4-1988

Distributed Control System for a Juggling Robot

Daniel E. Koditschek
*University of Pennsylvania, kod@seas.upenn.edu*

Martin Buehler
*Yale University*

Follow this and additional works at: [http://repository.upenn.edu/ese_papers](http://repository.upenn.edu/ese_papers)

Part of the [Electrical and Computer Engineering Commons](http://repository.upenn.edu/ese_papers), and the [Systems Engineering Commons](http://repository.upenn.edu/ese_papers)

**Recommended Citation**


**NOTE:** At the time of publication, author Daniel Koditschek was affiliated with Yale University. Currently, he is a faculty member in the Department of Electrical and Systems Engineering at the University of Pennsylvania

This paper is posted at ScholarlyCommons, [http://repository.upenn.edu/ese_papers/681](http://repository.upenn.edu/ese_papers/681)

For more information, please contact repository@pobox.upenn.edu.
Distributed Control System for a Juggling Robot

Abstract
The juggling work takes its place within a larger program of research concerned with the development of unified methodologies for robot task representation, planning, and control. The talk will summarize this large context, and then provide a more detailed look at progress in juggling to date.

For more information: Kod*Lab

Disciplines
Electrical and Computer Engineering | Engineering | Systems Engineering

Comments
NOTE: At the time of publication, author Daniel Koditschek was affiliated with Yale University. Currently, he is a faculty member in the Department of Electrical and Systems Engineering at the University of Pennsylvania

This presentation is available at ScholarlyCommons: http://repository.upenn.edu/ese_papers/681
Distributed Control System for a Juggling Robot

A talk presented at the Second Annual Workshop on Parallel Computing
Portland, Oregon
April, 1988

Daniel E. Koditschek and Martin Bühler
Center for Systems Science
Yale University, Department of Electrical Engineering

We would like to acknowledge
Inmos Corporation,
Weitek Corporation,
PMI Corporation,
whose support, in conjunction with
NSF Presidential Young Investigator Award DMC-8552851
has made this research possible.

April 1, 1988
Outline of Talk

The juggling work takes its place within a larger program of research concerned with the development of unified methodologies for robot task representation, planning, and control. The talk will summarize this larger context, and then provide a more detailed look at progress in juggling to date.

1. Geometric Representation of Robotic Tasks
   (a) Abstract goals represented as optimization problems
   (b) Optimization problems “solved” by integrating gradient vector fields on an analog computer
   (c) Optimization Problems “solved” by integrating dissipative mechanical vector fields on a (properly compensated) robot-arm-analog computer.

2. Distributed Real Time Control of the Robot as Analog Computer
   (a) Goal: provably correct, automatic synthesis of real time distributed control network conceived as “analog patch panel”
   (b) Preliminary distribution experiments for a candidate robot control algorithm upon a 16 node network

3. Juggling: Robotics in an Intermittent Dynamical Environment
   (a) Geometric representation of the “environmental control problem”
   (b) A theoretical solution to the environmental control problem
   (c) Experiments with robot implementation of juggling strategies on a three node network
Representing Abstract Goals Via Optimization Problems

Configuration Space: $J$ — the problem setting;

Cost Function:

$$\varphi : J \rightarrow \mathbb{R}$$

"encodes" a goal by penalizing undesired configurations and rewarding desired configurations;

Analog Computation:

$$\dot{x} = -\nabla \varphi (x)$$

"solves" the problem by climbing down the cost function gradient

Examples:

- A/D conversion; travelling salesman problem (Neural Networks)
- VLSI Placement (Simulated Annealing)
- Robot Navigation
Background I: Dynamical Systems

Gradient Vector Fields Are "Nice"

Unlike most classes of nonlinear dynamical systems, whose limiting behavior can include "cycling" and "wandering", gradient (non-degenerate) vector fields are guaranteed to produce solutions which tend toward a single equilibrium state.

**Proposition 1** Let \( \varphi \) be a continuously differentiable Morse function on the Riemannian manifold, \( J \). Suppose that \( \text{grad} \varphi \) is transverse and directed away from the interior of \( J \) on any boundary of that set. Then the negative gradient flow has the following properties:

1. \( J \) is a positive invariant set;
2. the positive limit set of \( J \) consists of the critical points of \( \varphi \);
3. there is a dense open set \( \tilde{J} \subset J \) whose limit set consists of the local minima of \( \varphi \),

But Can We Build Them as Desired?

Does there exist a cost function which goes "high" on the boundaries of the undesirable configurations yet has only a single minimum?
Background II: Differential Topology

Topology Constrains Possible Vector Fields

**Topological Invariants:** *Euler characteristic; Betti numbers*

**General Vector Fields:** the equilibrium behavior of vector fields on a smooth manifold is constrained by the Euler characteristic (Poincaré-Hopf Theorem)

**Gradient Vector Fields:** the minima, maxima, and saddles are further constrained by the Betti numbers (Morse Inequalities).

Favorable Results of Morse and Smale


**Theorem 1** Let $M$ be a closed connected $C^\infty$ manifold. There exists a (nice) non-degenerate function on $M$ with just one local minimum and one local maximum.

- less than three decades old;
- can show, in consequence, that $C^\infty$ navigation functions exist on $C^\infty$ manifolds with boundary, as well
- what about analytic navigation functions?
The Robot Navigation Problem

Technical Statement

Given an analytic connected configuration space, $J$, find a scalar valued map,

$$\varphi : J \to [0, 1],$$

which is

analytic may be specified with one mathematical expression
(no computer programming required)

admissible goes to zero at the desired destination point; goes to unity on the boundary components
(obstacle avoidance)

polar has only a single minimum
(succesful navigation with probablility one)

Morse hessian has full rank at the extrema
(technical: nondegeneracy condition)

Invariance Under Deformation

It is not hard to show that the navigation functions are invariant under analytic diffeomorphism, hence a navigation function on a "model space" yields (at least formally) a navigation function on all configuration spaces in its deformation class!

Constructions

To date, we have succeeded in building navigation functions for configuration spaces in the deformation class of a sphere world (i.e. boundary components are formed by non-intersecting spheres), whose obstacles are star shaped sets (i.e. there is a center point in the obstacle from which rays to any boundary point remain in the interior of the obstacle).
Background III: Physics

Dissipative Mechanical Systems

**Mechanical System:** Kinetic energy, $\kappa$, defines a *Riemannian metric* on the configuration space, $J$, giving rise to a *Lagrangian vector field*, $f_\kappa$ (a second order system on the phase space, $TJ$). According to the Principal of Least Action, the integral curves of the mechanical vector field, when projected back into the configuration space, are geodesics of the Riemannian manifold, $J$.

**Potential Energy:** A cost function, $\varphi$, on the configuration space, defines a potential energy function, $\bar{\varphi}$, when "pulled back to phase space". The *vertical lift* of the gradient vector field, together with the lagrangian vector field above, $f_\kappa$, defines a new vector field, $f_{\kappa + \bar{\varphi}}$ which conserves *total energy*,

$$\eta \triangleq \kappa + \bar{\varphi}.$$

**Dissipative Field:** The addition of "damping" in the form a *dissipative vector field*, $f_\delta$, on the phase space, with the property that

$$d\kappa f_\delta \leq 0,$$

results in a vector field under which total energy decays:

$$\dot{\eta} = d\eta [f_{\kappa + \bar{\varphi}} + f_\delta] = d\kappa f_\delta \leq 0.$$

**Total Energy is a Lyapunov Function**

Lord Kelvin, *Treatise on Natural Philosophy*, Univ. Cambridge Press, 1886, §345 :

**Theorem 2** It follows that when $\varphi$ is positive for all real values of the coordinates the system must as time advances come more and more nearly to rest in its zero configuration, whatever may have been the initial values of the coordinates and velocities.
The Mechanical Analog Computer

A Global Version of Lord Kelvin’s Result

Dissipative mechanical analog computers are “just as good as gradient analog computers” at integrating cost functions:

**Theorem 3** Let $\varphi$ be a twice differentiable Morse function on $J$. The set of “bounded total energy” states

$$\Psi^\eta \triangleq \{ v \in TJ : \eta \leq \eta_0 \} .$$

is a positive invariant set of the dissipative mechanical system $f$, within which all initial conditions excluding a nowhere dense set tend toward a point in the zero section of $TJ$ identified with a local minimum of $\varphi$.

Transient Properties?

Of course, in robotics, unlike the standard optimization literature, it is not merely the limit behavior, but the transient behavior which is of concern. While the previous theorem guarantees that a minimum of $\varphi$ can be achieved with no obstacle collisions, it says nothing about the manner in which that desired destination is approached.
A Tinkertoy Set for Exploration of Distributed Real Time Controllers

Philosophy:

- require computationally powerful "patch panel" for mechanical analog computer
- buy out of design of parallel hardware and concurrent software
- buy into design of network topology, process distribution strategy, and customization of computational identity (where needed).

Inmos Transputer:

Powerful CPU: 32 bit 10 MIPs, 1 Mflop RISC machine;

Fast Inter-Node Communication: 4 independent bi-directional 10 Mband DMA links

Powerful Development Environment: real-time multitasking operating system; high level structured language and support of concurrency; network symbolic debugger

Motion Control Board: The Yale XP/DCS.

- single extended Eurocard (100 mm x 220 mm) four layer printed circuit board, pin compatible with INMOS ITEM development system
- 124 Kbyte zero wait state local memory
- fiber optic link ports
- "bus expansion connector" and daughter board for fast i/o or co-processor interface,
Goal: Correct, Automatic Synthesis

Desire

- "map" abstract goals as encoded via geometric specification to control laws implemented on a specified distributed processing network
- automatic synthesis procedure (no human coding required)
- desired properties are provably invariant under transformation

\[
\text{human} \quad \downarrow
\]
\[
\text{task, robot description} \quad \rightarrow \quad \text{algorithm} \quad \rightarrow \quad \text{code} \rightarrow \quad \text{network} \quad \downarrow
\]
\[
\text{motors}
\]

Current Investigation

- "Computed Torque" Algorithm:

\[
\tau_{\text{desired}} = C(q, \dot{q})\dot{q} + M(q) (\ddot{r} + K_2[\dot{r} - \dot{q}] + K_1[r - q])
\]

- Generate \( \tau_{\text{desired}} \) from robot description, and generate C code from \( \tau_{\text{desired}} \), both via Symbolic Manipulation package. Then experiment with hand distribution schemes:

\[
\text{human} \quad \downarrow
\]
\[
\text{robot description} \quad \xrightarrow{\text{SMP}} \quad \text{"computed torque" code} \quad \text{Human} \quad \rightarrow \quad \text{network} \quad \downarrow
\]
\[
\text{network}
\]

- no proofs yet
Real Time Computational Issues

Precursors

Digital Control Theory
Distributed Computation

Real Time Sampling Considerations

Nyquist Rate: lower bound on sampling frequency for assured control — twice the highest expected frequency (in practice, one order of magnitude);

Global Rate: sampling frequency assuming complete computation upon exact data ("latency"?);

Local Rate: sampling frequency assuming asynchronous partial computation by each process upon most recent available data ("throughput"?);

Real Time Accuracy Considerations

Enlarged Controller:

Nominal Control Structure: \[
\hat{e}_1(n+1) = a_{11}\hat{e}_1(n) + a_{12}\hat{e}_2(n) \\
\hat{e}_2(n+1) = a_{21}\hat{e}_1(n) + a_{22}\hat{e}_2(n)
\]

Actual Control Structure: \[
c_1(n+1) = a_{11}c_1(n) + a_{12}c_2(n-1) \\
c_2(n+1) = a_{21}c_1(n-1) + a_{22}c_2(n)
\]

Controller Dynamics:

integration required for smoothing, adaptation, etc: need shared clock?

Numerical Accuracy:

on-line adjustment of internal accuracy for numerically unstable algorithms; on-line tradeoffs between accuracy and sampling rate, etc?
SMP Input file for RRR Planar arm.

/* KINEMATICS: Link Transformations: */
T[1] : { { Cos[q[1]], - Sin[q[1]], 0, 0 }, \n    { Sin[q[1]], Cos[q[1]], 0, 0 }, \n    { 0, 0, 1, 0 }, \n    { 0, 0, 0, 1 } }

T[2] : { { Cos[q[2]], - Sin[q[2]], 0, 1[1] }, \n    { Sin[q[2]], Cos[q[2]], 0, 0 }, \n    { 0, 0, 1, 0 }, \n    { 0, 0, 0, 1 } }

T[3] : { { Cos[q[3]], - Sin[q[3]], 0, 1[2] }, \n    { Sin[q[3]], Cos[q[3]], 0, 0 }, \n    { 0, 0, 1, 0 }, \n    { 0, 0, 0, 1 } }

T[4] : { { 1, 0, 0, 0, 0, 1[3] }, \n    { 0, 1, 0, 0 }, \n    { 0, 0, 1, 0 }, \n    { 0, 0, 0, 1 } }

/* DYNAMICS: Link Inertia Tensors: */
M[1] : { { 1[1]^2 m[1], 0, 0, 1[1] m[1] }, \n    { 0, 0, 0, 0 }, \n    { 0, 0, 0, 0 }, \n    { 1[1] m[1], 0, 0, m[1] } }

M[2] : { { 1[2]^2 m[2], 0, 0, 1[2] m[2] }, \n    { 0, 0, 0, 0 }, \n    { 0, 0, 0, 0 }, \n    { 1[2] m[2], 0, 0, m[2] } }

M[3] : { { 1[3]^2 m[3], 0, 0, 1[3] m[3] }, \n    { 0, 0, 0, 0 }, \n    { 0, 0, 0, 0 }, \n    { 1[3] m[3], 0, 0, m[3] } }
First Term of RRR equation of motion:

1]) + 1[1]*1[2]*mass[2]*qdotdot[2] + 2*(1[1,\ 
1]*1[2]*mass[3]*qdotdot[1]) + 1[1]*1[2,\ 
1]*mass[3]*qdotdot[2] + 2*(1[1]*1[3]*mass[3]\ 
*qdotdot[1]*Cos[q[3]]) + 1[1]*1[3]\ 
*mass[3]*qdotdot[2]*Cos[q[3]] + 1[1]\ 
*1[3]*mass[3]*qdotdot[3]*Cos[q[3]] - (1[1,\ 
1]*1[3])*mass[3]*qdotdot[2]^2*Sin[q[3]]\ 
+ -2*(1[1]*1[3])*mass[3]*qdot[1]*qdot[2,\ 
1]*Sin[q[3]]) + -2*(1[1]*1[3])*mass[3]*qdot[1,\ 
1]*Sin[q[3]]) + -2*(1[1]*1[3,\ 
1]*mass[3]*qdot[2]*qdotdot[3]*Sin[q[3]])\ 
+ Cos[q[3]]*(2*(1[2]*1[3]*mass[3]*qdotdot[1,\ 
1]) + 2*(1[2]*1[3])*mass[3]*qdotdot[2])\ 
+ -2*(1[1]*1[3])*mass[3]*qdotdot[3]*qdot[1]*qdot[2,\ 
1]*Sin[q[2]]) + -2*(1[1]*1[3])*mass[3]*qdotdot[3]*qdot[1,\ 
1]*Sin[q[2]]) + -2*(1[1]*1[3,\ 
1]*mass[3]*qdot[2]*qdotdot[3]*Sin[q[2]])\ 
*1[3])*mass[3]*qdotdot[3]) - 2*(1[2,\ 
1]*1[3])*mass[3]*qdotdot[2]*qdotdot[3])\ 
*Sin[q[3]] + -(1[1]*1[2])*mass[2]*qdotdot[2]\ 
-2) + -(1[1]*1[2])*mass[3]*qdotdot[2]^2) + -2*(1[1,\ 
1]*1[2])*mass[2]*qdotdot[1]*qdotdot[2]) + -2*(1[1,\ 
1]*1[2])*mass[3]*qdotdot[1]*qdotdot[2]) + -2*(1[1,\ 
1]*1[3])*mass[3]*qdotdot[1]*Sin[q[3]])\ 
+ -(1[1]*1[3])*mass[3]*qdotdot[2]*Sin[q[3]])\ 
1]*2*mass[2]*qdotdot[1] + 1[1,\ 

13
INMOS TRANSPUTER DEVELOPMENT SYSTEM
A 1 NODE NETWORK.

A 2 NODE NETWORK.

A 4 NODE NETWORK.

A 10 NODE NETWORK.
Picture of a particular 12-transputer distribution scheme for an RRR planar arm.

Computed torque inverse dynamics control algorithm.

The $m_i$'s are motor-transputers.

The $c_i$'s are dedicated to a single trigonometric term.

$M(q)$ is calculated at lower left.

$C(q, \dot{q})\dot{q}$ is calculated at lower right.

$[\ddot{r} - K_1(r - q) - K_2(\dot{r} - \dot{q})]$ is calculated at lower center.
Why Juggle?

1. Engage an inescapably dynamical environment

- Leave the realm of tasks involving
  - static geometry (e.g., obstacle avoidance)
  - time varying geometry (e.g., collision avoidance for multiple arms)
  - static force (e.g., compliant insertion)

- Confront a number of unactuated (intermittently coupled) degrees of freedom with a smaller number of actuated degrees of freedom.
  - legged locomotion
  - "fumbling" (pre-grasping)

2. Extend the class of geometrically specified robotic tasks as far as possible.

3. FUN!
Control System Overview

- Development System
  - code development, distribution, network analyzer/debugger
  - real time logging of remote nodes

- Puck Sensor
  - puck state measurement
  - asynchronous communication with motor controller

- Motor Controller
  - juggling algorithm
  - asynchronous communications with puck sensor
  - logging to development system
Prototype Juggler Kinematics

Robot coordinates in joint space:

\[ q = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \]

Body coordinates in world frame:

\[ z = \begin{bmatrix} p \\ \dot{p} \end{bmatrix}, \quad p = \begin{bmatrix} x_{body} \\ y_{body} \end{bmatrix}, \quad \dot{p} = \begin{bmatrix} \dot{x}_{body} \\ \dot{y}_{body} \end{bmatrix} \]

"Frozen last hit frame" with respect to world:

\[ ^0R_h \triangleq \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}; \quad ^0\dot{R}_h \equiv 0 \]

Flight model between hits:

\[ \begin{array}{c}
\ddot{p}(j + t) = \dot{p}'(j) - 2ta; \\
\quad a \triangleq \begin{bmatrix} 0 \\ \frac{1}{2}g \end{bmatrix}
\end{array} \]

\[ p(j + t) = p(j) + t\dot{p}'(j) - \frac{1}{2}ta \]

Robot control input — time elapsed between hits:

\[ u_1 \triangleq t \]

\[ \theta(j + t) = \arctan \left( \frac{y(j+t)}{x(j+t)} \right) \]
Impact Model

Assumptions:

No friction at contact:

\[ h_x'_{robot} = h_x_{robot} \]

Infinite robot mass:

\[ h_y'_{robot} = h_y_{robot} \]

Attenuated energy restitution:

\[ h_y'_{body} = h_y'_{robot} = \alpha \left( h_y_{body} - h_y_{robot} \right); \quad \alpha \in (0, 1) \]

Relationship Between Velocities Before and After Impact:

Robot control input — impact velocity normal to bar:

\[ u_2 = h_y_{robot} \]

New velocity after impact:

\[ h_p' = \begin{bmatrix} 1 & 0 \\ 0 & -\alpha \end{bmatrix} h_p + \begin{bmatrix} 0 \\ 1 + \alpha \end{bmatrix} u_2 \]

\[ \triangleq A h_p + h_b u_2. \]

Expression in the world frame:

\[ \dot{p}' = ^{0}R_h h_p' \]

\[ = ^{0}R_h A ^{0}R_h \tau p' + ^{0}R_h h_b u_2 \]

\[ \triangleq M(\theta)\dot{p} + b(\theta)u_2. \]

Juggling Environment Model:

\[ z_{j+1} = \begin{bmatrix} p_j + (M_p \dot{p}_j + b_j u_{2,j})u_{1,j} - au_{i,j} \\ M_j \ddot{p}_j + b_j u_{2,j} - 2au_{1,j} \end{bmatrix} \triangleq f(z_j, u_j). \]
Geometric Task Definition: The Vertical One-Juggle

Task set:
\[ \mathcal{T} \triangleq \text{span} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \] = \{ z \in \mathcal{Z} : \dot{x} = 0, y = 0 \}.

Definition:

The feedback law,
\[ g : \mathcal{Z} \to \mathcal{U} \]
constitutes a vertical one-juggle with respect to the task, \( z^* \in \mathcal{T} \), if \( z^* \) is a fixed point of the closed loop system,
\[ z^* = f_g(z^*); \quad f_g(z) \triangleq f(z, g(z)), \]
and is a stable attractor of the resulting discrete dynamics.

**Proposition 2** Given the discrete dynamical control system, and a point, \( z^* \in \mathcal{Z} \), there exists a feedback law, \( g : \mathcal{Z} \to \mathcal{U} \) such that \( z^* \) is a fixed point of the closed loop map,
\[ z^* = f_g(z^*), \]
if and only if
\begin{enumerate}
  \item \( z^* \in \mathcal{T} \);
  \item \( g(z^*) = u^* = \begin{bmatrix} -2/g \\ \frac{1}{1+\alpha} \end{bmatrix} \dot{y}^* \).
\end{enumerate}
Local Controllability Around the Task Plane

Proposition 3 If

\[ z^* \in T - 0 \]

then the system is locally controllable. That is, if

\[ A = D_z f[z^*, u^*]; \quad B = D_u f[z^*, u^*] \]

then \((A, B)\) is a completely controllable pair.

A Locally Stabilizing Feedback Law:

General affine feedback structure for \(z^* \in T:\)

\[ u(z) = g_z(z) + K(z - z^*) \]

Particular case:

\[ z^* = \begin{bmatrix} 5 \\ 0 \\ 0 \\ -5 \end{bmatrix} ; \quad K_f = \begin{bmatrix} -0.01 & 0.33 & -0.05 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix}, \]
Sample Radius of Initial Conditions with Bounded Solution

For typical settings, the Domain of Attraction resulting from the locally stabilizing feedback law has a “diameter” in phase space no larger than 6 % of desired fixed point.
Sample Radius of Initial Conditions with Contained Solution

For typical settings, the Domain of Containment (trajectory guaranteed to stay within the physical boundaries of our juggling plane) resulting from the locally stabilizing feedback law has a "diameter" in phase space no larger than 2% of desired fixed point. This is smaller than the error tolerance of the puck position sensing system.