ANNUAL PROGRESS REPORT

ONR GRANT NO. N00014-97-1-0599

Project title:

PRACTICAL CONTROL ALGORITHMS FOR NONLINEAR DYNAMICAL SYSTEMS USING PHASE-SPACE KNOWLEDGE AND MIXED NUMERIC AND GEOMETRIC COMPUTATION

Principal Investigator:

Dr. Feng Zhao, Associate Professor
Department of Computer & Information Science
The Ohio State University
2015 Neil Avenue
Columbus, Oh-43210-1277

September 17, 1998
September 11, 1998

Dr. Michael F. Shlesinger
ONR 331
Office of Naval Research
Ballston Center Rower One
800 North Quincy Street
Arlington, VA 22217-5660

Dear Mike,

I would like to summarize our research progress during the period of October 1, 1997 through September 30, 1998. The project, titled “Practical control algorithms for nonlinear dynamical systems using phase-space knowledge and mixed numeric and geometric computation”, is funded under the ONR YI Award (grant No. N00014-97-1-0599). Our goal is to develop high-performance computational tools for designing control systems for a class of complex physical systems.

- Developed a verification algorithm for verifying control laws using phase-space geometric modeling of dynamical systems. The algorithm evolves a hierarchically-refined bound of system nonlinear dynamics and can address practical concerns such as sensor, actuator, and modeling uncertainties in a systematic manner. We have applied the algorithm to the maglev control system prototype and compared the results against the physical measurements.

- Constructed a physical experiment to study distributed acoustic sensing. The experiment comprises an enclosed chamber measured $1.7m \times 0.846m \times 0.201m$, and an 8-channel A/D and D/A system with 6 microphones and a speaker.

- Started to investigate control system design and optimization for distributed parameter physical systems (systems modeled by partial differential equations).

  - Jeff May and Feng Zhao, “Verification of control laws using phase-space geometric modeling of dynamical systems.” IFAC AIRTC-98.

I have enclosed copies of the above mentioned articles. Please do not hesitate to contact me if
you need additional information.

Best Regards,

Feng Zhao
Associate Professor
VERIFICATION OF CONTROL LAWS USING PHASE-SPACE GEOMETRIC MODELING OF DYNAMICAL SYSTEMS

Jeff May * Feng Zhao**

* Department of Computer and Information Science
  Ohio State University
  Columbus, OH 43210
  Email: may-j@cis.ohio-state.edu
** Xerox Palo Alto Research Center
  3333 Coyote Hill Road
  Palo Alto, CA 94304
  Email: zhao@parc.xerox.com

Abstract: This paper presents an algorithm for verifying control laws using phase-space geometric modeling of dynamical systems. The algorithm evolves a hierarchically-refined bound of system nonlinear dynamics and can address practical concerns such as sensor, actuator, and modeling uncertainties in a systematic manner. The algorithm has been applied to verifying a control law for a magnetic levitation system, and the computational results are compared against the performance of the actual physical system.

Keywords: Control systems, Verification, Phase space, Geometric approaches, Computational methods.

1. INTRODUCTION

Control verification ensures correct behaviors for controlled physical systems. Important applications range from safety-critical systems such as aircraft controllers, where improper behavior can result in loss of life, to cost-critical systems such as factory controllers, where a faulty controller can result in costly inefficiency. Unfortunately, obtaining a closed-form analytic solution to the verification problem is often impractical. The nonlinear nature of many man-made systems requires that approximations be made to apply most verification techniques. Uncertainties such as modeling and sensing error make it difficult to express the range of possible behaviors of the system in a tractable form. Thus, verifying a controlled system frequently requires that linear approximations be made, and that considerations for factors such as modeling error and sensing error be omitted.

This paper describes a computational verification algorithm that relies on evolving phase-space geometric models of system dynamics. The contributions of this paper are threefold: (1) It introduces a novel hierarchical refinement of bounds on system dynamics to avoid unnecessary over-approximation of nonlinear behavior; (2) The phase-space representation it uses can model nonlinearity and uncertainty in a systematic, intuitive manner; (3) The algorithm has been applied to verify a nonlinear maglev control system prototyped in our laboratory. Although the phase space of the example used in the paper is two dimensional, the algorithm is applicable to higher-order control systems as well as those whose dynamics does not admit a closed-form analytic description
but whose states can be fully observed via experimental means.

A number of approaches to verification exist for simple linear systems. Recent work has focused on developing methodology for more complex systems such as hybrid systems (see e.g. Alur et al., 1996; Bouajjani et al., 1993). The approaches in (Henzinger et al., 1995; Puri et al., 1996; Greenstreet and Mitchell, 1998; Dang and Maler, 1998) use a polygonal or grid-based representation for phase-space regions of a hybrid system during verification, and share several similar concepts with the algorithm presented here. Our algorithm evolves from earlier work in phase-space control synthesis (Bradley and Zhao, 1993; Zhao et al., 1997).

2. PHASE-SPACE VERIFICATION ALGORITHM

The phase-space verification algorithm is used to verify proper regions of operation that have the desired limit behaviors for stabilization control systems (i.e. control systems that are designed to stabilize the plant in a goal region). Other properties such as overshoot or convergence rate can also be verified with only minor changes to the algorithm. The algorithm is applicable to discrete-time control systems with a fixed sampling frequency. The underlying dynamics of the plant may be continuous, discrete, or hybrid. It is assumed that the system is designed to operate within a bounded region of the phase space. Let the system dynamics be described by: $\dot{x} = F(x, u)$, where $x$ is the system state, and $u$ is the control input. Since the system is discretely sampled, the dynamics of the controlled system can be written as $x_{t+1} = f(x_t)$, where $x_t$ denotes the system state at time $t$, and $f(x_t) = \int F(x, u)$ after one time period (the controller output at $x_t$).

The algorithm proceeds as follows:

1. Partition the phase-space region of interest into a finite set of cells $C$.
2. Determine the initial controllable region $R_{cont}$.
3. For each cell $c$ in $C - R_{cont}$, do:
   a. Find a polytope $p_c$ bounding the image of $c$ under $f$.
   b. Compute the “escape polytope” $e_c = p_c - c$.
   c. If $e_c$ is contained within $R_{cont}$, and $f$ generates no cycles within $c$, mark $c$ as verified, and set $R_{cont} = R_{cont} \cup c$.
4. If any new cells were added to $R_{cont}$ in step 3, repeat step 3.
5. If the region of interest has been verified, or if a pre-specified number of steps has been taken, quit. Otherwise, form a new set $C'$ by subdividing the unmarked cells in $C$. Set $C = C'$, and go to step 3.

The partitioning of the phase space in step 1 is arbitrary; however, regular partitions are often used, and certain types of control suggest preferred partitions. For example, control based on cell maps suggests an initial partition identical to the one used to generate the cell maps (Hsu, 1987).

Determination of the initial controllable region requires a bit more effort. There are two basic approaches. If the controller has already been verified for a certain region $R_1$ (e.g. using analytic techniques), and the verification algorithm is being used to extend that region, $R_{cont}$ is set to $R_1$, and all cells that are fully contained within $R_1$ are marked. This approach is taken, for example, when a local controller (e.g. one based on linear techniques) is being augmented by a global controller.

If no “pre-verified” region is available, $R_{cont}$ cannot just be set to the goal region, because it is possible that for the given controller, the plant can start out in the goal region, but later exit it and never return. Thus, in this approach, a set $R_{cont}$ of “core cells” must be found. The “core cells” have the following two properties:

1. Every $c \in R_{cont}$ is in the goal region.
2. For every $c \in R_{cont}$, the image of $c$ under $f$ is contained in $R_{cont}$.

A maximal set of core cells (for a given phase space partition) can be generated by selecting all cells contained in the goal region, and then iteratively eliminating cells whose image bound lies outside the set of selected cells.

Finding a polytope $p_c$ bounding $f(c)$ can be achieved in many ways. One of the simplest and most efficient ways to find a suitable $p_c$ is to compute the minimum and maximum values for each component of $F$ over the cell $c$ and form a bounding box for $f(c)$ using these values. In hybrid systems terms (see e.g. (Branicky, 1995)), each cell can be thought of as a discrete state, and the bounding polytope is determined by approximating system dynamics with a rectangular differential inclusion.

In step 3, we are checking for two properties:

1. $\forall x : f^n(x) \notin c$ for some $n \in \mathbb{N}$. That is, all trajectories of the system starting within $c$ eventually exit $c$.
2. $\forall x : f^n(x) \notin c \land f_{n-1}(x) \in c \Rightarrow f_n(x) \in R_{cont}$. That is, when a trajectory exits $c$, it reaches a cell that has already been marked as verified.
Checking the second property is straightforward. To check the first property we intersect \( f(c) \) with \( c \) to form a polytope \( p_i \). This process is repeated with \( p_i \) until the intersection is empty, or a pre-specified number of iterations, \( i \), is exceeded. Thus, we replace the first property with a stronger condition—that all trajectories leave \( c \) within \( i \) time steps.

Assuming that a regular, rectangular initial partition is used, and that subdivision is done uniformly, the space requirement of the algorithm is \( O((2^d) \times n) \) where \( d \) is the dimension of the phase space, \( s \) is the level of subdivision, and \( n \) is the number of cells in the initial partition. Thus, the memory requirements depend linearly on the size of the initial partition, and exponentially on the level of subdivision.

Step 1 typically has time complexity linear in \( n \), although a complex partition may require more time. Step 2 requires at most \( O(n^2) \) time if a core set is being determined. Each iteration of step 3 takes \( O(n) \) time (for the comparison with \( R_{\text{cont}} \) in 3c), so the entire algorithm requires \( O((2^d) \times n^2) \). As with the space complexity, the time complexity has an exponential dependence on the level of subdivision, and a polynomial dependence on the size of the initial partition. With a regular partition, the comparison in step 3c depends only on the cells intersected by the image bound, which is typically far less than \( n \). Similarly, the order of selection in step 3 can have a large impact on the efficiency of the algorithm. In practice, the verification of a cell far from the initial controllable region depends on the verification of cells nearer the region. An intelligent ordering, that starts from the cells nearest the controllable region and works outward, often results in more cells being verified for each iteration of step 3, thus reducing the number of times step 3 must be iterated to a number far less than \( n \).

2.1 Proof of Soundness

In this section, it is shown that if the system starts in a cell that has been marked as verified, the system will eventually progress to a state within the goal region.

**Proof:** Number the cells of the phase space partition in the order that they are marked, with all initially verified cells (or core cells) numbered zero. Consider cell number \( i \), with \( i > 0 \). Since the cells are numbered by order of marking, all trajectories starting in cell \( i \) eventually flow into a lower numbered cell (since cell \( i \) will be marked only if its image bound lies in cells that have already been marked). Thus, by induction, all trajectories starting in a cell with a positive number will eventually flow into a cell numbered zero. By assumption (or by the definition of core cell), no trajectories starting from a zero numbered cell will leave the set of zero numbered cells, so all trajectories will eventually progress to a state within the goal region.

Note that this only guarantees that marked cells exhibit proper behavior—there is no guarantee that all cells that exhibit proper behavior will be marked (i.e. the algorithm is sound but not necessarily complete).

2.2 Enhancements and Optimizations

In the above description, several practical issues, such as measurement error, controller output error, and modeling error are not mentioned. However, because of the geometric nature of the computation, such considerations can be incorporated in a straightforward fashion. Measurement error can be accounted for by expanding the cell when the bounding polytope is determined. Controller output error and model error can be dealt with (assuming the error is bounded) by expanding the image bounding polytope corresponding to the potential range of the function \( f \) describing the dynamics.

Continuous time systems can be verified by selecting a time period, and examining the evolution of the system over this time period (i.e. treating the system as a discretely-sampled system, and using the base verification algorithm). If necessary, this time period can be iteratively reduced to provide a less conservative bound on system behavior (as is done with the phase-space partition in the base algorithm).

Certain other properties of a system can be verified with minor modifications to the base algorithm. For example, suppose it is necessary to verify not only that the system reaches the goal region, but also that the percent overshoot is limited. In this case, when a cell is being checked, its image bound must fall within marked cells and each of these intersected cells must be annotated as having trajectories with a tolerable maximum distance to the goal region. The newly marked cell is then annotated with its maximum distance as well (computed as the maximum of the annotated values of the intersected cells and the distance of the cell itself).

In the base algorithm, only the behavior after one sampling period is considered. This is because the bounding polytope of the image of a cell increases in size exponentially with time, thus making the bound less accurate the longer the time period considered. However, when the system dynamics are "slow" in comparison to the partition gran-
Fig. 1. Slow dynamics with respect to the partition granularity and the sampling period results in dependence on an adjacent cell.

Fig. 2. "Spiraling" trajectories in phase space.

Fig. 3. Cyclic dependence between cells created by spiral trajectories above.

Fig. 4. Sequence of images with decreasing unverified area. The shaded area represents the previous verified region. The transparent boxes represent a sequence of image bounds (b is the image bound of a, c is the image bound of b − R_{cont}, et cetera).

regularity and the sampling period, a cell's image bound will often overlap with a neighbor, resulting in a dependence from the cell to its neighbor (Since the neighbor must be verified before the cell in consideration can be; see figure 1). If the system has "spiraling" trajectories (figure 2), a cycle of dependence (figure 3) between a set of unverified cells can occur. In this case, the partition must be subdivided several times to properly verify the system. This subdivision is costly—both memory usage and computation time scale exponentially with the level of subdivision in the worst case. In these cases, the algorithm can be optimized by continuing to iterate the function f when doing so is beneficial. If the escape polytope e_c does not lie entirely within verified cells, the polytope is clipped against those unverified cells it intersects, and the resulting polytopes are recursively checked. This process continues until either the cell is verified (i.e. the iteration produces an image Bound that is contained within the verified region), or no further "progress" is made. The current implementation considers "progress" to Be made as long as the unverified area (volume) occupied By the Bounding polytope is decreasing in size (see figure 4). Other criteria may also Be useful.

Algorithmically, this optimization replaces steps 3a - c with a call to the following function on cell c:

```
Bool CheckRegion(region r)
(1) Find a polytope p_r Bounding the image of r under f.
(2) Compute the "escape polytope" e_r g p_r = r.
(3) Set r_{unverified} g e_r = R_{cont}.
(4) If r_{unverified} φ return true.
(5) Otherwise, if volume(r_{unverified}) ≤ volume(r),
    return CheckRegion(r_{unverified}), else return false.
```

3. RESULTS AND ANALYSIS

The verification procedure has Been tested on the controller for a magnetic levitation system (figures 5 and 6). This system is a useful testBed since it is inherently unstable and nonlinear (due to the inverse square law of magnetic attraction).

The initial verifiable region is oBtained using a local controller, generated using a linearized model of the original nonlinear system. This linear controller is augmented By a nonlinear, phase space Based gloBal controller (as in (Loh, 1997; Zhao et al., 1997)). The equilibrium point was set to 11.6 millimeters from the Bottom of the solenoid to the center of mass of the steel Ball. The coordinate system is chosen such that the displacement vector points downwards from the solenoid to the steel Ball. The local controller was used when the Ball was within one millimeter of the equilibrium point with a velocity having absolute value less than 0.05 meters per second.
Fig. 6. Photo of the actual magnetic levitation testBed prototyped at Ohio State University.

Outside this region, the gloBal controller was invoked.

The optimized algorithm was used to generate a

Fig. 7. Region of the magnetic levitation system's phase space that is marked as verified by the verification algorithm.
4. RELATED WORK

The HyTech system (Alur et al., 1996) uses convex polyhedra to represent regions of a hyBrid system and shares the concept of verification through exploration of the pre-images of the desired goal states. Hy-Tech allows the expression of properties to be verified as explicit mathematical formulas. For many properties, representation as a simple formula is more flexiBle than the implicit representation of properties through cell annotation. Hy-Tech is designed to easily represent hyBrid systems, which must be represented by an augmented phase space in our approach. However, the geometric approach used here allows for a more straightforward representation of real-world uncertainties. Furthermore, our algorithm is guaranteed to terminate, and allows for a systematic refinement of approximations of dynamics.

Recently, several researchers suggested verification techniques based on projecting phase-space regions. In (Greenstreet and Mitchell, 1998), the authors described a method for computing projection polyhedra oBtained from integrating initial regions. While an efficient and exact algorithm exists for two-dimensional linear systems with convex spaces, a general higher-dimensional polyhedra have to be reconstructed from a series of two-dimensional subSpace projections. Because the paper did not provide implementation details and computational results for the higher-dimensional case, it is difficult to evaluate the effectiveness of the algorithm. The approaches in (Puri et al., 1996; Dang and Maler, 1998) share a grid-based representation of phase space with the algorithm developed in this paper. In comparison, our algorithm introduces a hierarchically refined Bound of system dynamics to avoid unnecessary over-approximation.

5. CONCLUSIONS

An algorithm for verification of control laws using phase-space geometric modeling was presented. This algorithm is applicaBle to a wide range of control systems that are continuous, discrete, or hyBrid, and can be used with a variety of forms of control laws. Once a Bound for measurement, controller output, and modeling uncertainty is oBtained, considerations for these types of uncertainties can easily be incorporated into the verification algorithm. The algorithm was applied to a nonlinear controller for a magnetic levitation system, and the resulting region of staBility was compared to that of the actual physical system.

Future avenues of research include exploring the applicaBility of the algorithm to other real systems, and investigating optimal initial partitions for different control laws. Furthermore, more accurate and flexiBle techniques for measuring the region of staBility of the physical maglev system need to be developed so that the results of the verification algorithm can more readily be compared to those of the actual system.

6. ACKNOWLEDGMENT

This work is conducted at Ohio State University, supported in part by ONR Y1 grant N00014-97-1-0599 and NSF NYI grant CCR-9457802. We thank S. Loh for designing and prototyping the maglev control testBed.

7. REFERENCES

Tutorial MP2

Intelligent Simulation

Feng Zhao and Chris Bailey-Kellogg

Fifteenth National Conference on Artificial Intelligence

Madison, Wisconsin

Monday, July 27, 1998

All contents copyright ©1998, Feng Zhao and Chris Bailey-Kellogg
Tutorial MP2

Intelligent Simulation

Feng Zhao and Chris Bailey-Kellogg

Fifteenth National Conference on Artificial Intelligence

Madison, Wisconsin

Monday, July 27, 1998
ACKNOWLEDGEMENT

- Ken Yip is a major collaborator in developing the Spatial Aggregation theory and language. Elisha Sacks has collaborated on developing imagistic reasoning.

- Members of Intelligent Simulation Group at Ohio State have contributed towards the development and implementation of spatial aggregation language and applications (Project INSIGHT Web page: http://www.cis.ohio-state.edu/insight/). In particular, Iván Ordóñez, Xingang Huang, and Qiang Wang helped define and implement part of the SAL library; Jeff May and Shiou Loh developed the Maglev control algorithms and experiment; Xingang Huang built a prototype program for weather data analysis; Iván Ordóñez investigated the modeling and analysis of diffusion-reaction systems.

- The work on Spatial Aggregation is supported in part by NSF Young Investigator award CCR-9457802, ONR Young Investigator award N00014-97-1-0599, a Sloan Research Fellowship, NSF grant CCR-9308639, and a Xerox Foundation grant.
Tutorial Roadmap

- Introduction and Motivation
- Overview of Spatial Aggregation (SA):
  Imagistic reasoning, theory, ontology, language
- Examples of Spatial Aggregation:
  KAM, MAPS, HIPAIR, Fluids
- Cognitive Foundation and Related Work
- Spatial Aggregation Language (SAL)
- SA Applications:
  Distributed optimization and control, weather data analysis, maglev control experiment, diffusion-reaction morphogenesis
- References

Physical Fields are Ubiquitous

- Temperature, sound, fluid, ...
- Practical applications:
  - Smart buildings
  - Constellation of space probes
  - Weather map interpretation
  - MEMS arrays
Examples of Physical Fields

- Turbulent fluid flow

A fluid field showing high density regions and large vortex structures

- Weather maps

A 300mb weather map over North America showing temperature (iso-therms) and wind velocity streamlines
- Movie Camera

Film advancing mechanism  A configuration space

E. Sacks and L. Joskowicz

---

- Heat treatment of materials

Temperature distribution over a piece of material

S. Vavasis
Characteristics of Physical Fields

- Spatially distributed
- Continuous and data rich
- Multiple spatio-temporal scales
- Large D.O.F. and often nonlinear

Conventional Simulation

- Model-simulate-verify scenario:
  Iterate till the questions can be answered.
- Simulation software is monolithic.
- Mostly numerical computation.
Intelligent Simulation

A body of computational theories, techniques, and programming tools that combine numeric, geometric, and symbolic methods to solve problems in scientific data mining and engineering design.

Characteristics:

- Mixed numeric, geometric, symbolic computation on continuous and discrete representations of physical phenomena
  - Intelligent scientific computing [Abelson, Eisenberg, Halfant, Katzenelson, Sacks, Sussman, Wisdom, and Yip, 1989]

- Articulable and structured models explicating simplifying assumptions, causes, effects, time, space, scales etc.
  - Compositional modeling [Falkenhainer & Forbus, 1991]

- Modular building blocks for rapid prototyping of programs
  - MATLAB-like tools
Tutorial Roadmap

- Introduction and Motivation
  - Overview of Spatial Aggregation (SA):
    - Imagistic reasoning, theory, ontology, language

- Examples of Spatial Aggregation:
  - KAM, MAPS, HIPAIR, Fluids

- Cognitive Foundation and Related Work

- Spatial Aggregation Language (SAL)

- SA Applications:
  - Distributed optimization and control, weather data analysis,
    maglev control experiment, diffusion-reaction morphogenesis

- References

Overview of Spatial Aggregation (SA)

- **Input:** data-massive, numerical field
  
  *E.g.* weather maps, seismic signatures, numerical simulation data

- **Output:** high-level description of structure, behavior, and control actions
  
  *E.g.* stability regions and bifurcation diagrams for dynamical systems, C-space free region diagrams for mechanical mechanisms, synthesized control reference trajectories

- **Task domains:** Sensor data interpretation; Control

- **Central computational problem:** Uncover structures of physical fields
Imagistic Reasoning

Perceptual operations on image-like, analogue representation of physical systems

- Analogue representation:
  - Information rich, pictorial:
    * Shannon-Weaver measure (lots of bits!)
    * Structures/relations implicitly represented
    * E.g., light intensity array, temperature field, fluid motion velocity data, phase space
  - Continuous
    * The representation varies smoothly w.r.t. parameter changes except for singular points

- Direct manipulation of and tight coupling to physical data:
  - Primarily perceptual, secondarily symbolic
  - Combine deductive reasoning with perceptual processing
  - No need to generate intermediate symbolic predicates and to perform expensive search on them
  - Reason about geometric structures and their interactions
  - Explain behaviors by directly inspecting structures and their changes w.r.t. perturbations
**Context**

- Central AI problem: map signal to symbols and back
- Spatial Aggregation as a realization of imagistic reasoning
- Draw upon techniques from computer vision, qualitative reasoning, scientific computing, and computational geometry
- Modeling: abstraction/economy of description/data reduction
- Control: goal to action; global objective to local constraints

**Tasks Suitable for Spatial Aggregation**

- Explanation generation
- Computer-aided tutoring
- Fault diagnosis and prediction
- Scientific data mining
- Design problems involving complex geometries
Desired Properties

- **Abstractness**: Able to compute abstract global properties.
- **Open-endedness**: Basic operators modular and composable to support a variety of task domains.
- **Efficiency**: Local and non-goal-specific operators.
- **Soundness**: Descriptions consistent with known physical and mathematical principles.
- **Succinctness**: Results containing qualitatively important distinctions.

Reasoning Tasks

- **Infer structural descriptions**: Identify field objects and their shapes, sizes, locations, distribution, evolution over time.
- **Classify**: Assign semantic labels to objects and configurations.
- **Infer correlations**: Determine how geometry/distribution of one type of objects correlate with those of another?
- **Check consistency**: Given two objects or configurations, test if they are pairwise consistent.
- **Infer incremental behavior**: Given an instantaneous configuration, predict its short-term behaviors.
- **Infer behavioral descriptions**: Explain and summarize object evolution by domain-specific interaction rules.
Basic Elements of Spatial Aggregation

- Field ontology:
  How to describe the problem.

- Neighborhood graphs and generic operators:
  How to decompose the problem and formulate problem-solving steps.

- Multi-layers of spatial aggregates/transformation:
  How to actually solve the problem.

---

Field Ontology

- A field is a mapping from one continuous space to another: $R^m \rightarrow R^n$
  - A grey-level image: $R^2 \rightarrow R^1$
  - Temperature field of the room: $R^3 \rightarrow R^1$
  - Wind velocity field in the air: $R^3 \rightarrow R^3$

- Metric on a field $\rightarrow$ Topology $\rightarrow$ Continuity

- A field is analogue: pointwise, numerical, information rich
Spatial Objects

- A physical field exhibits multiple spatio-temporal scales. E.g. temperature decays at different rate

- Spatial objects:
  - By continuity, a continuous space partitions into a moderate number of (open) regions separated by compact boundaries
  - Points in a region share similar properties w.r.t. tasks
  - Focus on discrete regions and transitions among them while abstracting away points in a region

- By locality, global properties of a field can be computed by iteratively combining local properties
Elements of Field Ontological Abstraction

- Spatial Objects
  - Geometric description
  - Feature description
    Examples: temperature; pressure; velocity

- Constitutive Laws
  Examples:
  - Fourier's law: heat flux = \(-k \frac{dT}{dx}\)
  - Ohm's law: charge flux = \(-\gamma \frac{dV}{dx}\)
  - Hooke's law: stress = \(E \frac{du}{dx}\)

- Spatial Neighborhood Structures
  Examples:

- Minimal spanning tree
- Regular grid neighborhood graph
- Delaunay mesh neighborhood graph

- Interaction Rules
  Examples:
  - Causal interaction
  - Consistency rules
  - Update rules
Neighborhood Graphs

- Neighborhood graph (N-graph): nodes are objects in a field and edges explicitly encode adjacencies
- Objects are aggregated into an N-graph based on spatial proximity
- N-graphs modularize computation:
  - As a uniform interface between multiple layers of aggregates
  - As a glue for operators that manipulate, filter, transform, and search aggregates

- Examples of useful neighborhood structures:
  - Phase-space state clusters form connected components of N-graph
  - C-space free-space regions give rise to connectivity in N-graph
- N-graphs support efficient computation on aggregates of spatial objects: search, classification, consistency checking, etc.
  - Adjacency structure localizes constraint propagation and path search
  - Inconsistency among adjacent objects focuses analysis
  - Transitive closure on adjacent objects form equivalence classes of objects
Examples of N-Graphs

A Minimum spanning tree

A regular grid of nodes
A mesh of triangle elements

Example: Fluid Flow

Two adjacent, counter-rotating velocity streamlines A and B in an incompressible planar fluid flow. Inconsistency rule for A and B flags that there must be missing features, in this case recirculation zones, in the annular region between A and B.
Multi-Layer Transformation of N-Graphs

- A small set of operators construct and transform spatial aggregates
- Identical set of operations at each layer, parameterized by task and domain specific metric and relations
  - *Aggregate* forms a neighborhood graph explicitly encoding adjacencies.
  - *Classify* forms equivalence classes of similar neighboring objects.
  - *Redescribe* maps equivalence classes to higher-level objects.
Spatial Aggregation Language (SAL)

- Support rapid prototyping of problem solvers for imagistic reasoning tasks
- Provide data types and operators
- Modular construction of transformations, organized by parameterized N-graph constructs

Language Features

- Data types:
  N-graph and its constructors, accessors, modifiers.
  Examples of N-graph: 4-adjacency, MST, and Voronoi diagram.
  Field and its constructors, accessors, modifiers.
  Examples of Field: array, grid, K-D tree, etc.
- Interface operators:
  aggregate, classify, re-describe, localize, search,
  incremental-analyze, pairwise-consistent?, consistent?
  A user must specify the neighborhood relation, field metric, and equivalence relation for these operators.
- Geometric utilities: intrinsic-geometry, contain?
intersect, \( \partial, \delta \).

- Interface to numerical and image processing libraries:
  FFT, convolution, integrator, linear system solver, vector/matrix algebra.

SA Example: Trajectory Bundling

Task: given a set of sample points, to identify groups of trajectories with similar limit behaviors.
Trajectory Bundling Example Steps

(a) sample points
(b) MST
(c) curve graph
(d) trajectory bundles

Page MP2-37

Trajectory Bundling Data Flow

trajectory curves → aggregate (connected substructure) ➔ curve N-graph ➔ redescribe (trajectory bundles)

curve classes ➔ classify ➔ limit behavior, threshold

points → aggregate (minimal spanning tree) ➔ point N-graph ➔ redescribe (trajectory curves)

point classes ➔ classify ➔ relative distance

Page MP2-38
SA Example: Boundary Tracing

Task: to group boundary segments from the same object.

```
0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 0
0 0 0 1 0 0 0 1 0
0 1 1 1 1 0 1 0 1
0 1 0 1 0 1 0 1 0
0 1 0 1 1 1 1 1 0
0 1 0 0 0 1 0 0 0
0 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0
```

Boundary Tracing Example Steps

(a) boundary pixels
(b) 4-adjacency graph
(c) boundary segment graph
(d) object contours
### Boundary Tracing Data Flow

- **contour consistency rules** → **consistent?** → **aggregate**
  - nearness neighborhood

- **segment N-graph** → **redescribe** → object contours
  - boundary segment classes

- **collinearity, threshold**

- **pixels** → **aggregate**
  - 4-adjacency neighborhood

- **pixel N-graph** → **redescribe** → boundary segments
  - pixel classes

- **classify**
  - value, junction

---

### Tutorial Roadmap

- **Introduction and Motivation**
- **Overview of Spatial Aggregation (SA):**
  - Imagistic reasoning, theory, ontology, language

- **Examples of Spatial Aggregation:**
  - KAM, MAPS, HIPAIR, Fluids

- **Cognitive Foundation and Related Work**
- **Spatial Aggregation Language (SAL)**
- **SA Applications:**
  - Distributed optimization and control, weather data analysis, maglev control experiment, diffusion-reaction morphogenesis

- **References**
Examples of Spatial Aggregation

- KAM: analyzing nonlinear dynamical systems (Ken Yip)
- MAPS: synthesizing control laws (Feng Zhao)
- HIPAIR: performing kinematic analysis of mechanisms (Leo Joskowicz & Elisha Sacks)
- Mining data from fluid dynamics simulation (D. Silva, N. Zabusky, et al.; Ken Yip)

KAM

- Task: Interpret qualitative behaviors of Hamiltonian systems such as the solar system
- Input: state equation of a Hamiltonian system, parameter ranges.
- Output: a summary of qualitatively distinct behaviors
- Key ideas:
  - Phase-space geometric analysis
  - Classification of geometric objects based on their shapes
  - Local compatibility rules and constraint propagation
- Application: KAM was used in solving an open problem in fluid dynamics – predicting onset of chaotic motion in wave tanks
Phase Portrait of a Hamiltonian System

Henon's description of the motion of a star within a galaxy:

\[ x_{n+1} = x_n \cos \alpha - (y_n - x_n^2) \sin \alpha \]
\[ y_{n+1} = x_n \sin \alpha + (y_n - x_n^2) \cos \alpha \]

Important questions: when does the system undergo simple periodic motion? or even chaotic motion?

A Hamiltonian system does not dissipate energy.

---

Phase-Space Geometric Analysis

- Poincare's Idea: geometries of phase space encode complex local and global dynamical behaviors.
  E.g. Swinging pendulum in a magnetic field.

- Terminology:
  - Phase space: a space spanned by state variables of a dynamical system (E.g. a n-D Euclidean space)
  - Vector field: a mapping that takes a point in phase space to a direction
  - Orbit: an integral curve of the vector field
  - Phase portrait: union of orbits filling a phase space
  - Bifurcation: qualitative changes in behaviors as system parameters vary
Phase-Space Description of Behaviors

- State: a point in phase space
- Trajectory (orbit): 1-D curve
- Equilibrium state: a stationary point in vector field, classified as attracting (stable), repelling (unstable), and saddle (meta-stable).
- Limit cycle (periodic orbit): a closed 1-D curve
- Chaotic orbit: irregular trajectory with no apparent structure
- Basin of attraction: a region that maps to an attractor in the limit (also called the stability region)
- Poincare section: a slice of phase space; behaviors are analogously defined in the section: periodic, almost periodic, island chain, separatrix, chaotic.

Examples of Phase-Space Orbits

- Attractor
- Repellor
- Saddle
- An attracting limit cycle
- A chaotic orbit (in 3D or higher)
- Two basins of attraction (left/right)
Good References on Phase-Space Analysis


- J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag, New York, 1983. (A more advanced Graduate text on nonlinear dynamical system analysis)

How Does KAM Recognize an Orbit?

- Embed the orbit in a data structure called MST

- Compute the distinguishing characteristics (branching factor, edge length, etc.) of MST

- Apply pre-defined classification rules to parse the orbit into one of the known types: periodic orbit (fixed point), quasiperiodic orbit (KAM curve), island chain, separatrix, chaotic orbit

**Key Observation:** KAM performs similar computations in searching phase spaces and parameter spaces, after individual orbits are identified.
Orbit Identification Using Geometric Signatures

An MST embedding orbit points

A magnifying box showing edges classified as inconsistent. After deleting the inconsistent edges from the MST, KAM parses the structure into one of the known orbit types: 4-island chain.
KAM's Local Consistency Rules

- Orbits do not intersect
- Neighboring orbits have to be compatible in flow directions

KAM uses the local rules to incrementally identify missing features of phase space, avoiding an exhaustive search.
Examples of Spatial Aggregation

- KAM: analyzing nonlinear dynamical systems (Ken Yip)
  - MAPS: synthesizing control laws (Feng Zhao)
- HIPAIR: performing kinematic analysis of mechanisms (Leo Joskowicz & Elisha Sacks)
- Mining data from fluid dynamics simulation (D. Silva, N. Zabusky, et al.; Ken Yip)

MAPS

- Task: Synthesize control laws for dissipative systems based on a qualitative analysis (e.g. Balancing a pole)
- Input: state equation of a dissipative system, initial and goal states, optimality constraints, admissible control values.
- Output: control paths connecting the initial and goal states and satisfying design constraints.
- Key ideas:
  - Path search in phase space
  - Flow pipes (equivalence classes of behaviors)
  - Flow pipe graph (reachability)
- Application: MAPS was used to synthesize a nonlinear, global control law for a magnetic levitation system.
Problem: Stabilizing a Buckling Steel Beam

Buckling in steel beams (under axial loads) induces structural failures in buildings or bridges.

How to stabilize buckling by controlling marginal stability?

---

Control Synthesis as Path Search in Phase Spaces

- Control law as a path connecting initial and goal states in phase space
- Searching out a stack of continuous phase spaces parameterized by different control actions — Expensive!!!
Flow Pipes

- Partition a continuous space into a manageable collection of discrete objects
- Each flow pipe represents an equivalence class of trajectories:
  - Exhibiting similar behaviors (e.g. flow into the same sink)
  - Homotopy equivalent (continuously deformable to each other)

![Flow Pipes Diagram](image)

Phase-Space Navigation

- Extract qualitative similar trajectories (flow pipes)
- Plan control reference trajectory

![Phase-Space Navigation Diagram](image)
Flow-Pipe Graph

- Formed by intersecting flow pipes from a stack of parameterized phase spaces
- Flow pipe graph encodes path reachability and optimality constraints
  - Edge weight represents "goodness" of the edge
  - Shortest path search suffices

Find Solution by Searching Out Flow-Pipe Graph

- Force control is switched on
- Region R is projected onto the initial phase plane.
Synthesized Anti-Buckling Control Law

Control reference trajectory

Control vs. time

Computational Structure of MAPS

Page MP2-63

Page MP2-64
Discussion I

- The input point sets to both KAM and MAPS come from numerical simulation of system models or sensor measurements.
- While KAM deals with cross-section of 3D structures, MAPS processes 3D (or higher) structures directly and uses the qualitative description to synthesize nonlinear control actions.
- Although designed for different tasks, KAM and MAPS' computational structures are strikingly similar:
  - Object aggregation, classification, and re-description at each layer
  - Aggregate objects are manipulated as primitive objects at the next higher level.

Examples of Spatial Aggregation

- KAM: analyzing nonlinear dynamical systems (Ken Yip)
- MAPS: synthesizing control laws (Feng Zhao)

> HIPAIR: performing kinematic analysis of mechanisms (Leo Joskowicz & Elisha Sacks)

- Mining data from fluid dynamics simulation (D. Silva, N. Zabusky, et al.; Ken Yip)
HIPAIR

- Task: Perform kinematic analysis of fixed-axes mechanisms (e.g. gear box)
- Input: Shape and motion type description for each interacting pair of parts in a mechanism
- Output: Realizable configurations of the mechanism represented by a CS region diagram
- Key ideas:
  - Feasibility analysis in configuration space (CS)
  - Incremental computation exploiting spatial adjacency of CS regions, avoiding an exhaustive search

Application: HIPAIR is used to analyze 2,500 common mechanisms from Artobolevsky’s four-volume encyclopedia *Mechanisms in Modern Engineering Design*:

- Include couplers, indexers, and dwells used in printing presses, mills, motion-picture cameras, and cars
- Found 66% kinematic pairs and 58% mechanisms are feasible
An Example: 3-Finger Cam-Follower

A cam follower

Its configuration space

Configuration Space (CS)

- Each dimension for a single D.O.F. of a mechanism.
  E.g. The CS for a rigid part in a plane is 3D: two for translational D.O.F. and one for rotational D.O.F.

- Point: a configuration of the mechanism

- Feasible configuration: physically realizable configuration

- Infeasible configuration: physically unrealizable configuration (e.g. two overlapping rigid parts)

- Free space region: a set of feasible configurations

- Blocked space region: a set of infeasible configurations

- Region diagram: a graph with free space regions as nodes and spatial adjacencies as edges
An Example of Configuration Space

A robot in a 2D physical space

Its CS for translations

Adapted from P.H. Winston, 1992

Computational Structure of HIPAIR
Discussion II

- HIPAIR exhibits similar computational patterns as KAM and MAPS, using the same set of operators.

- Our objective is to extract these common building blocks to better understand why these programs work and to build future programs using these blocks.

Examples of Spatial Aggregation

- KAM: analyzing nonlinear dynamical systems (Ken Yip)

- MAPS: synthesizing control laws (Feng Zhao)

- HIPAIR: performing kinematic analysis of mechanisms (Leo Joskowicz & Elisha Sacks)

- Mining data from fluid dynamics simulation (D. Silva, N. Zabusky, et al.; Ken Yip)
Interpreting Fluid Simulation Data

- Visiometrics (Silver and Zabusky): extract and track features
- Structural inferences as an example of Spatial Aggregation (Yip): extract and classify structures

An example of fluid data (Raleigh-Taylor instability):

Visiometrics

- Task: Extract and track features from fluid simulation data sets (e.g. low pressure regions, high vorticity areas)
- Input: Gridded data set over a spatial domain
- Output: Features and their evolution in time
- Key ideas:
  - Regions of fluid field as coherent objects
  - Field segmentation
  - Object correspondence
- Application: visualize turbulence data
Extracting Objects

- Fluid objects: a set of adjacent points above or below a threshold value plus their boundaries
- Extracting objects: use 3D segmentation or region growing to extract connected thresholded region
- Object attributes: mass, centroid, maximum, volume, moment, bounding surface, skeletons
Tracking Objects

- Correspondence:
  - Match an object in a field with one in another field
  - Use object attributes

- Two modes of tracking:
  - Postprocessing: identify all objects, then correlate
  - Preprocessing: identify objects in the initial frame and search for matching objects in subsequent frames

Evolution of Objects

- Continuation: object persists over time
- Birth/Death: creation or disappearance of objects
- Bifurcation: object breaks into pieces
- Collide: objects merge to form a single object

An object is said to correspond to another if their overlapping region exceeds certain threshold value.
Application of Spatial Aggregation: 
Infer Structures in Fluid Data

- Task: Extract structures from fluid simulation data sets (e.g. vortex bundles and their spatial adjacency)
- Input: Data set over a spatial domain
- Output: Structural description (such as vortices, streaks, shear layers)
- Key ideas:
  - Object cohesiveness
  - Aggregate and classify object structures
  - Explaining physics by structural morphogenesis of fields
- Application: build geometric models for fluid data

Building a Conceptual Model for Fluid Phenomena

The phenomenon is explained in terms of objects, their spatial distribution and interaction, their temporal evolution, their changes in shapes, and birth/death of the objects. [Yip, IJCAI97]
Aggregation and Classification

- Field data is aggregated into isosurfaces by a modified marching cube algorithm
- A vortex object is defined as a set of adjacent curves that share similar properties
- Extract shape description for a vortex using generalized cylinders

Future work:

- Correlate spatial objects
- Establish cause-effect relation

Tutorial Roadmap

- Introduction and Motivation
- Overview of Spatial Aggregation (SA):
  Imagistic reasoning, theory, ontology, language
- Examples of Spatial Aggregation:
  KAM, MAPS, HIPAIR, Fluids

▷ Cognitive Foundation and Related Work
- Spatial Aggregation Language (SAL)
- SA Applications:
  Distributed optimization and control, weather data analysis, maglev control experiment, diffusion-reaction morphogenesis
- References
Cognitive Foundation and Related Work

- Development of Object Perception (Spelke et al.)
- Visual Routines (Ullman)
- Visual and Spatial Reasoning

Object Perception

How does our perceptual system extract and track discrete objects from cluttered environments?

Two aspects of the problem:

- Individuation and unit formation: what counts as a single entity and how to carve up a scene into distinct bodies?
- Object persistence (identification and correspondence): how do multiple descriptions, over time or space, pertain to a single entity?
An example: Perception of object boundaries
A car and a trailer, or a bumperless car-trailer and a bumper?

A yellow duck on a red truck or a duck-like truck?

Adapted from Spelke et al., 1995

Objects

- **Cohesion**: An object is internally connected and externally bounded; it maintains its connectedness and boundedness over time and space.

- **Continuity**: An object exists continuously and moves on a path that is connected over time and space.

How do we extract meaningful objects from a continuous field based on spatio-temporal cohesion and continuity?
Infants' Perception of Objects

Three Principles of Spelke et al.:

- Cohesion: A moving object maintains its connectedness and boundaries

- Contact: Objects move together if and only if they touch

- Continuity: A moving object traces exactly one connected path over space and time
Ullman’s Visual Routines

- A set of computational routines for analyzing shape properties and spatial relations such as elongation, inside/outside relation, and connectedness.

  E.g. Boundary tracing, area activation, counting intersections.

- Does not require object recognition

Examples of spatial properties/relations:

- Inside-outside
- Elongation
- Closure
- Pointiness

Requirements

- Abstractness: establish abstract properties and relations
  - *Support* for a property or relation: the set of points that give rise to the property or relation.
  - Abstract property or relation is one with non-local support.
    - E.g. Inside/outside relation; Closure property.

- Open-endedness: establish a large variety of relations and properties
  - Due to computational feasibility considerations
  - Different routines are assembled from a small number of elementary routines
  - New routines are assembled from the same set to meet additional computational goals
• Complexity: efficiency in using space and time
  – Use the same set of routines for different tasks
  – Apply the same computation to different locations
  – Need selective attention and shift of focus.

Computational Considerations

• What are the base set of routines?
• How are complex routines built out of simpler ones?
• Where and how are routines triggered?
• How are routines sequenced?
Qualitative Spatial Reasoning

- MD/PV theory (Forbus et al., 1991)
  - Metric Diagram (MD): numerical and symbolic descriptions of a scene
  - Place Vocabulary (PV): a quantization of the space according to task-specific criteria
- A qualitative model for distributed parameter physical fields (Lundell, 1996)
  - Region partition; quantity space; influence propagation
- Spatial hierarchy (Kuipers and Levitt, 1988)
  - Four-level topological and metric descriptions
  - Robot navigation and mapping of spaces

- Spatial calculus (Randell, Cui, and Cohn, 1992)
  - Region-connection calculus (RCC) based on binary topological relations
Diagrammatic reasoning

- Gelernter geometry theorem prover (Gelernter, 1963)
- Nevins' geometry theorem prover (Nevins, 1975)
- Constraint propagation (Stallman and Sussman, 1977)
- Diagrammatic representation (Larkin and Simon, 1987)
- A comprehensive collection of papers (Glasgow, Narayanan, Chandrasekaran, 1995)

Analogue simulation

- WHISPER (Funt, 1980)
- "Molecular" simulation (Gardin and Meltzer, 1989)
- Direct motion simulation (Chandrasekaran and Narayanan, 1990)
Tutorial Roadmap

- Introduction and Motivation
- Overview of Spatial Aggregation (SA):
  Imagistic reasoning, theory, ontology, language
- Examples of Spatial Aggregation:
  KAM, MAPS, HIPAIR, Fluids
- Cognitive Foundation and Related Work
  ▶ Spatial Aggregation Language (SAL)
- SA Applications:
  Distributed optimization and control, weather data analysis,
  maglev control experiment, diffusion-reaction morphogenesis
- References

SAL Introduction

Goal: to support programming in the style of SA.

- Data types and operations at the right level of abstraction.
- Interactive programming environment for exploring structures
  in physical data.
SAL Data Types Overview

- Primitive object: representation of a physical object.

- Compound: collection of primitive objects or compounds. Particularly useful compounds:
  - Sets: Spaces, equivalence classes, etc.
  - Relations: Fields, Ngraphs, topological substructure, etc.

- Abstraction: compound → higher-level primitive.

Physical Objects

Meteorology: sample points, isobar curves, pressure cells
Physical Objects:
Structure vs. Geometric Properties

- Topological structure specifies how parts are related.
- Geometric properties (e.g. edge length, angle, curvature, area) depend on metric, coordinate system, etc.
- Ex: same structure, different coordinate system or coordinates:

![Diagrams showing different coordinate systems with the same structure.]

- Separate data types for structure and geometric properties.

Topological Structure

Specify how a physical object’s parts are related to each other:

- Implicitly by geometric construction
  Ex: $x^2 + y^2 \leq r^2$
- Explicitly by relations among components
  - Space/subspace relation; e.g. cube and its faces
  - Adjacency; e.g. collection of triangles in a mesh
- Cell Complex is a particular hierarchical explicit representation.
Cell

- Homeomorphic (continuously deformable) to ball of same dimension.

- Ex: point, closed line segment or curve segment, triangle with interior or surface patch, solid cube.

- Hierarchical structure:
  - A cell has faces that are lower-dimensional cells.
  - A cell’s proper faces are its faces of the next lower dimension.

Example Cells
Cell Complex

Collection of cells such that

1. Each cell's faces are in the complex.
2. A non-empty intersection of two cells is a face of each.
   
   - Legal cell complexes:

   ![Legal cell complexes diagram]

   - Illegal cell complexes:

   ![Illegal cell complexes diagram]

Atomic Cell Complexes


- *Non-atomic* cell complex: a complex for multiple cells.
Example Cell Complex Relationships

Example Cell Complex Queries

Queries for structure of cell complex (only proper relationships illustrated above):

- **Face**: cell structure
  Ex: $ABCD \rightarrow AB \rightarrow A$

- **Co-face**: inverse of face
  Ex: $A \rightarrow AB \rightarrow ABCD$

- **Adjacency**: share a face
  Ex: $AB \rightarrow BC$
Geometric Properties

Geometric properties for a structure depend on distance function, coordinate system, etc.

- Point: coordinates
- Segment: length
- Curve: length, curvature at given points
- Quadrilateral: angles, edge lengths, area

*Geometric Objects* derived from structure objects cache appropriate properties.

Metric Space

A *Metric Space* defines a distance-measuring function (*metric*) and reference frame relative to which geometric properties are defined.

- *Coordinate Space* does this in terms of a specific coordinate system.
- Example metrics:
  - 2-D Euclidean:
    \[ d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \]
  - Distance between polygons based on centroids.
Geometry Construction

Given a structure and a metric space, establish geometric properties for the structure.

- Algebraically
  Ex: \( x^2 + y^2 \leq r^2 \)

- Bottom-up, from geometries of substructure
  Ex: Given lengths of triangle's sides, compute angles and area.

- Top-down, establishing geometries of substructure
  Ex: Given angles and area of triangle, compute lengths of sides.

Local and Embedding Metric Spaces

- A geometric object defines a local metric space for its substructure.
  Ex: 1-D parameterization of a curve, 2-D parameterization of a square

- A geometric object can be embedded in another metric space.
  Ex: 2-D square embedded in a 3-D cube
Transformations

Geometric properties are relative to a metric space, but can often be easily transformed from one metric space to another, related one.

- From local metric space to embedding metric space
  Ex: point's coordinates on edge to coordinates in rectangle

  ![Diagram of point's coordinates on edge to coordinates in rectangle]

  \[ t = 0.5 \]
  \[ (x,y) = (0,0.5) \]

- From embedding metric space to local metric space
  Ex: point's coordinates in rectangle to coordinates on edge

- Between metric space and transformed version
  Ex: 2-D Euclidean to rotated 2-D Euclidean (area, angles, etc. remain constant; coordinates change)

  ![Diagram of 2-D Euclidean to rotated 2-D Euclidean]

  \[ (x,y) = (2,2) \]
  \[ (u,v) = (3,0) \]
Physical Object Representations: Summary

- Separate structure from geometric properties.
- Define structure hierarchically with cell complex.
- Define geometric properties for given structure and metric space.

Compounds

A Compound is a collection of primitive objects or compounds.

- Flat compounds: collections of primitive objects
- Structured compounds: collections of other compounds
  Ex: a relation is a set of pairs of objects.

SAL includes a number of predefined compound types with special semantics.
Space

A *Space* is a collection of primitive objects.

- Select sub-spaces satisfying predicate.
  Ex: all quadrilaterals created so far
  Ex: all faces of a particular cube

- Distribute operations over objects in space.
  Ex: for each quadrilateral, find its area.

- Extract global property of collection.
  Ex: find average area of all quadrilateral objects.

Metric Space Indexing

In addition to defining a metric, a Metric Space can also cache indexing information for spatial queries on its objects.

- Nearest object to a given object (or coordinates).

- Objects within some given distance of an object (or coordinates).

Common indices include arrays, *k*-d trees, etc.
Field

- Mapping $\mathbb{R}^n \rightarrow \mathbb{R}^m$.
- Associates feature points with position points.
  Ex: $\{(\text{point, temperature})\}$ $(\mathbb{R}^2 \rightarrow \mathbb{R}^1)$
  Ex: $\{(\text{point, wind velocity})\}$ $(\mathbb{R}^2 \rightarrow \mathbb{R}^2)$
- Element-wise operations (implicitly distributed):
  Ex: add/scale fields
- Field values can be interpolated.

Ngraph

- Set of neighborhood adjacencies relating objects in a space:
  $\{(\text{object, neighbor})\}$.
- Adjacencies localize computation:
  - Local comparisons
    Ex: how does the wind direction here compare to that at neighbors?
    Ex: how far away is this neighbor compared to others?
  - Local interaction rules
    Ex: update temperature at a node based on temperatures at surrounding nodes.
Ngraph Manipulation

- Explicit construction: specify neighbors for each object.
- Implicit construction: specify criterion.
  Ex: delaunay triangulation or $k$-nearest neighbors
- Graph-theoretic operations manipulate structure.
  Ex: union, intersection, subgraph, closure, connected components, search, neighborhood walk

Ngraphs and Cell Complexes

Some Ngraphs have corresponding Cell Complexes:

- Ex: minimal spanning tree builds segments corresponding to adjacencies between points.

- Ex: mesh builds elements with points as vertices and edges corresponding to adjacencies.
Grid

Rectilinear geometries in *grid* support directional neighbor queries.

![Diagram](image)

neighbor along x-axis, positive direction

---

Ngraph Equivalence Classes

- SA theory provides the *classify* operator to find equivalence classes of objects in Ngraphs.

- Ex: isothermal regions with temperatures in the same bin. Ex: trajectories in the wind vector field.

- An *equivalence class* is the set of objects in the transitive closure of the intersection of a neighborhood relation with an equivalence predicate.

- Goal: an efficient mechanism for computing equivalence classes.
Ngraph Equivalence Class Mechanism

1. Localize computation with Ngraph.

2. Eliminate inconsistent adjacencies.

3. Find connected components in resulting graph.

Compounds: Summary

- *Space* treats objects as a group.

- *Field* relates objects and features embedded in a physical continuum.

- *Ngraph*
  1. Explicates object adjacency.
  2. Localizes computation.
  3. Supports efficient extraction of coherent groups.
Abstraction

Given a group of objects and their structure, abstract them into a single higher-level primitive object.

- Ex: bottom-up construction of Cell Complexes.
- Ex: redistribution of equivalence classes in Ngraph.

- Additional operations make sense on the higher-level object.
  Ex: curvature of a curve segment previously represented by sampled points.
- Associated features can "tag along."
  Ex: average temperature in region

SAL Library Summary

A C++ library provides a number of implementations and operations for SAL data types.

- Primitive objects
  - Represent localized chunks of space and associated features.
  - Cell complexes
    * Point, segment, n-line, n-gon
    * Face, co-face, adjacency
  - Geometric objects
    * Point, segment, polyline, polygon
    * Coordinates, volume
• Compounds
  
  – Organize groups of primitive objects.
  – Space
    * Topological space, Euclidean metric space
    * Map, filter, nearest member
  – Field
    * Scalar field, vector field
    * Add, scale, combine, interpolate
  – Ngraph
    * Delaunay, MST, k-nearest, grid
    * Union, subgraph, connected components

• Abstraction

  – Map groups of objects and their structure into single aggregate objects that can be treated as primitives.
  – Cell complex construction, convex hull, spline construction
  – Redescribe, localize
SAL Programming Environment Demo

SAL provides an interpreted, interactive programming environment, layered over the library implementations.

Demo: compute paths of heat flow.

1. Localize computation with an ngraph.
2. Simulate diffusion with local interaction rules.
3. Calculate gradient vectors.
4. Filter ngraph, preserving edges roughly parallel to vectors.
5. Group resulting edges into curves.

```plaintext
// Read locations for data
points = read_points("p.in")
// Localize computations with an ngraph
trig = aggregate(points, delaunay)
adj_8_graph = aggregate(points, near(1.5))
grid = aggregate(points, regular_grid(1))
```
// Establish a single source location and its state
src = graphical_select(points)
sources = set(src); source_value = state(sources, init_val=1)

// Establish a field of temperature values for the points; simulate diffusion
temp = value_field(points, init_val=0)
interior = filter(p in points, size(neighbors(grid, p)) == 4)
update_converging(p in interior, temp,
    average(map(neighbors(grid, p), temp)) +
    exists(source_value(p)) ? source_value(p) : 0)

// Form isotherms by classifying neighboring points as to whether or not
// their temperatures fall in the same bin
temps = map(points, temp); dt = max(temps) - min(temps)
isotherms = classify(a in adj_8_graph, same_bin(a, temp, width=dt/5))
// Build gradient vector field by estimating derivative
grad = vector_field(p in interior,
  vector((temp(directional_neighbor(grid,p,0,1)) -
      temp(directional_neighbor(grid,p,0,-1)))/2,
  (temp(directional_neighbor(grid,p,1,1)) -
    temp(directional_neighbor(grid,p,1,-1)))/2))

// Normalize for visibility
grad_dir = vector_field(p in interior, normalize(grad(p)))

// Compare gradient vector direction with direction from point to neighbor
to = filter_ngraph(a in grad_graph,
  dot(grad_dir(node(a)),
    normalize(subtract(node2(a), node1(a))))
  > 0.9)
/ Find to-neighbor with most-similar gradient vector; break ties with
// distance between point and neighbor
best_to = aggregate_explicit(p in interior,
    set(max(n in neighbors(to, p),
        dot(grad_dir(p), grad_dir(n))
    - 0.1*distance(p, n))))

// Do same thing in opposite direction [slided]
// Combine the two best graphs
both = union_ngraph(best_from, best_to)

// Find matches where o1's best to-neighbor is o2, and o2's best
// from-neighbor is o1; i.e. the symmetric subgraph
matches = make_symmetric(both)
// Jump an abstraction level: convert classes of connected points into curves
point_classes = classify(matches)
trajs = redescribe(point_classes, path_to_curve)

// Find the average gradient magnitude along the curves
traj_mag = state(t in trajs, integrate_path(grad_mag, t))

SAL Programming Style

1. Formulate problem.
   - Describe the task in terms of geometric/topological structures.  
     Ex: in trajectory bundling (running example), the task is to 
     determine qualitatively-similar states. Geometrically, the states 
     are represented as points and their qualitative behaviors as 
     trajectory bundles.
   - Input: spatially-distributed data, captured as a field.  
     Ex: sample points in state space.
   - Output: high-level, abstract descriptions derived from the data.  
     Ex: trajectory bundles.
SAL Programming Style

2. Identify relevant domain knowledge.

- Continuity and different spatio-temporal scales give rise to objects in the data.
  Ex: objects include sample points, trajectories, and bundles of trajectories.

- Objects have different relationships, e.g. adjacency and structure/substructure.
  Ex: objects have geometric nearness/adjacency relationships; points comprise trajectories which comprise bundles.

- Objects interact and evolve.

SAL Programming Style

3. Identify layers of abstraction.

- Build a hierarchy relating objects of different granularities.
  Ex: points → trajectories → bundles.

- Describe abstraction transformations and pre-/postconditions for application.
  Ex: abstract linearly-connected points into curves and "similarly-shaped" adjacent curves into bundles.
SAL Programming Style

4. Flesh out internals of each abstraction layer.

- Guided by layers of abstraction: connect output of previous layer to input of next layer.

- Build a neighborhood graph explicating an adjacency relation based on domain knowledge and structure of output. 
  Ex: relate points with MST (close to linearly-connected); relate trajectories based on connected substructure.

- Filter the neighborhood graph based on predicates from domain knowledge and abstraction preconditions. 
  Ex: eliminate long edges in MST; test curve similarity.

Tutorial Roadmap

- Introduction and Motivation
- Overview of Spatial Aggregation (SA): Imagistic reasoning, theory, ontology, language
- Examples of Spatial Aggregation: KAM, MAPS, HIPAIR, Fluids
- Cognitive Foundation and Related Work
- Spatial Aggregation Language (SAL)
  > SA Applications: Distributed optimization and control, weather data analysis, maglev control experiment, diffusion-reaction morphogenesis
- References
SA Applications Roadmap

- Distributed optimization and control (C. Bailey-Kellogg)
- Weather data analysis (X. Huang)
- Maglev control experiment (J. May and S. Loh)
- Qualitative analysis of diffusion-reaction systems (I. Ordóñez)

Thermal Material Processing

C. Doumanidis, IEEE Control Systems, August 1997

- Task: rapid prototyping for thermal fabrication (welding).
- Set-up:
  - Plasma-arc heat source; tungsten torch
  - High-speed servodriven X-Y positioning table
  - 256x200 (0.12mm resolution) infrared camera
- Approach: feedback control on linearized model; parameters identified at run-time.
Rapid Thermal Processing

T. Kailath et al., in *The Control Handbook*, 1996

- Task: to maintain a uniform temperature distribution to ensure high yield for semiconductor curing.

- Set-up: Concentric rings of lamps; separate power control for each zone.

- Approach: feedforward control with feedback tuning on linearized model.
Rapid Thermal Processing Set-Up

T. Kailath et al., in *The Control Handbook*, 1996

---

Our Motivating Example: "Local Warming"

- **Design task**: determine the placement and actions of controls to achieve a desired temperature profile.
- **Input**: geometry, boundary conditions, material properties, design constraints
- **Output**: number, locations, and control actions for heat sources
Approach

- Model domain knowledge in SA framework.
- Find structures in temperature data.
- Exploit structures to determine control locations.
- Exploit structures to determine control parameters.

Mathematical Models for Heat

- Steady-state: asymptotic temperature distribution
  \[ k \nabla^2 \phi + Q = 0 \]
  * \( \nabla^2 \) is the Laplace operator.
  * \( \phi \) is the temperature field.
  * \( k \) represents the material properties.
  * \( Q \) is the contribution from heat sources.
  - Temperature and heat source contribution are functions of spatial variables.
- Transient: temperature profile over time
  \[ \frac{\partial \phi}{\partial t} = k \nabla^2 \phi + \dot{Q} \]
  - Temperature and heat source contribution are functions of spatial variables and time.
SA Model for Temperature Computation

- **Field**: \{ (point, temperature) \}
  
  For transient, also discretize in time.

- **Neighborhood graph**: finite difference grid or finite element mesh

- **Object interaction rules**: propagation based on heat equation, with heat source contributions at discrete locations/times.

SA vs. Traditional Engineering Methods

SA provides a qualitative reasoning approach to programming with physical fields:

- The structure of a field discretization is explicitly represented.

- Behaviors are inferred using a small number of operations on the field structure, so that results can be explained in terms of local object interaction and evolution.

- Objects in a field can be manipulated at multiple, non-uniform layers of abstraction.

As a result, SA supports a variety of inference, explanation, tutoring, and design tasks.
Structures in Temperature Data

Control design will utilize structures uncovered in temperature data:

- Rates of decay in different directions indicate rates of heat flow.
- Effects of controls are predominantly local.
- Effects of controls can be linearly superposed.

Structures in Temperature Data: Isotherms

Use SA classify operator to find equivalence classes of similar temperature values in neighborhood structure.
Related Work: Qualitative Physical Fields

Lundell (AAAI-96) provides a qualitative model for processes such as diffusion.

- Represent topology of qualitatively-similar regions in fields.
- Intersect fields (e.g. temperature field and shade field).
- Model spatio-temporal evolution of fields with region interactions.

Structures in Temperature Data: Gradient Vector Directions

Use SA field operations to calculate gradient vectors. Direction points downhill; magnitude represents rate of change.

(Lengths normalized for visibility.)
Structures in Temperature Data: Gradient Trajectories

Use SA *classify* operator to group gradient vectors into trajectory curves based on similarity in direction.

Curves are downhill paths through the temperature landscape; rate of descent indicates rate of heat flow.

---

Structures in Temperature Data: Thermal Hills

Temperature decays away from heat source location:

- Predominantly local effects for controls.
- Different hill shapes at different points.
- Different hill slopes in different directions.
Thermal Hill Linearity

- **Scalability:** Scaled thermal hill is equivalent to thermal hill with source value scaled.

\[
10 \times \begin{array}{c}
\text{(source = 1)}
\end{array} = \begin{array}{c}
\text{(source = 10)}
\end{array}
\]

- **Superposability:** Sum of thermal hills is equivalent to thermal hill with both sources active.

\[
\begin{array}{c}
\text{(source 1 on)}
\end{array} + \begin{array}{c}
\text{(source 2 on)}
\end{array} = \begin{array}{c}
\text{(both sources on)}
\end{array}
\]

Influence Graph

Abstract thermal hill for field nodes \( F \) and control nodes \( C \):

- Vertices \( V = C \cup F \).
- Edges \( E = C \times F \).
- Edge weights \( w : E \rightarrow \mathcal{R} \) such that \( w((c, f)) \) is the field value at \( f \) given a unit control value at \( c \).
- A thermal hill is a pictorial representation of an influence graph from one control node.
**Field Nonlinearities**

- Material properties can vary nonlinearly with position.
- The temperature is still linearly dependent on control values.
- Discretization of heat equation yields linear equations relating temperature and heat source input at a node with temperatures at neighboring nodes.
- An influence graph exposes linear dependence on control values, encapsulating the nonlinearities in field.
- Edge weights measured for a physical system represent a form of system identification, where exact material properties are unknown.

**Viewpoints on Influence Graph**

- Physicist: discretized Green's function.
- Engineer: distributed impulse response, transfer function.
- Mathematician: (decentralized form of) the inverse of the capacitance matrix modeling heat diffusion.
Summary of Structures in Temperature Data

- Aggregated gradient vector trajectories represent directions/rates of heat flow.
- Influence graph encapsulates locality and linear superposability of control effects, hides nonlinearities in field.

Structure Design

- Goal: determine number and positions of controls.
- Approach: decompose the problem so as to minimize the coupling between subparts.
Structure Design: Physical Insight

- Dumbbell-shaped material heated at both ends:

- Fourier's law of heat conduction: heat flux is proportional to temperature difference.

- Therefore, sparse isotherms (small temperature decay) indicate small heat flux and thus weak coupling.

- Decompose along direction of small decay.

Decomposition Algorithm

- Subtract
- Classify on direction
  - Compare average gradient magnitude
- Partition along direction

Weak coupling direction
Control location
Decomposition Example: Isotherms

A single heat source placed near the center of mass does not adequately control the entire sheet. However, as with dumbbell-shaped material, isotherm separations indicate amount of coupling.

---

Decomposition Example: Gradient Trajectories

Examine gradient magnitudes along trajectories to find weak directions. Partition field along weak trajectories.
Decomposition Example: Partition

This partition corresponds to our geometric intuition; the method works even for varying material properties, where geometry isn’t enough.

Structure Design Summary

- Structures in data (gradient magnitudes along trajectories) indicate weak coupling directions.
- Principled method to decompose problem: partition so that sub-problems are weakly coupled.
SA Decomposition vs. Domain Decomposition

- Domain decomposition algorithms solve partial differential equations by dividing a field into subregions and (iteratively) combining the solutions for the subregions.
- SA decomposition exploits the same kind of physical knowledge to partition a field.
- SA decomposition is performed in terms of structures in data (gradient trajectories).
- SA can explain design decisions in terms of these structures and the physical knowledge they represent.

Decentralized Parameter Optimization

- Given control locations from structure design:

- Goal: determine control values.
- Approach: SA local interaction rules between control nodes and field nodes.
- Algorithm:
  Repeat: independently adjust each control value so as to reduce error in temperature profile.
Decentralized Parameter Optimization Algorithm

![Diagram]

Page MP2-177

Improving Decentralized Parameter Optimization

Leverage structural knowledge (locality, linearity in control):

1. Evaluate temperature field efficiently.
2. Reduce communication.
3. Jointly optimize where appropriate.
Influence-Based Field Evaluation

At each optimization step, evaluate control’s effect on temperature.

- Standard approach: iteratively solve heat equation.
- Fast approach: exploit linear superposability.
  For modified control value, subtract old scaled influence, add new scaled influence:

  Subtract old    Add new
  (Equivalently, add influence scaled by control value difference.)
Influence-Based Field Evaluation
Performance Data

- P shape (1103 nodes); 4 controls
- C++-based SA library on a 100MHz Pentium / Linux / gcc
- 49 seconds to solve iteratively
- 0.02 seconds to solve with influence graph mechanism.
- Savings at each step of optimization!
  (Most tests took 20-100 iterations.)

---

Reduced Communication

At each step, estimate error due to control adjustment.

- Slow but accurate: check all field nodes.
- Faster but less accurate: check more strongly-influenced nodes more often:

- Trade control accuracy for communication.
Reduced Communication Algorithm

Reduced Communication Performance Tests

Optimizers:

- Matlab-based (centralized) optimizers: Gauss-Newton and Broyden-Fletcher-Golfarb-Shanno
- SA-based (decentralized) optimizers: SA1–SA4, communication proportional to influence (different proportionality constants)

Problems (20x20 grid):

- 4 heat sources near corners
- 4 heat sources near center
- 16 heat sources tiled over domain
Reduced Communication Performance Data

- Number of iterations to converge to solution.
- Total control-field communications during optimization.
- Error when converged.

<table>
<thead>
<tr>
<th>4-corner</th>
<th>GN</th>
<th>BFGS</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
</tr>
</thead>
<tbody>
<tr>
<td># iter</td>
<td>19</td>
<td>14</td>
<td>21</td>
<td>19</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td># comm</td>
<td>34624</td>
<td>18144</td>
<td>27215</td>
<td>10572</td>
<td>2332</td>
<td>1144</td>
</tr>
<tr>
<td>error</td>
<td>.2028</td>
<td>.2028</td>
<td>.2028</td>
<td>.2037</td>
<td>.2281</td>
<td>.252</td>
</tr>
<tr>
<td>4-center</td>
<td>20</td>
<td>14</td>
<td>24</td>
<td>21</td>
<td>19</td>
<td>31</td>
</tr>
<tr>
<td># iter</td>
<td>1.0</td>
<td>.7</td>
<td>1.2</td>
<td>1.05</td>
<td>.95</td>
<td>1.55</td>
</tr>
<tr>
<td># comm</td>
<td>25920</td>
<td>18144</td>
<td>31104</td>
<td>21004</td>
<td>.9880</td>
<td>.5188</td>
</tr>
<tr>
<td>error</td>
<td>.3459</td>
<td>.3459</td>
<td>.3459</td>
<td>.3463</td>
<td>.3621</td>
<td>.3922</td>
</tr>
<tr>
<td>15-tiled</td>
<td>50</td>
<td>213</td>
<td>35</td>
<td>49</td>
<td>46</td>
<td>75</td>
</tr>
<tr>
<td># iter</td>
<td>1.0</td>
<td>3.804</td>
<td>.6429</td>
<td>.875</td>
<td>.8214</td>
<td>1.339</td>
</tr>
<tr>
<td># comm</td>
<td>200304</td>
<td>1104192</td>
<td>186024</td>
<td>161700</td>
<td>50535</td>
<td>30031</td>
</tr>
<tr>
<td>error</td>
<td>.1104</td>
<td>.1104</td>
<td>.1107</td>
<td>.1181</td>
<td>.1189</td>
<td>.1347</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
<td>1.003</td>
<td>1.015</td>
<td>1.022</td>
<td>1.157</td>
</tr>
</tbody>
</table>
Reduced Communication Performance Data

Note long flat region where communication is reduced without significantly impacting error.

Page MP2-187

Joint Optimization

Controls may be more or less independent.

- Coupling:

- Actions of one affect potential actions of other.

- Might require more iterations to converge.

- Might converge to sub-optimal solution: independently adjusting either control value increases error, but increasing one and decreasing the other decreases error.
Joint Optimization Algorithm

1. Link control to neighboring controls (proximity or overlapping influences):

2. Establish “supervisor” that can shift control value from one control to another:

3. Optimize supervisors concurrently with other controls.
Joint Optimization Performance Tests

SA-coop optimizer: supervisors between neighboring control pairs

Problems:

- 4 heat sources packed near edge of 20x20 grid
- 16 heat sources packed near center of 20x20 grid
- 15 time-step heat sources of 5x5 grid

Joint Optimization Performance Data

<table>
<thead>
<tr>
<th></th>
<th>GN</th>
<th>BFGS</th>
<th>SA</th>
<th>SA-coop</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-packed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># iter</td>
<td>19</td>
<td>96</td>
<td>78</td>
<td>28</td>
</tr>
<tr>
<td>error</td>
<td>.6725</td>
<td>.6725</td>
<td>.6737</td>
<td>.6725</td>
</tr>
<tr>
<td>16-packed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># iter</td>
<td>56</td>
<td>207</td>
<td>167</td>
<td>70</td>
</tr>
<tr>
<td>error</td>
<td>.3357</td>
<td>.3357</td>
<td>.3371</td>
<td>.3357</td>
</tr>
<tr>
<td>15-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># iter</td>
<td>94</td>
<td>226</td>
<td>64</td>
<td>25</td>
</tr>
<tr>
<td>error</td>
<td>.1189</td>
<td>.1190</td>
<td>.1189</td>
<td>.1189</td>
</tr>
</tbody>
</table>
Joint Optimization Performance Data

Parametric Design Summary

- Structures in data (influence hills) expose locality, linearity in control.

- Principled method to trade off computation, communication, and control quality: exploit locality and linearity of control encapsulated in influence graph.
Generality of Techniques

- Locality assumption is not valid for highly conductive materials — not easily decomposable.
- Where locality holds, these techniques scale well by exploiting it.
- Linearity in control applies to many processes (heat conduction, gravity, electrostatics, incompressible fluid flow).

Summary

Use SA structure-based design for control of heat.

- Distributed ("local warming") framework.
- Explicitly use physical knowledge (weak coupling directions, locality, linearity in control).
- Structures make explicit where trade-offs can be made.
- Structures are useful for explanation.
SA Applications Roadmap

- Distributed optimization and control (C. Bailey-Kellogg)
- Weather data analysis (X. Huang)
- Maglev control experiment (J. May and S. Loh)
- Qualitative analysis of diffusion-reaction systems (I. Ordóñez)

Multi-Level Structures in Weather Data

Sample points, isobar curves, pressure cells
Weather Map Analysis


- "At 850mb, the polar front is located parallel to and on the warm side of the thermal packing."

- "Identify 700mb trough and ridge positions to establish vertical consistency (stacking) with surface and 850mb features."

- "Major and minor 500mb troughs are good indicators of existing or potential adverse weather."

- "Lows tend to stack toward colder air aloft while highs tend to stack toward warmer air aloft."

Key Observation in Weather Analysis

- Think of weather features as geometric objects with spatial distribution and movement.

- Identify properties of these objects (shape, size, density).

- Rules of thumb correlate different weather features and establish prediction patterns.
Structures in Weather Data: Thermal Packing

Closely-packed isotherms:

![Isotherms Diagram]

Structures in Weather Data: Pressure Troughs

Sharply-bending isobars:

![Isobars Diagram]
Structures in Weather Data: Pressure Cells

Extremal pressure regions:

Weather Interpretation Tasks

Identify and describe features in weather data:

- Find pressure cells, troughs, ridges, fronts, ....
- Extract geometric properties: location, shape, size, ....
- Describe features at multiple layers of granularity.

Task characteristics:

- The problem is ill-defined and difficult to quantify.
- Massive amounts of spatially-distributed data — efficient algorithms required.
Our Motivating Example:
Pressure Cell Identification

- Task: given pressure data, find low/high pressure cells.
- SA-based approach
  1. Build local topology for isobars.
  2. Extract appropriate extremal isobars.
- Unlike numerical thresholding techniques, the same SA approach applies to other structure finding problems in weather data (locating trough/ridges, fronts)

Isobar Topology Construction

Key insight: distinguish two types of spatial adjacency on input data.
Strong Adjacency

A *strong* adjacency bonds lower-level objects into higher-level objects.

Weak Adjacency

Use remaining *weak* adjacencies to form topology on higher-level objects.

Strength of connection can be a factor.
Cell Identification

- Cells are local extrema with neighbors on only "one side."
- Ex:

  ![Iso-curves](image1)
  ![Topological N-graph](image2)

- Find local extrema that aren't articulation points (points whose removal would disconnect the graph).

A Structure Finding Algorithm

1. Spatial Objects → aggregate → N-Graph Relations
2. classify
   - Strong Adjacencies
   - Weak Adjacencies
3. re-describe
4. aggregated spatial objects
5. adjacency-aggregate → aggregated N-Graph Relations

Page MP2-210
Extended Example: Pressure Data Points

Extended Example: Point Ngraph

Aggregate points with Delaunay triangulation.
Extended Example: Strong Adjacencies

Classify edges connecting points with the same pressure value.

Extended Example: Isobars

Redescribe points and strong adjacencies into curves.
Extended Example: Isobar Network

Aggregate isobars based on weak adjacencies between underlying points.

Extended Example: Cells

Identify isobars that are local extrema in terms of pressure and are not articulation points.
Summary

A computational mechanism for identifying and extracting implicit structures in spatial data:

- Identify strong and weak adjacency relations among spatial objects.
- Aggregate strong adjacencies to construct higher-level objects; aggregate weak adjacencies to relate higher-level objects.
- Organize spatial information in a structured model for further processing.
- The mechanism has been shown useful in labeling high/low pressure cells.

SA Applications Roadmap

- Distributed optimization and control (C. Bailey-Kellogg)
- Weather data analysis (X. Huang)
- Maglev control experiment (J. May and S. Loh)
- Qualitative analysis of diffusion-reaction systems (I. Ordóñez)
Problem Statement

Focus on the stabilization problem:

- Given: a physical system and a goal state (or set of goal states)
- Objective: derive a control law to drive the system into one of the goal states

Goals

- To supplement standard linear techniques when such techniques are inadequate.
- To replace the analytical analysis of current nonlinear techniques with computational exploration.
- To improve control by exploiting geometric features of phase space.
- To make explicit the tradeoffs between various control criteria.
Phase-Space Control Framework

- Phase-space geometric analysis
- Cell-based mapping
- Multi-layer spatial aggregation
Example: Behavioral Boundaries

A "behavioral boundary" in phase space is an area of phase space where the field direction in neighboring cells differs substantially.

```
Goal

"Bad region"
```

Geometric Interpretation of Performance

- **Controllability**: the set of reachable states forms a region in phase space
- **Stability**: set of initial states that evolve to same limit set
- **Robustness**: certain types of uncertainties can be modeled as regions
- **Optimality**: considerations on resource consumption parameterize phase-space trajectories
The Maglev Experiment

- Highly nonlinear system
- Has practical applications in magnetically levitated transportation systems (i.e. EMS systems)

Model:

\[ \dot{x} = v \]
\[ \dot{v} = g - \frac{L_0 x_0 I^2}{2mx^2} \]
Data Flow in Synthesizing Phase-Space Control Laws

1st Level

- Discretized phase-space
  - Aggregate
  - Physical adjacency
- Neighborhood graph
  - Classify
  - Behavioral boundary criterion
- Behavioral classes

2nd Level

- Localize
  - Discretized phase-space with class information
  - Aggregate
  - Adjacency based on dynamics
- Neighborhood graph with weighted edges
  - Classify
  - Connectivity criterion
- Controllable and uncontrollable classes

Search

Control table

Shortest path search
Instantiation of SA Framework

- Phase and control space discretized uniformly.
- Behavioral boundary is used at first level classification.
- Adjacencies at 2nd level are determined using 4th-order Runge-Kutta integrator.
- Edges at the 2nd level are weighted by the distance to the nearest behavioral boundary.
- The graph is searched for minimal distance paths using an iterative method.
Synthesized Control

Control Graph Plot

Related Work

- Hsu: Cell-based mapping
- Bradley and Zhao: Phase-space control synthesis algorithms
- Caines and Wei: Dynamically consistent hybrid systems
- Control of chaos
Summary

- SA provides a framework for exploiting geometric features of the phase space.
- Phase-space features can be used to derive useful performance metrics.
- This framework has been successfully applied to a practical application (maglev).

Open Questions

- What other geometric features are useful for control and analysis?
- How can the framework be efficiently applied to high-dimensional systems?
- How can the framework be expanded to handle poorly-modeled systems or systems that change over time?
SA Applications Roadmap

- Distributed optimization and control (C. Bailey-Kellogg)
- Weather data analysis (X. Huang)
- Maglev control experiment (J. May and S. Loh)
- Qualitative analysis of diffusion-reaction systems (I. Ordóñez)

Introduction:
Morphogenesis and the Diffusion-Reaction Model

- Morphogenesis: how do organisms grow into particular shapes?
- Turing model for pattern development in space:
  - diffusion-reaction
  - Represent two or more substances that diffuse and react with each other in space, changing their concentrations during the process.
  - When diffusion rates are different and reaction processes are appropriate, the concentration of substances tends to form spatial and temporal patterns (Turing patterns).
Understanding Turing Patterns: Beyond morphogenesis

- Diffusion-reaction models describe numerous natural phenomena, including morphogenesis, well-known chemical processes, population dynamics, and formation of large structures in the universe.

- Understanding this set of phenomena is important in a variety of scientific endeavors.

- Although mathematically simple, Turing patterns have been studied analytically with very limited success. Numerical simulation is often used to elaborate the model behaviors.

- Our goal is to attain a systematic qualitative description and classification of these phenomena through computational means.

Models of Diffusion: The heat equation

Diffusion is governed by the heat equation:

\[
\frac{\partial u}{\partial t} = D_u \nabla^2 u
\]

(1)

- \(\nabla^2\): the Laplace operator

- \(u\): a scalar field representing intensity or concentration

- \(D_u\): diffusion rate

Irregularities in the field intensity tend to even out, and the substance spreads through space.
Diffusion-Reaction with Two Substances: A generic model of pattern formation

Two heat equations are coupled by reaction components:

\[
\frac{\partial u}{\partial t} = D_u \nabla^2 u + f_u(u, v)
\]

\[
\frac{\partial v}{\partial t} = D_v \nabla^2 v + f_v(u, v)
\]

Typically, the substance that diffuses faster is continuously fed into the system, and is consumed by the reaction, a process that produces the other substance.

The Gray-Scott Model: A model of glycolysis

The Gray-Scott model for a real chemical process:

- Two substances, U and V, with the transformations:
  - \(U + 2V \rightarrow 3V\): U is transformed into V.
  - \(V \rightarrow P\): V is lost by transformation into an inert compound, at a rate K.
  - U is constantly fed into the system, both U and V are removed from the system by the feed process.
  - U diffuses faster than V.

- The model:

\[
\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1 - u)
\]

\[
\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F + k)v
\]
Simulation of Gray-Scott Model in 2D: Complex behaviors from simple model

- Gray-Scott model exhibits a wide variety of behaviors, depending on the choice of its two parameters.
- These behaviors have been observed both in chemical experiments and numerical simulations, and they range from stable hexagonal arrays to chaotic motion of unstable shapes.

Qualitative Analysis and Spatial Aggregation: Identify and extract coherent objects in fields

- Represent the implicit structure.
- Form a theory of object interaction and evolution.
- Explicate cause-effect relationships among objects.
- Spatial Aggregation is well suited for finding structural organization in fields.
Aggregation through Sampling:
The embedded field heuristic

- Embed the field with sampling particles (floaters):
  Particles "float" in the field, repelling each other to minimize
  an energy measure, multiplying in regions of low density and
  perishing whenever their density is too high.

- Spatial aggregation aggregates floaters to form a structural
  description of the underlying field

- Floaters tend to persist in time, except at bifurcations

Example: An N-Graph on a sampled field

- Floaters form groups that tend to mirror structures in the field

- Aggregation of floaters produces discrete polygons that
  approximate coherent objects

- Floaters vary their positions as the field changes

- Floater persistence simplifies structural correspondence over
  time
Tracking Structures in Time:
An illustration of the effect of floater persistence

- Explicit registration and representation of objects
  - A body is shown splitting into two separate objects.
  - Floaters modify their positions to reflect changes in the objects.
- Further processing uses such representation: discovery of causal relations; temporal histories of objects

Summary

- Studied a particular Diffusion-Reaction model
- Developed the "embedded field" method to sample and register implicit structures in a continuous field in concisely
- Persistence of sampling floaters simplifies structural correspondence over time
Tutorial Roadmap

- Introduction and Motivation
- Overview of Spatial Aggregation (SA):
  Imagistic reasoning, theory, ontology, language
- Examples of Spatial Aggregation:
  KAM, MAPS, HIPAIR, Fluids
- Cognitive Foundation and Related Work
- Spatial Aggregation Language (SAL)
- SA Applications:
  Distributed optimization and control, weather data analysis,
  maglev control experiment, diffusion-reaction morphogenesis

> References

References for Further Readings

- Qualitative Physics ontologies:
  - J. DeKleer and J.S. Brown, A qualitative physics based on confluences.
  - D. Weld and J. DeKleer (eds.) *Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, 1990
  - Qualitative Reasoning Home Page:

- Intelligent Simulation
  - H. Abelson, M. Eisenberg, M. Halfant, J. Katzenelson, E. Sacks, G.J.
Spatial Aggregation and imagistic reasoning:


KAM:


MAPS and automated phase-space analysis:

Page MP2-251


HIPAIR:


Interpreting fluid simulation:


• SA application: distributed optimization and control

- C. Bailey-Kellogg and F. Zhao, “Qualitative Analysis of Distributed Physical Systems with Applications to Control Synthesis.” *AAAI-98.*


---

• SA application: weather data analysis


• SA application: maglev control algorithm and experiment


SA Applications: diffusion-reaction morphogenesis


Cognitive Foundation and Related Work


- Spatial data mining:


Some of the above papers are online at http://www.cis.ohio-state.edu/insight