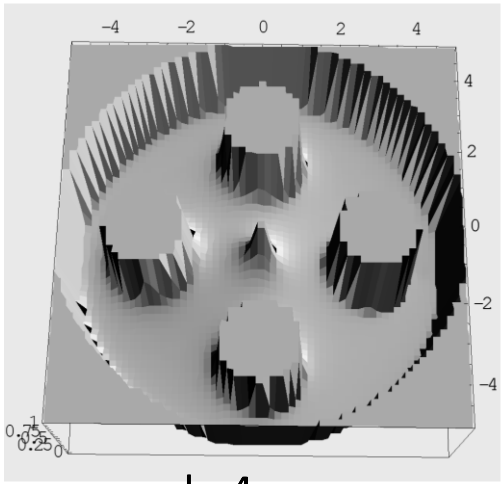
k=8

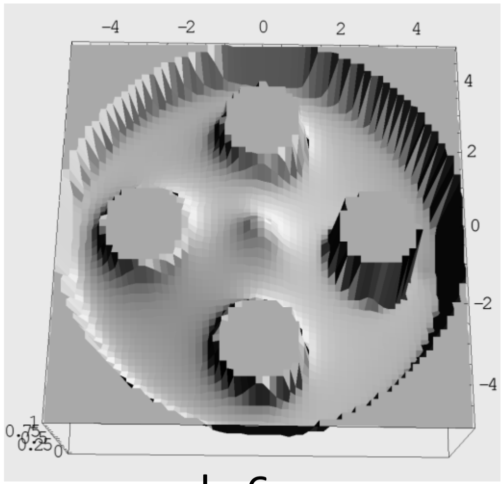


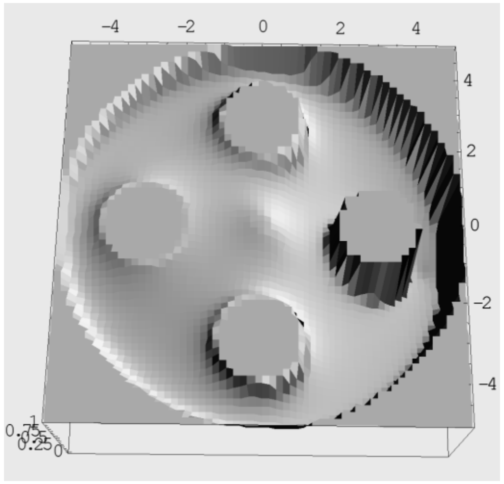
k=10

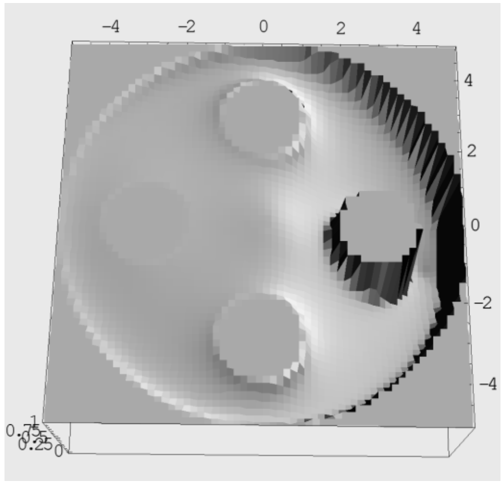
# Navigation Function at different k

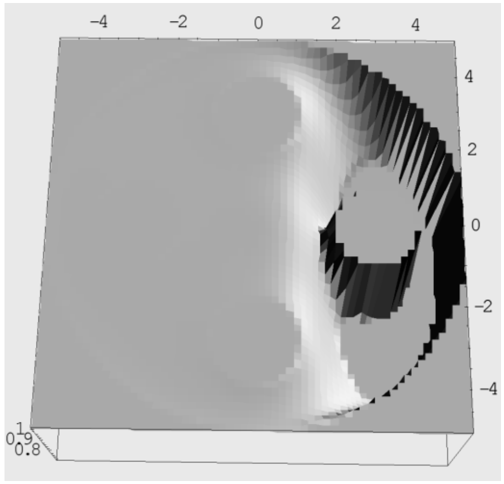

k=3


k=4


k=6

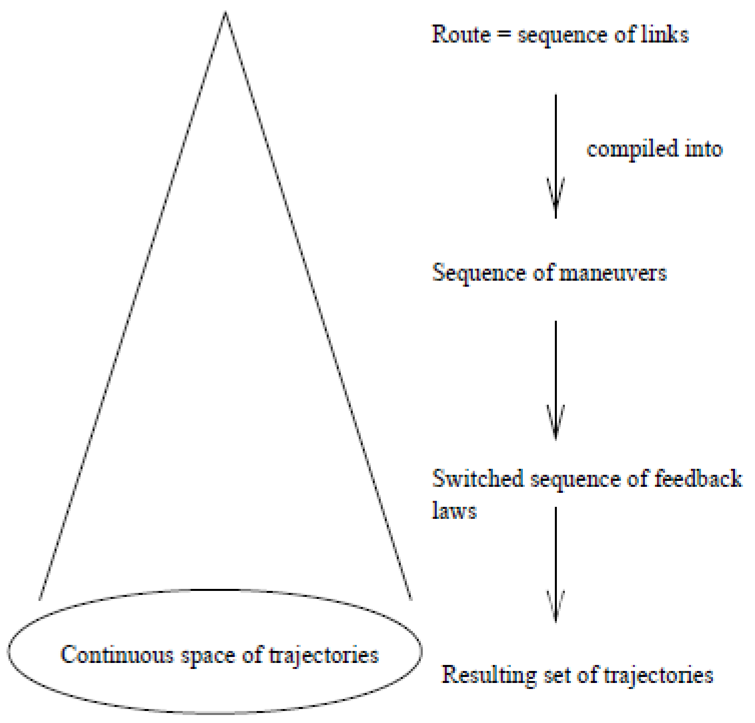

k=7


k=8

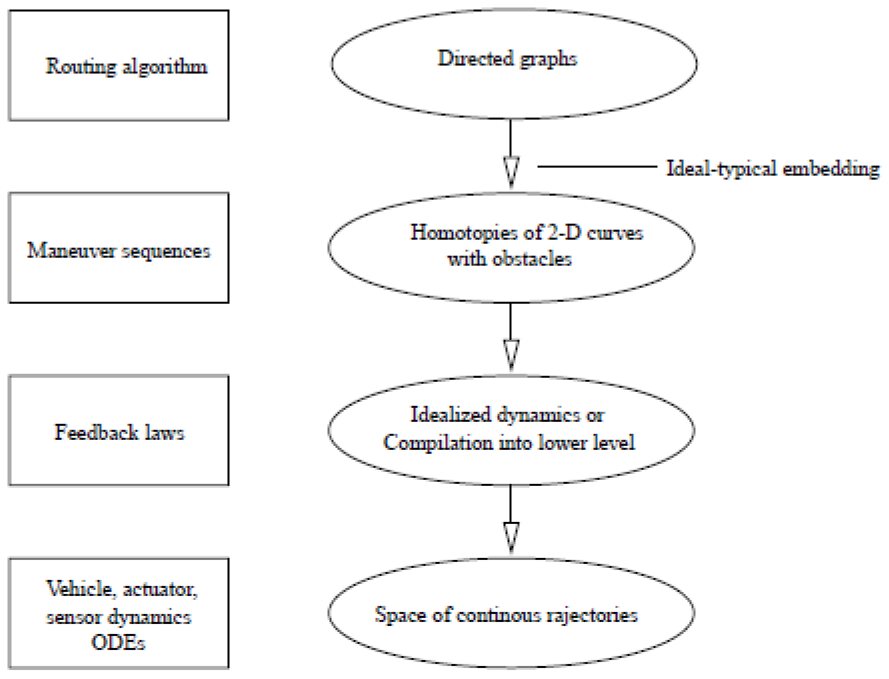

k=10

# Two Views on Complex Problems

## One-World Semantics


Route = sequence of links
compiled into
Sequence of maneuvers
Switched sequence of feedback laws
Resulting set of trajectories
Continuous space of trajectories

## Multi-World Semantics


Routing algorithm
Maneuver sequences
Feedback laws
Vehicle, actuator, sensor dynamics ODEs
Directed graphs
Ideal-typical embedding
Homotopies of 2-D curves with obstacles
Idealized dynamics or Compilation into lower level
Space of continous rajectories

[Mitter+Varaiya – see refs]

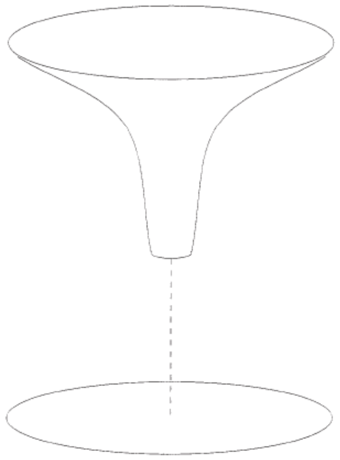
## On Multi-World Semantics

Properties and proofs of correctness are interpreted accordingly at each level:

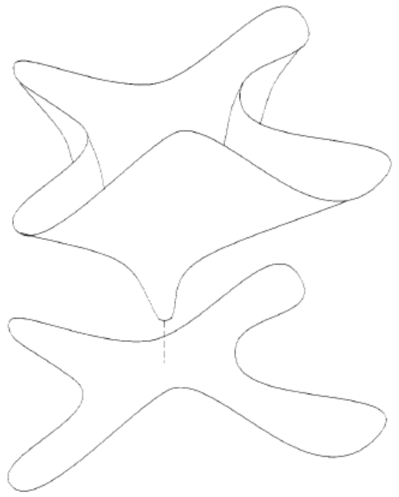
- *Proof* of ‘*route-finding algorithm is correct*’ is conducted in the semantics of directed graphs
  - Could also involve logical predicates and reasoning
- *Proof* of *‘correctness of algorithm for traversing link with sequence of maneuvers’* is conducted in semantics of homotopy of curves in 2-D space with obstacles
- *Proof* of *‘stability of feedback law’* is conducted in space of continuous trajectories.

# Burridge et al. – Sequential Composition

- Focus on "dynamical pick and place" task
  - Need for manoeuvres such as 're-grasp'
- Hybrid control architecture (two-worlds)

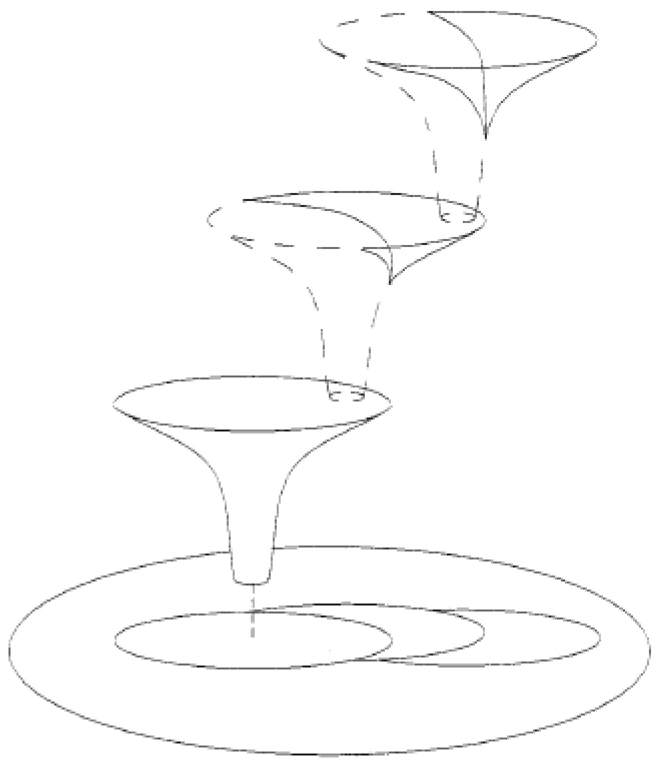


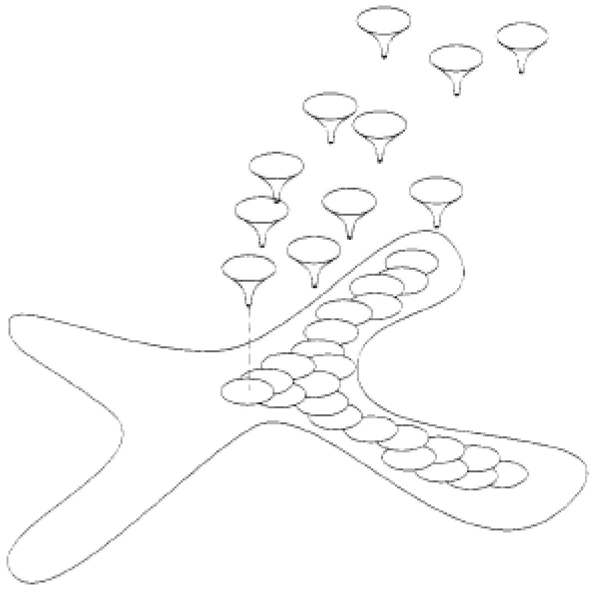
A 'symbol ' consists of a local feedback control strategy (like a potential func.)



*<u>Problem</u>*: How to construct such a thing over a large complex space of interest?

# Sequential Composition – Preimage Backchaining





# Experimental Setup

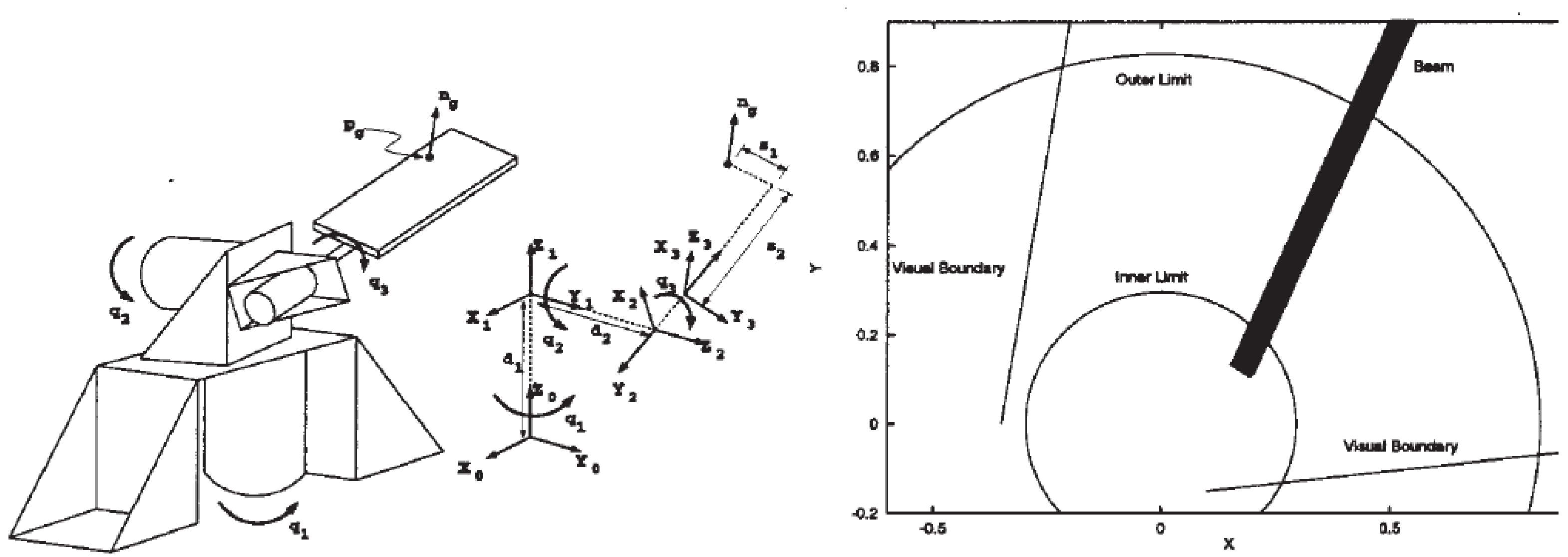


Fig. 5. The Bühgler Arm. The three joint values, $q_1$, $q_2$, and $q_3$ are referred to as $\phi$, $\theta$, and $\psi$, respectively, in this paper (a). The horizontal workspace with obstacles: the beam, inner and outer paddle limits, and the boundary of the visible workspace (b).

Note: Remember that the system can only be intermittently actuated! What does this mean for a std. feedback control law (e.g., PID or LQR)?

## How exactly is a ‘funnel’ encoded?

First, the problem is translated – via clever dynamical systems encoding – to a smaller one involving something they call the mirror law. This enforces a stable mapping in a state space.

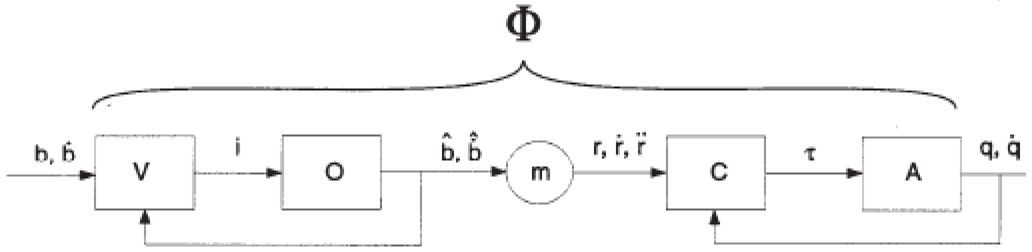


Fig. 6. Flow chart showing the various functional blocks of the system: vision, $V$; observer, $O$; mirror law, $m$; control, $C$; and actuation, $A$. The parameters of interest to this paper all reside in $m$.

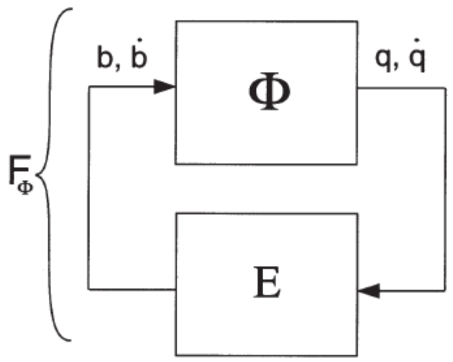
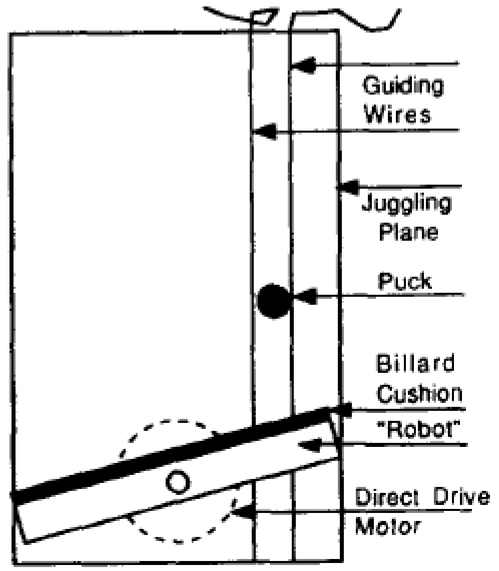


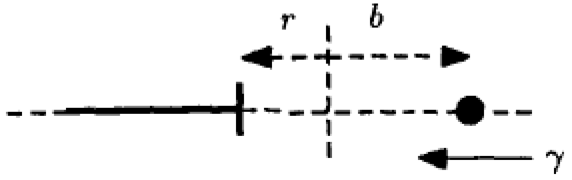
Fig. 7. The closed-loop dynamics, $F_\Phi$, induced by $\Phi$ and the environment, $E$.

# Towards the Mirror Law

Start with a line juggler



Task:



An open loop way would be to enforce post-contact vel.

$$\dot{b}' = -\alpha\dot{b} + (1+\alpha)\dot{r} = c(\dot{b}, \dot{r}),$$

The free dynamics of the ball is simply:

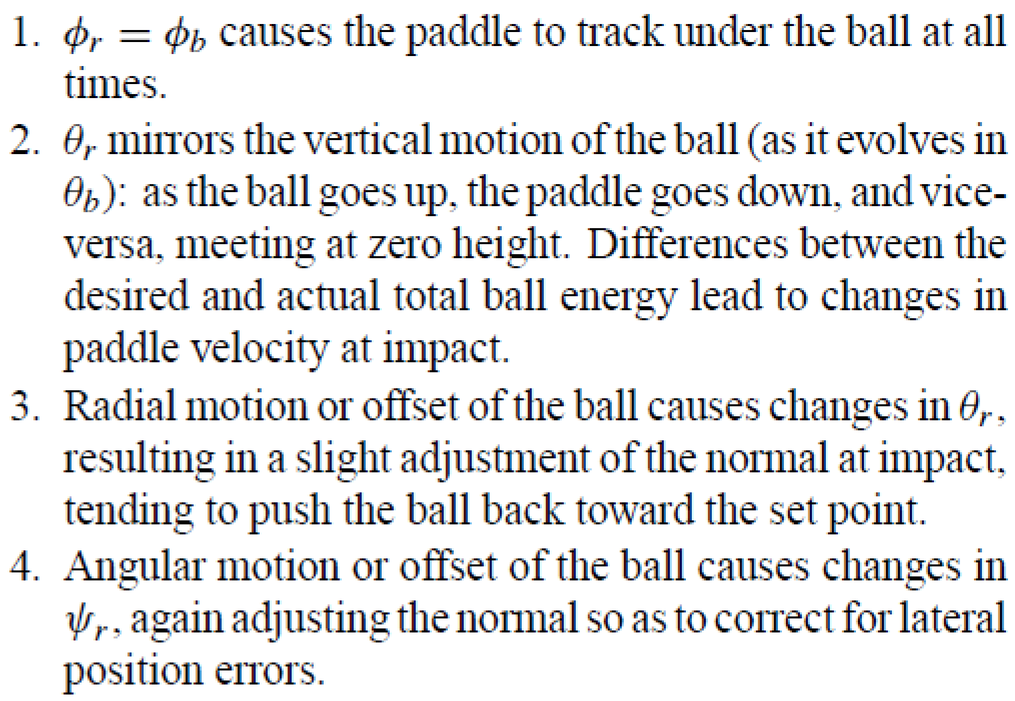
$$\begin{bmatrix} b(t) \\ \dot{b}(t) \end{bmatrix} = \begin{bmatrix} b' + \dot{b}'t - \frac{1}{2}\gamma t^2 \\ \dot{b}' - \gamma t \end{bmatrix}.$$

This works but the result isn't very stable – small noise can have big effects

## Mirror Law Control

Define a mapping from the phase space of ball to configuration space of robot arm

So, mirror law is based on getting the effector to $q(b) = (\phi_b, \theta_b, 0)$



1. $\phi_r = \phi_b$ causes the paddle to track under the ball at all times.
2. $\theta_r$ mirrors the vertical motion of the ball (as it evolves in $\theta_b$): as the ball goes up, the paddle goes down, and vice-versa, meeting at zero height. Differences between the desired and actual total ball energy lead to changes in paddle velocity at impact.
3. Radial motion or offset of the ball causes changes in $\theta_r$, resulting in a slight adjustment of the normal at impact, tending to push the ball back toward the set point.
4. Angular motion or offset of the ball causes changes in $\psi_r$, again adjusting the normal so as to correct for lateral position errors.

## Mirror Law

- Defining the vertical energy and radial distance as:

$$\eta \triangleq \gamma b_z + \frac{1}{2}\dot{b}_z^2 \quad \text{and,} \quad \rho_b \triangleq \sin(\theta_b) s_b$$

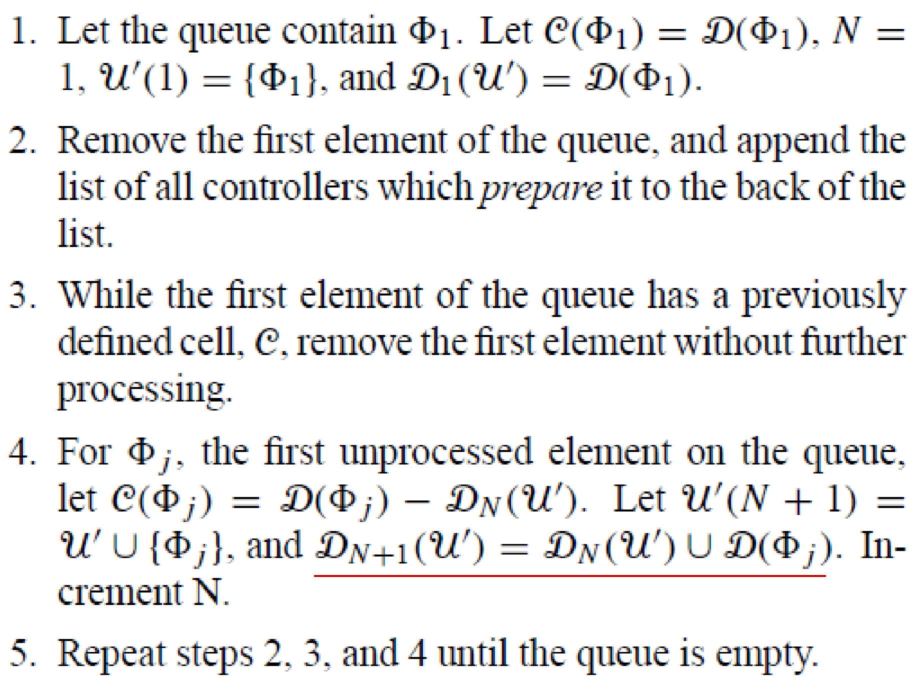
- We can describe the mirror law as:

$$m(w) \triangleq \left[ \begin{array}{c} \underbrace{-\frac{\pi}{2} - (\kappa_0 + \kappa_1(\eta - \bar{\eta}))\left(\overbrace{\phi_b}^{\text{(i)}} + \frac{\pi}{2}\right)}_{\text{(ii)}} + \\ \underbrace{\kappa_{00}(\rho_b - \bar{\rho}_b) + \kappa_{01}\dot{\rho}_b}_{\text{(iii)}} \\ \underbrace{\kappa_{10}(\phi_b - \bar{\phi}_b) + \kappa_{11}\dot{\phi}_b}_{\text{(iv)}} \end{array} \right]$$

**Note**: Sophistication of these expressions is minimal… (PD, really) The controller itself is of low complexity!

# Complete Control Strategy

- A set of controllers $\mathcal{U} = \{\Phi_1, ..., \Phi_N\}$ is designed to handle various scenarios
- Scenarios include:
  - Juggle (mirror law)
  - Palming
  - Catching



1. Let the queue contain $\Phi_1$. Let $\mathcal{C}(\Phi_1) = \mathcal{D}(\Phi_1)$, $N = 1$, $\mathcal{U}'(1) = \{\Phi_1\}$, and $\mathcal{D}_1(\mathcal{U}') = \mathcal{D}(\Phi_1)$.
2. Remove the first element of the queue, and append the list of all controllers which *prepare* it to the back of the list.
3. While the first element of the queue has a previously defined cell, $\mathcal{C}$, remove the first element without further processing.
4. For $\Phi_j$, the first unprocessed element on the queue, let $\mathcal{C}(\Phi_j) = \mathcal{D}(\Phi_j) - \mathcal{D}_N(\mathcal{U}')$. Let $\mathcal{U}'(N+1) = \mathcal{U}' \cup \{\Phi_j\}$, and $\underline{\mathcal{D}_{N+1}(\mathcal{U}') = \mathcal{D}_N(\mathcal{U}') \cup \mathcal{D}(\Phi_j)}$. Increment N.
5. Repeat steps 2, 3, and 4 until the queue is empty.

# Key Question: How do you know $\mathcal{D}$?

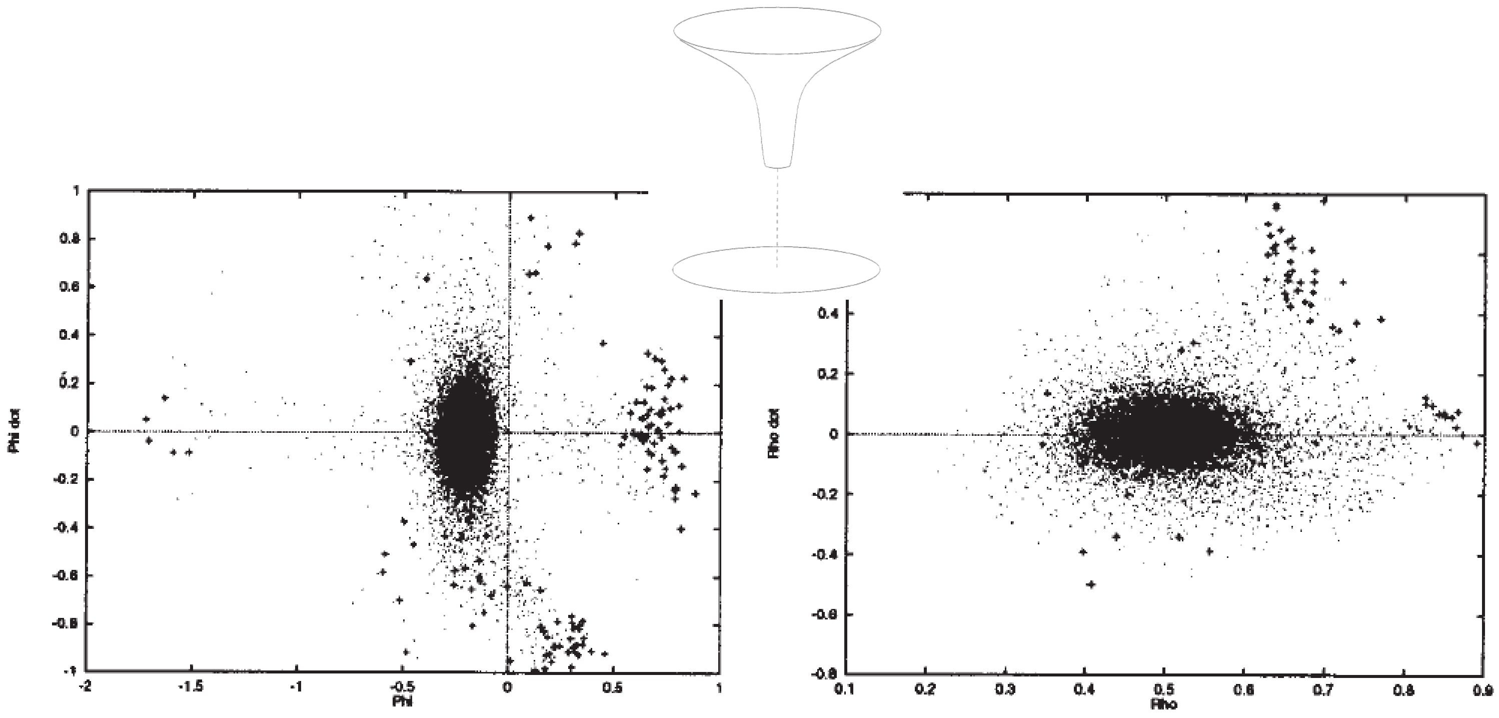
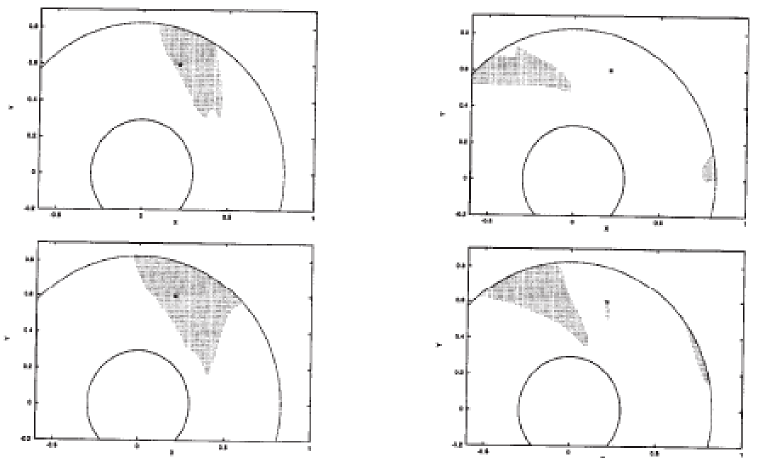


Fig. 8. Empirical data used to estimate the juggling domain, $\mathcal{D}(\Phi_J)$. Each dot (+ sign) represents in apex coordinates a ball trajectory that was successfully (unsuccessfully) handled under the action of $\Phi_J$. Because of the properties of the vertical subsystem, most of these points are at nearly the same height, so only the horizontal coordinates are plotted.

# Results



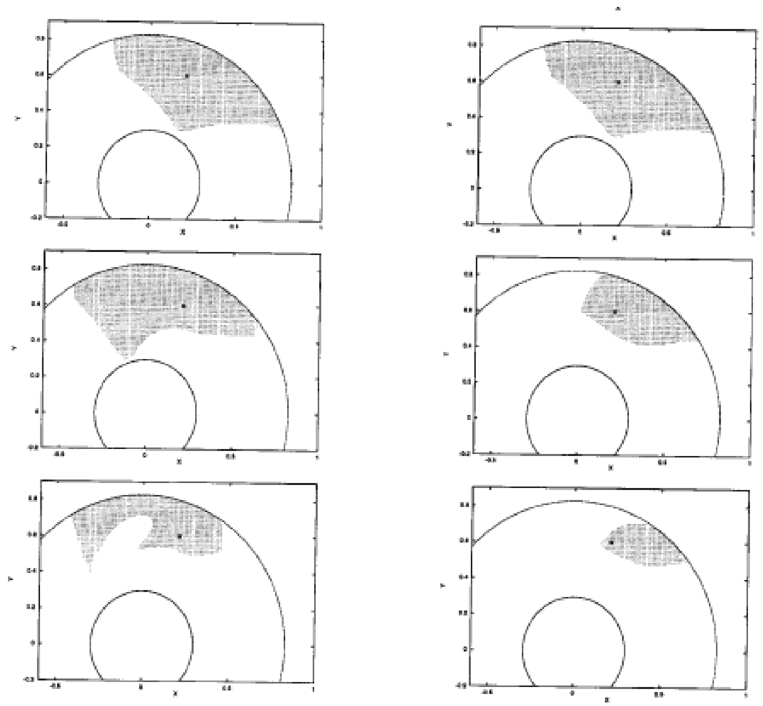


Fig. 11. Shaded regions denote initial conditions that were successfully contained in the workspace while being brought to the goal via iteration of $f_{\Phi_J}$: varying initial $\dot{x}_i$ from negative (top) to positive (bottom) with $\dot{y}_i = 0$ (left column); varying initial $\dot{y}_i$ from negative to positive with $\dot{x}_i = 0$ (right column). The scale for all plots is meters.

# Results

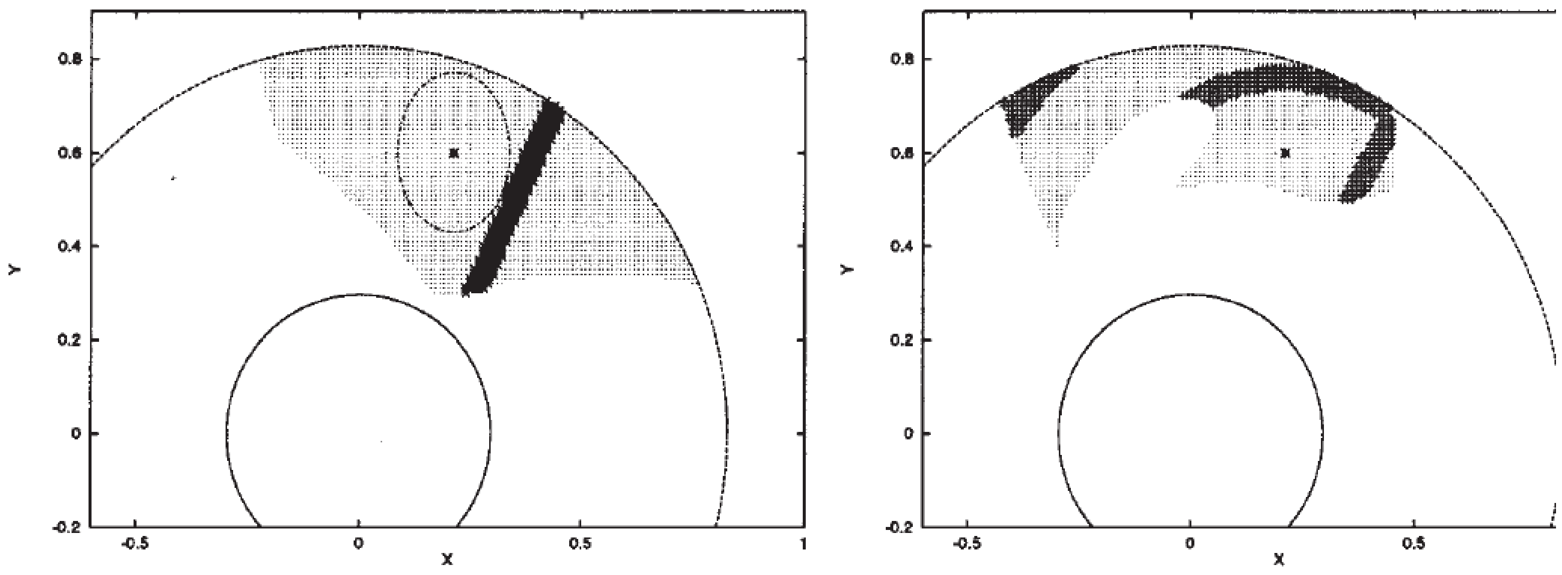


Fig. 13. The safe domain for $\Phi_{J_0}$ with the beam inserted. Light-gray areas represent successful initial states, while darker areas show states that eventually hit the beam. Zero initial velocity is shown (a), and the appropriate slice of $\mathcal{D}_0$ is added for comparison. For $\dot{x}_i = 1.5\frac{m}{s}$ (b), preimages of the beam are evident.

# What does it mean to transition?

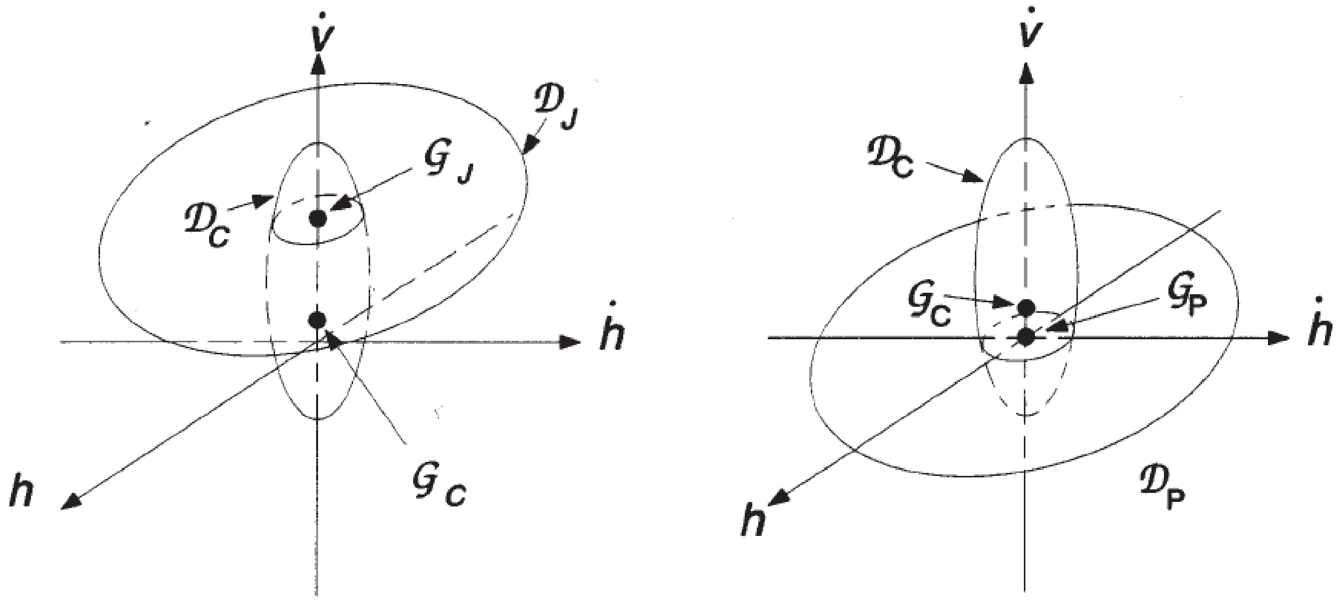


Fig. 15. Juggle to catch to palm: the juggle drives all of its domain to $\mathcal{G}_J$, which lies within the domain for the catch, but with nonzero vertical energy (a); the catch reduces the vertical energy of the ball, bringing it down into the domain for palming (b).

# Results

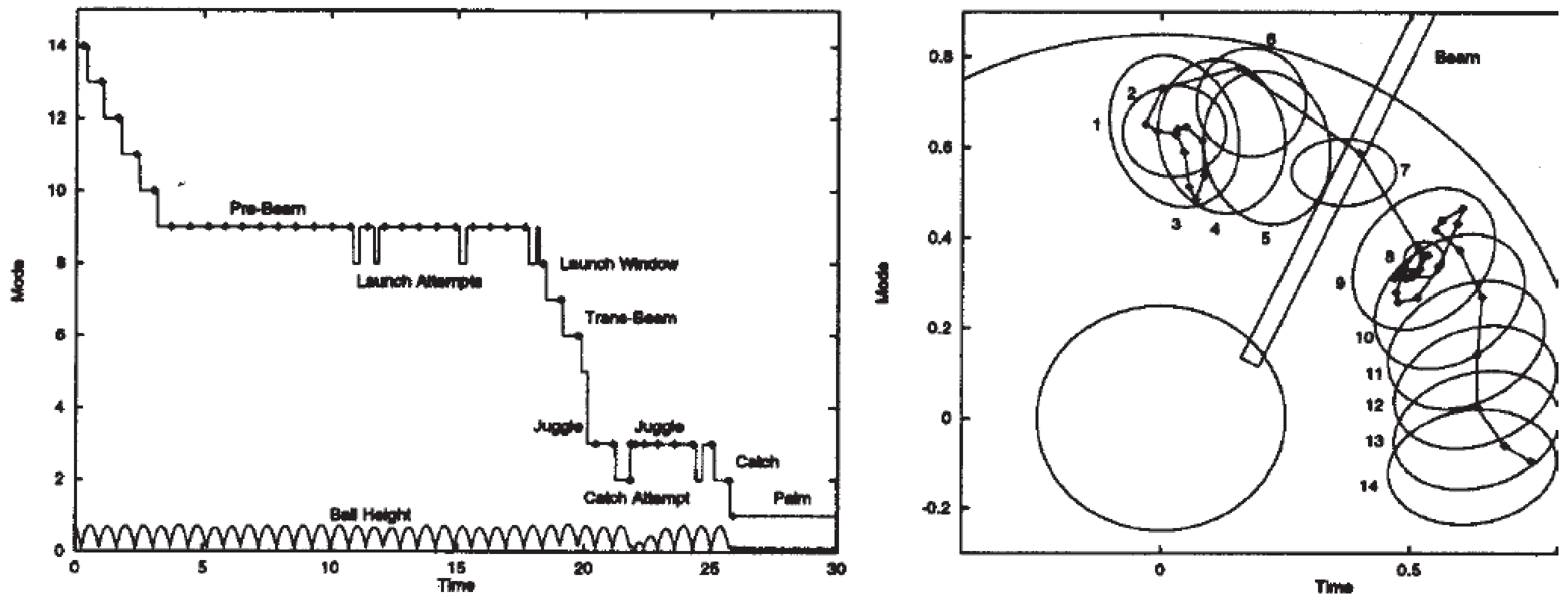


Fig. 17. The cell number or mode—also the ellipse number from Figure 16—plotted against time for a typical run, with a trace of the ball's vertical position added for reference (a). The dots represent the mode at the moment of impact. The horizontal projection of the same trace, superimposed on the deployment of Figure 16 (b). The dots show the apex positions.

# References

1. P. Varaiya, A question about hierarchical systems, In *System Theory: Modeling, Analysis and Control*, Kluwer, 2000.

2. C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, Symbolic Planning and Control of Robot Motion, *IEEE Robotics and Automation Magazine - special issue on Grand Challenges of Robotics*, 14(1): 61-71, 2007.

3. R.R. Burridge, A.A. Rizzi, D.E. Koditschek, Sequential composition of dynamically dexterous robot behaviours, *Intl. J. Robotics Research* 18: 534-555, 1999.