Spatial Computing
& the challenge of
Engineered Emergence

Jacob Beal
(& Jonathan Bachrach)
MIT CSAIL
Networked devices are **filling** our environment...
Networked devices are filling our environment...
Networked devices are filling our environment...

How do we program aggregates robustly?
Outline

- What is Spatial Computing?
- Continuous Space Programs
- Proto & Amorphous Medium
- Engineered Emergence
Spatial Computers

Robot Swarms

Biological Computing

Sensor Networks

Reconfigurable Computing

Cells during Morphogenesis

Modular Robotics

Definition

Continuous

Proto

Emergence
More formally...

• A spatial computer is a collection of computational devices distributed through a physical space in which:
  – the difficulty of moving information between any two devices is strongly dependent on the distance between them, and
  – the “functional goals” of the system are generally defined in terms of the system's spatial structure
More formally...

- A spatial computer is a collection of computational devices **distributed** through a physical space in which:
  - the difficulty of moving information between any two devices is **strongly dependent on the distance** between them, and
  - the “functional goals” of the system are **generally defined** in terms of the system's spatial structure
Emergence

Continuous

Proto

Graphs

Crystalline (e.g. CAs)

Amorphous/Continuous

(w. Dan Yamins)

Definition

Continuous

Proto

Emergence
Emergence as the Proto Continuous

Graphs

Crystalline (e.g. CAs) to Amorphous/Continuous

Density to space complexity to jitter

Grain size

Definition → Continuous → Proto → Emergence
Emergence

Proto

Continuous

Definition

Crystalline (e.g. CAs)

Amorphous/Continuous

density

space complexity

jitter

grain size

spatial computing

(w. Dan Yamins)
Example App: Mobile Streaming
Geometric Program: Channel

Source \rightarrow \textbf{Continuous} \rightarrow \textbf{Proto} \rightarrow \textbf{Emergence} \rightarrow \textbf{Destination}

(cf. Butera)
Geometric Program: Channel

Definition → Continuous → Proto → Emergence

Source

Destination

(cf. Butera)
Geometric Program: Channel

Source  (cf. Butera)  Destination

Definition → Continuous → Proto → Emergence
Geometric Program: Channel

Source \rightarrow \textcolor{red}{\textbf{Continuous}} \rightarrow \text{Proto} \rightarrow \text{Emergence} \rightarrow \textbf{Destination}

(cf. Butera)
Geometric Program: Channel

Definition → **Continuous** → Proto → Emergence (cf. Butera)
Geometric Program: Channel

Definition → Continuous → Proto → Emergence

(cf. Butera)
Geometric Program: Channel

Definition \rightarrow \textbf{Continuous} \rightarrow \text{Proto} \rightarrow \text{Emergence} (cf. Butera)
Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness

(we'll come back to this in a bit...
Amorphous Medium

- Continuous space & time
- Infinite number of devices
- See neighbors' past state

Approximate with:
- Discrete network of devices
- Signals transmit state

Definition 
Continuous 
Proto 
Emergence
Computing with fields

source \arrow{->}\text{gradient} \arrow{->}\text{gradient} \arrow{->}\text{destination}

\text{distance} \arrow{<=>}\text{+} \arrow{->}\text{dilate}

\text{width} \arrow{->}\text{proto} \arrow{->}\text{Emergence}

Definition \arrow{->}\text{Continuous} \arrow{->}\text{Proto}
Computing with fields

source

gradient

+ 

<=

distance

<=

dilate

width

10

Definition  Continuous  Proto  Emergence
(def gradient (src) ...)
(def distance (src dst) ...)
(def dilate (src n)
  (<= (gradient src) n))
(def channel (src dst width)
  (let* ((d (distance src dst))
         (trail (<= (+ (gradient src)
                     (gradient dst))
              d)))
   (dilate trail width)))
Proto's Families of Primitives

- **Pointwise**
- **Feedback**
  - delay
  - +
  - 41
  - 7
  - 48

- **Restriction**
- **Neighborhood**
  - restrict
  - nbr
  - any-hood

Definition ➔ Continuous ➔ Proto ➔ Emergence
Modulation by Restriction

Definition → Continuous → Proto → Emergence
In simulation...
Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness
Other Approaches

- Lots of C hacking...
- Regiment / WaveScope
- Viral: TOTA / Smart Messages / Paintable Computing
- Pattern formation: [Coore] / [Nagpal] / [Kondacs] / [Stoy] / [Goldstein], ...
- Abstract: Kairos, EgoSpaces, Logical Neighborhoods, “views”, ...

Definition — Continuous — Proto — Emergence
On to Emergence...

• What is emergence?
  – Greater than the sum of its parts?
  – Unpredictable from local interactions?
  – Only definable for the aggregate?
“Engineered Emergence”

Routine design of the behavior of aggregates of unreliable devices with complicated interaction patterns.
“Engineered Emergence”

Routine design of the behavior of aggregates of unreliable devices with complicated interaction patterns.
Not quite...

- Nature's toolkit is still better

*Proto is good for library building: global-to-local + modulation by restriction*
1st: Flocking

13 lines of Proto

Definition → Continuous → Proto → Emergence
2\textsuperscript{nd}: Self-Healing Gradients

14 lines of Proto

Definition \quad \text{Continuous} \quad \text{Proto} \quad \text{Emergence}
3\textsuperscript{nd}: Dispersion

3 lines of Proto
Together: Guided Flocking

8 lines of Proto = 38 total
Four Useful Principles

- Self-Scaling
- Sparseness
- Gradual degradation
- Failure simplification
Gradual Degradation

• Decoupling by low sensitivity to
  – Implementation details
  – Parameter values
  – Conditions of execution

• Use when you don't understand or can't control the environment.
Gradual Degradation: Implementation Details

- Plane wave at different resolutions:

- Definition
- Continuous
- Proto
- Emergence
Self-Scaling

- Decoupling through geometry:
  - specification of behavior (units)
  - implementation details (coordinate system)
- Use when you don't know the relationship between the behavior you want and the details of its implementation
Failure Simplification

- Decoupling by preferential selection of preferred failure type
- Use when you don't understand or can't prevent failures

We're used to preventing failures. What if we just manage their impact?
Self-Scaling, Failure Simplification: Neighborhood Ops

Definition → Continuous → Proto → Emergence
Sparseness

• Decoupling by making unwanted interactions rare.
• Use when devices need to make non-interfering decisions independently.

*If at first you don't succeed, just try again.*
Sparseness: Symmetry Breaking

• Temporary leadership via 1/f noise
Contributions

• Many networks are spatial computers
• Continuous-space geometric programs are simple, scalable, and robust
• Proto compiles global descriptions into approximate local implementations
• Emergent behaviors can be engineered with by modulating regions
• Four useful principles for engineering emergence