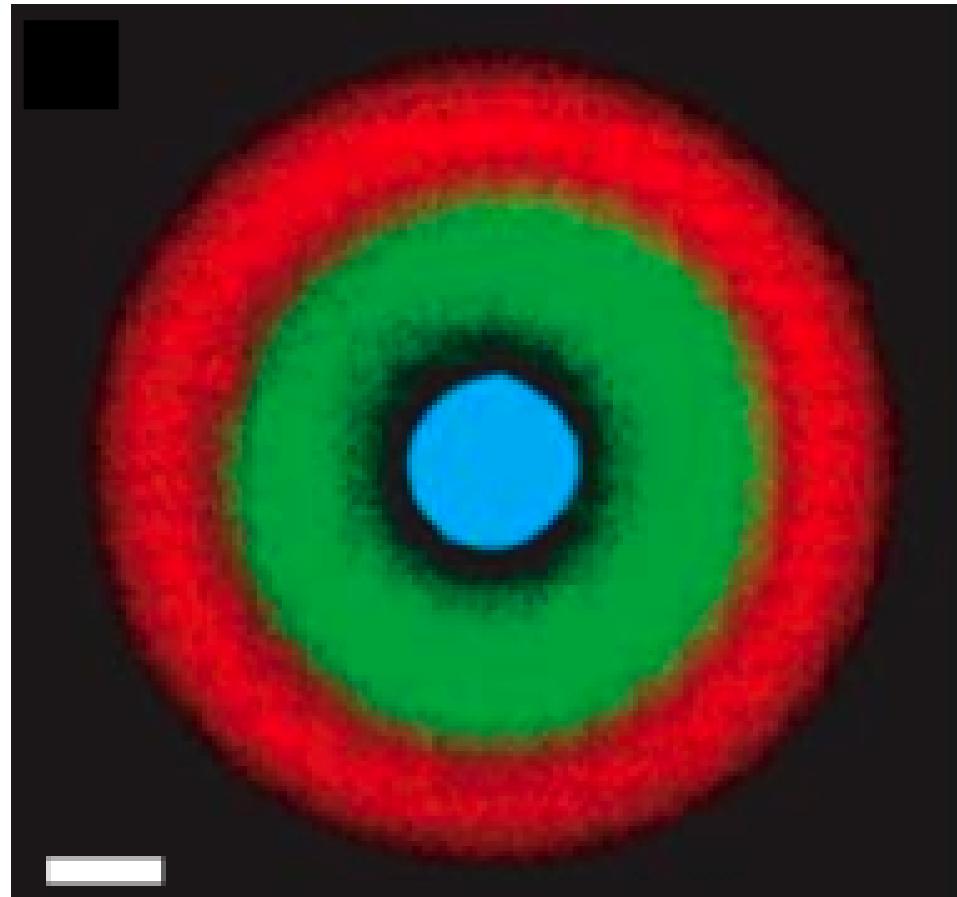


# Spatial Computing

*From Manifold Geometry to Biology*

Jacob Beal  
AMORPH Conference  
August, 2010

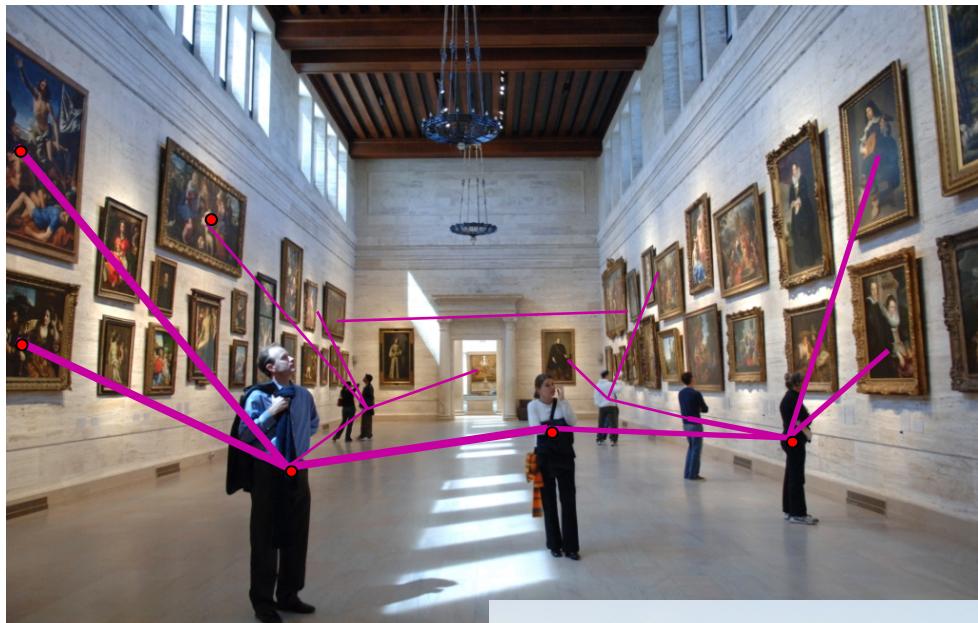
# How do you program $10^{12}$ cells?



[Weiss, '05]

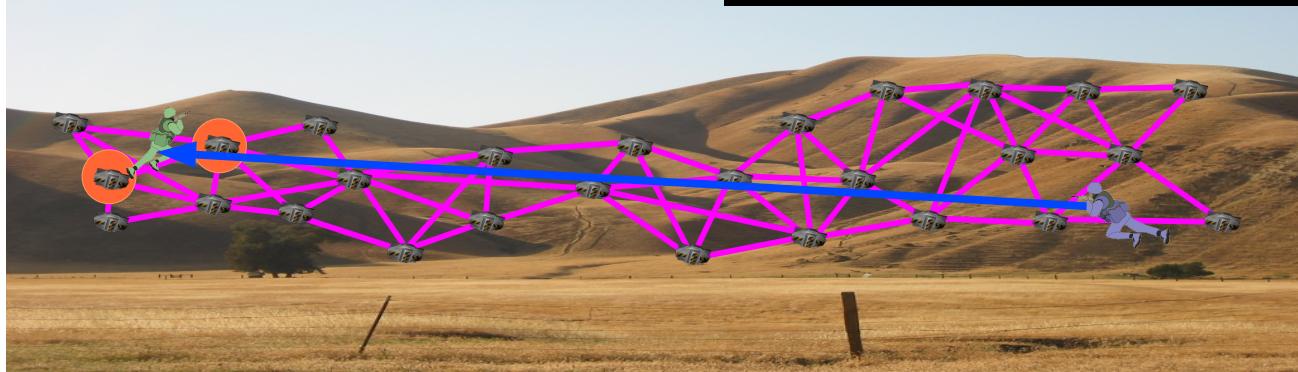
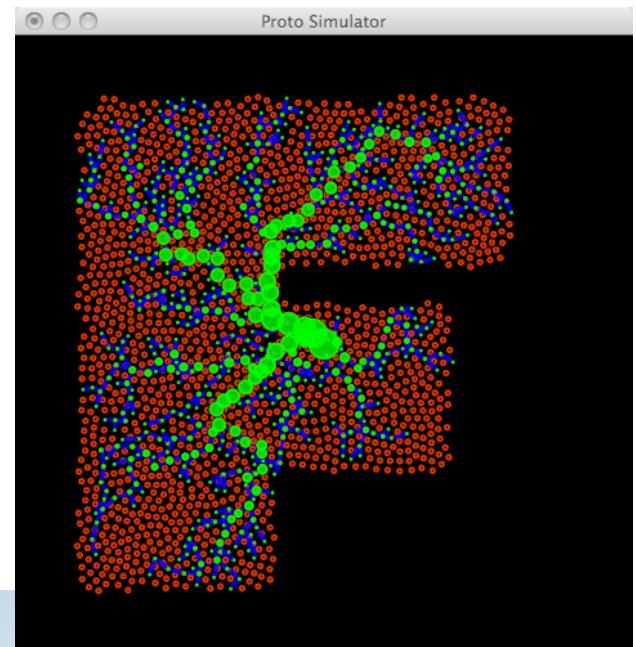
*Forget the cells... program the space!*

# When the world is geometric... take advantage of it!



Pervasive Computing

Morphogenetic Engineering



Sensor Networks

# Outline

- **What is Spatial Computing?**
- Global → Local → Global
- From Space to Robustness & Scalability
- The Biological World

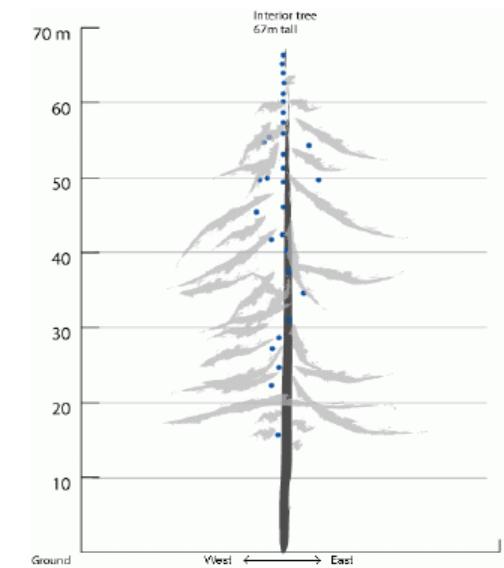
# Spatial Computers



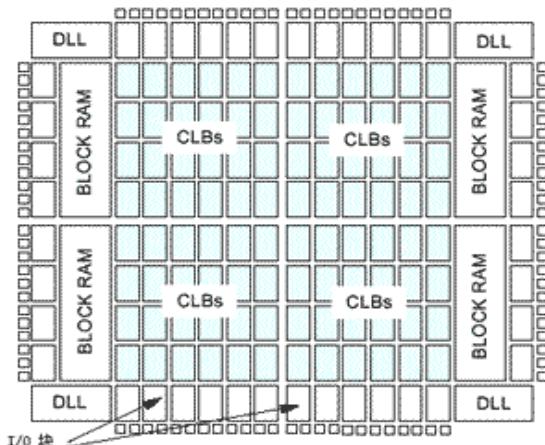
Robot Swarms



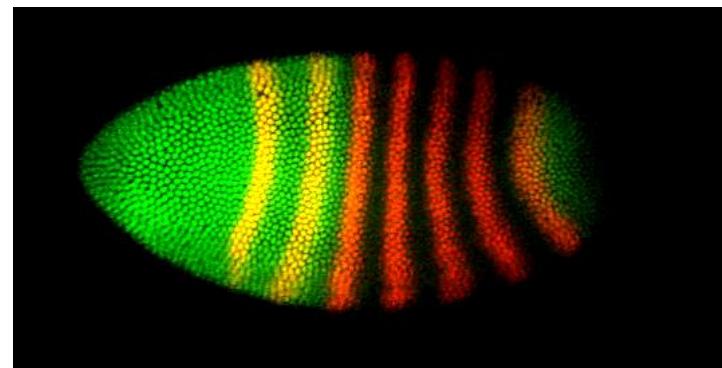
Biological Computing



Sensor Networks



Reconfigurable Computing



Cells during Morphogenesis



Modular Robotics

# More formally...

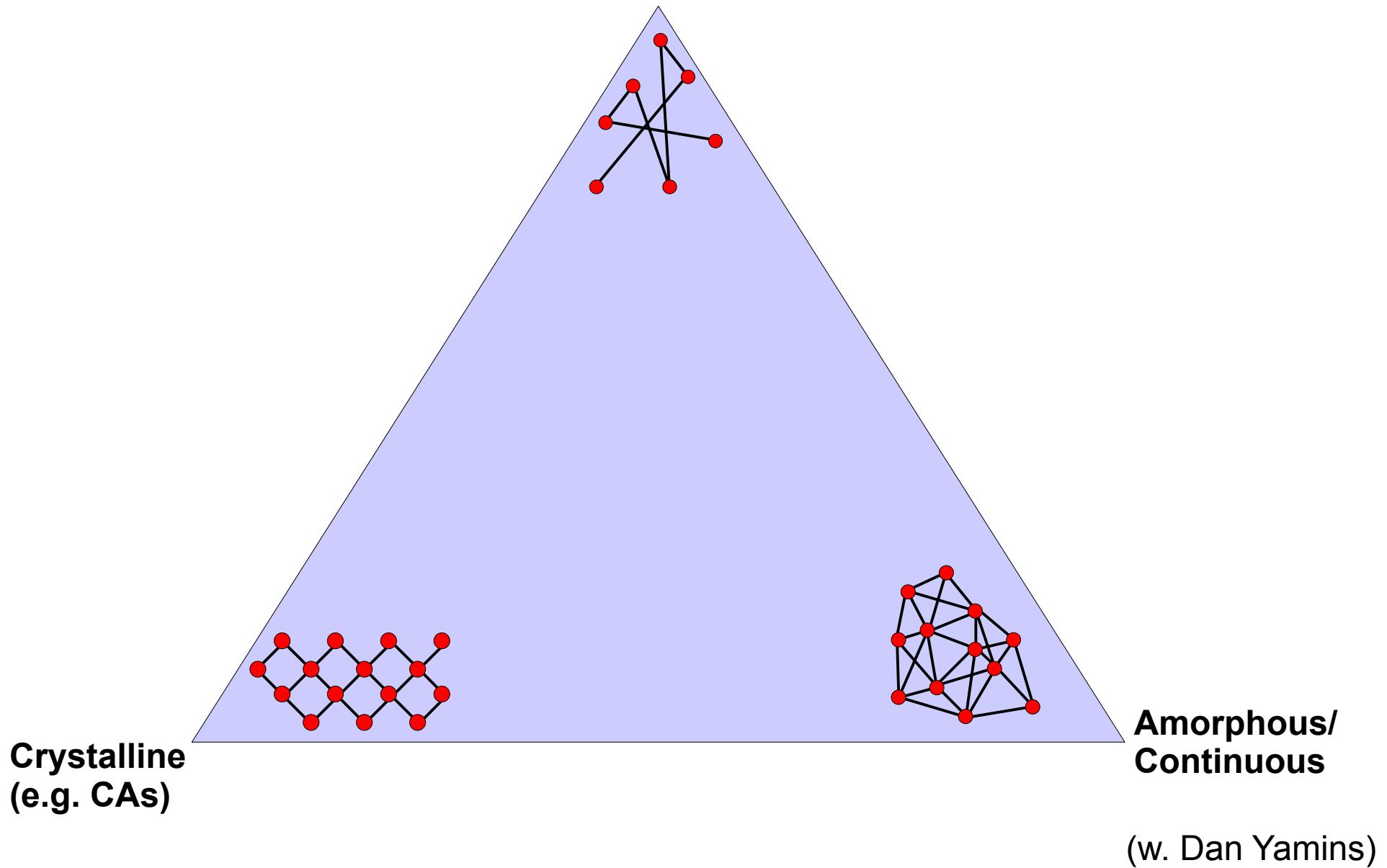
- A spatial computer is a collection of computational devices distributed through a physical space in which:
  - the difficulty of moving information between any two devices is strongly dependent on the distance between them, and
  - the “functional goals” of the system are generally defined in terms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

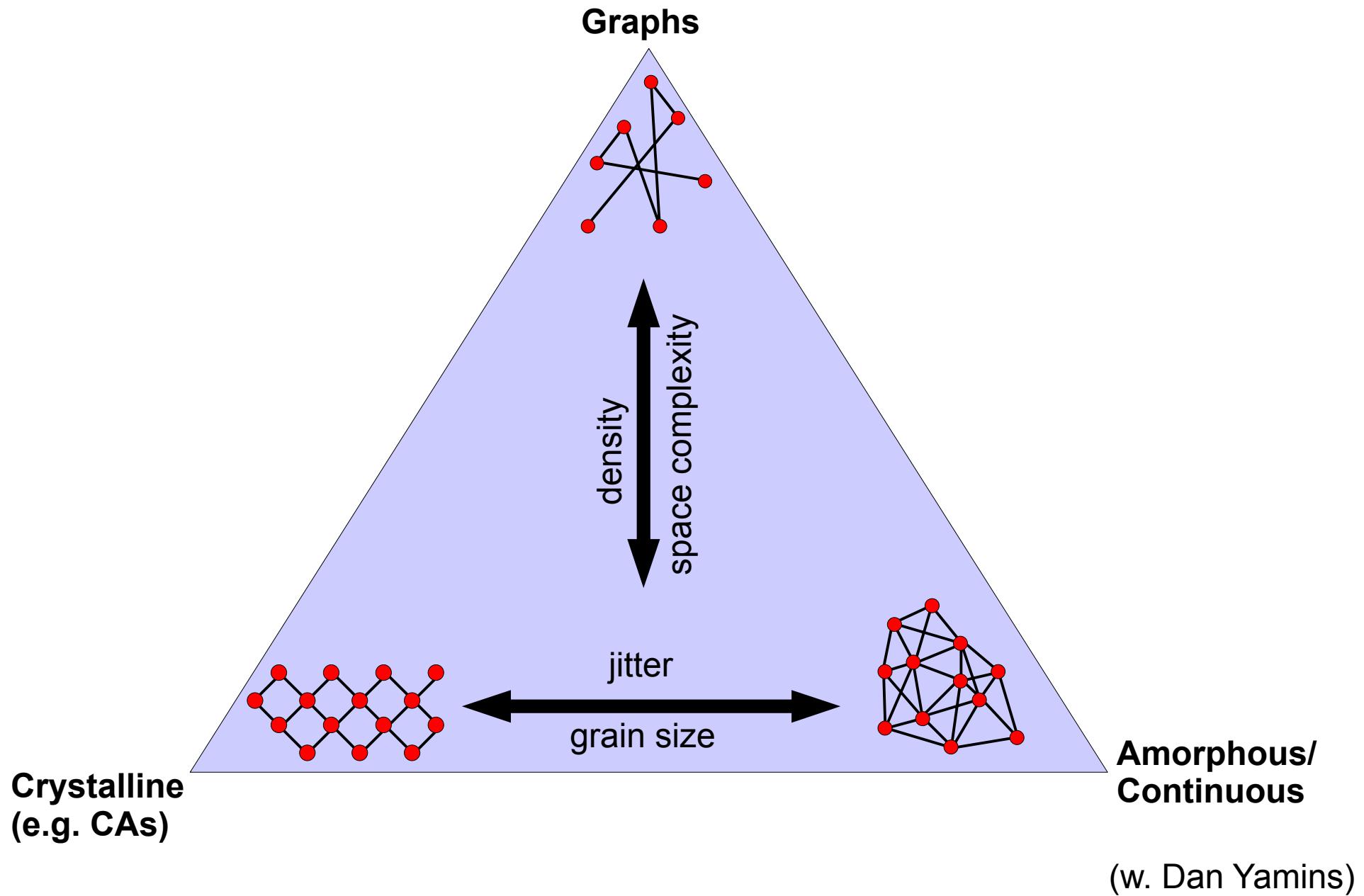
# More formally...

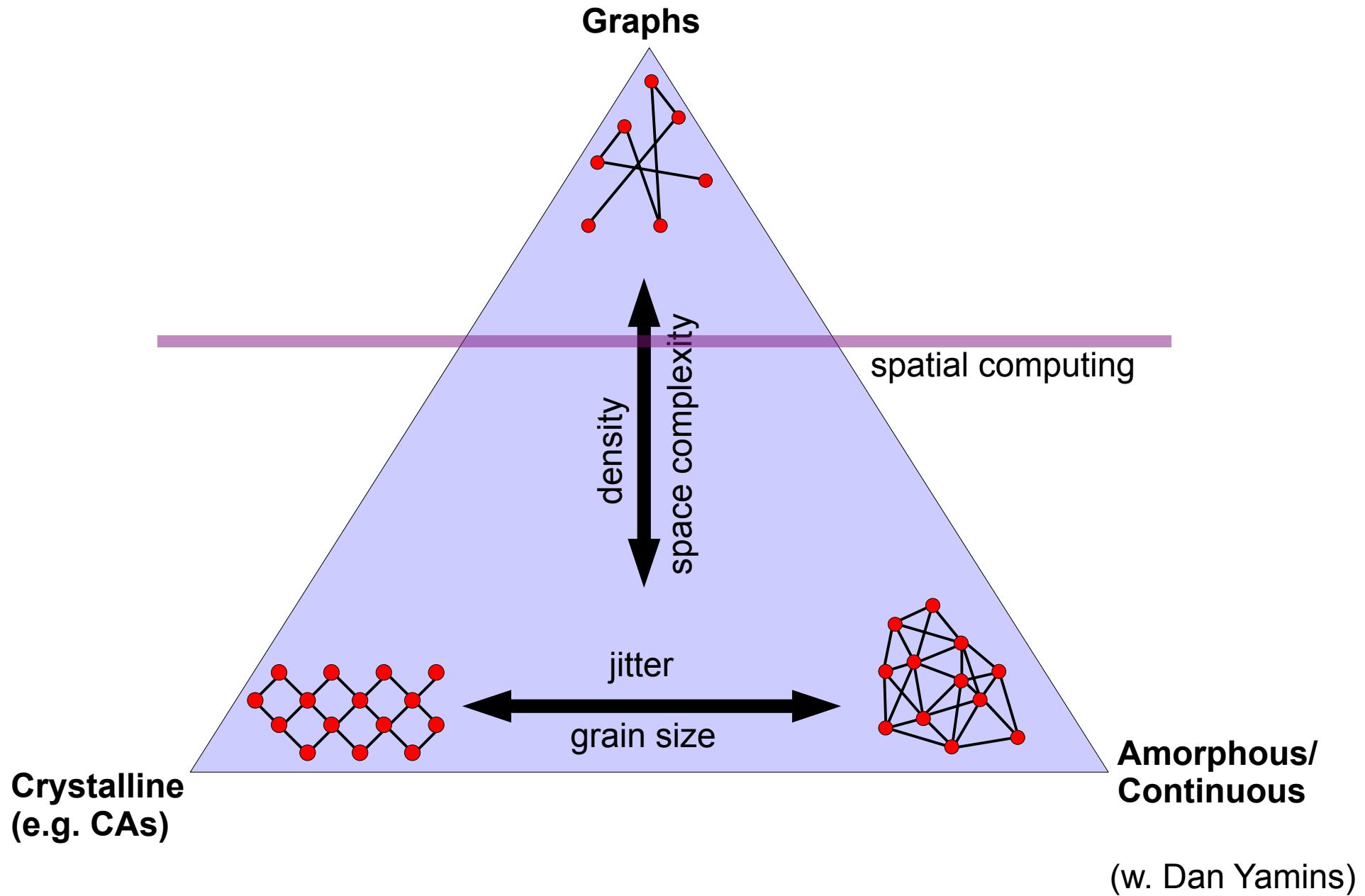
- A spatial computer is a collection of computational devices **distributed through** a physical space in which:
  - the difficulty of moving information between any two devices is **strongly dependent on the distance** between them, and
  - the “functional goals” of the system are **generally defined**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

*Notice the ambiguities in the definition*

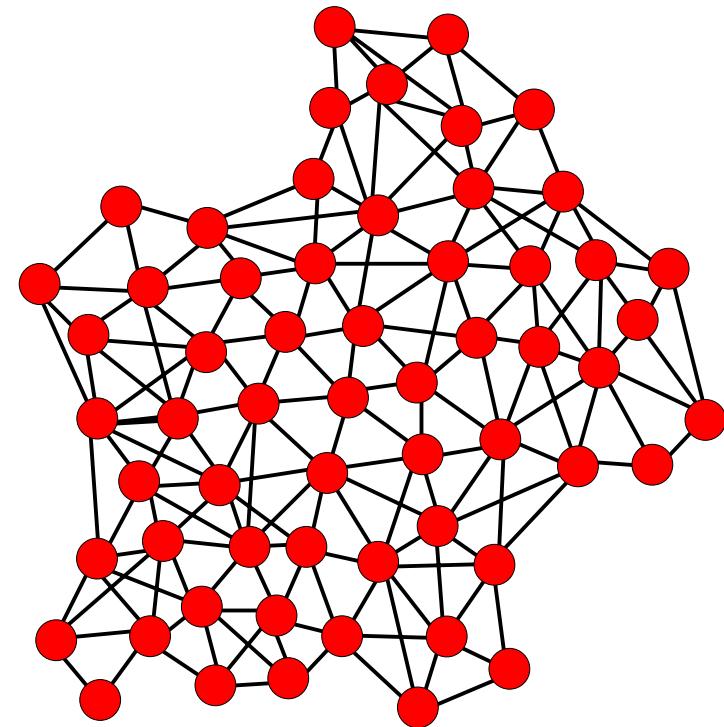
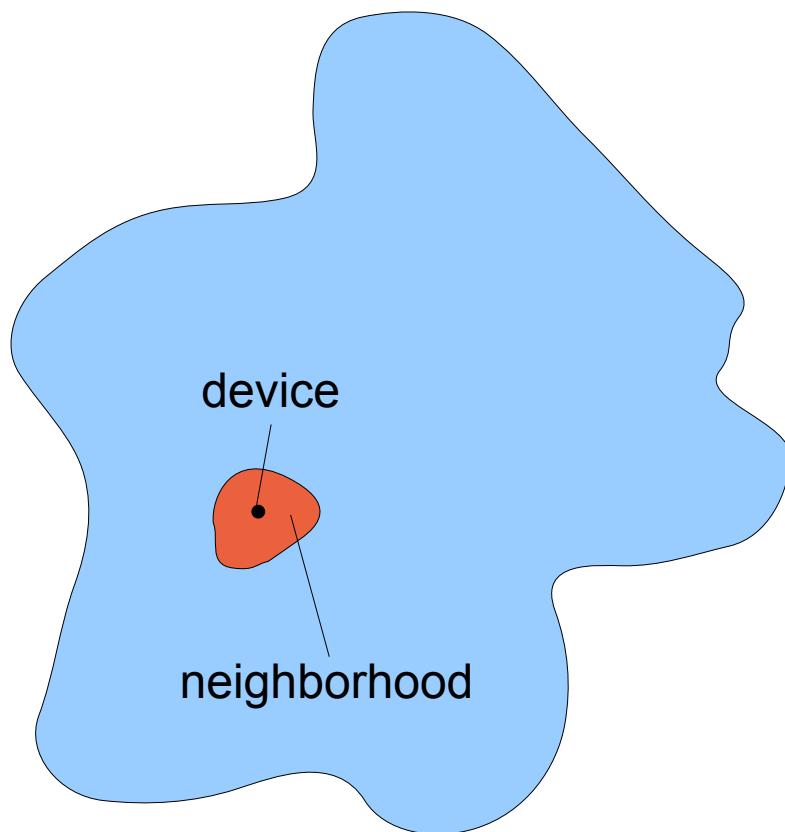
## Graphs







# Space/Network Duality



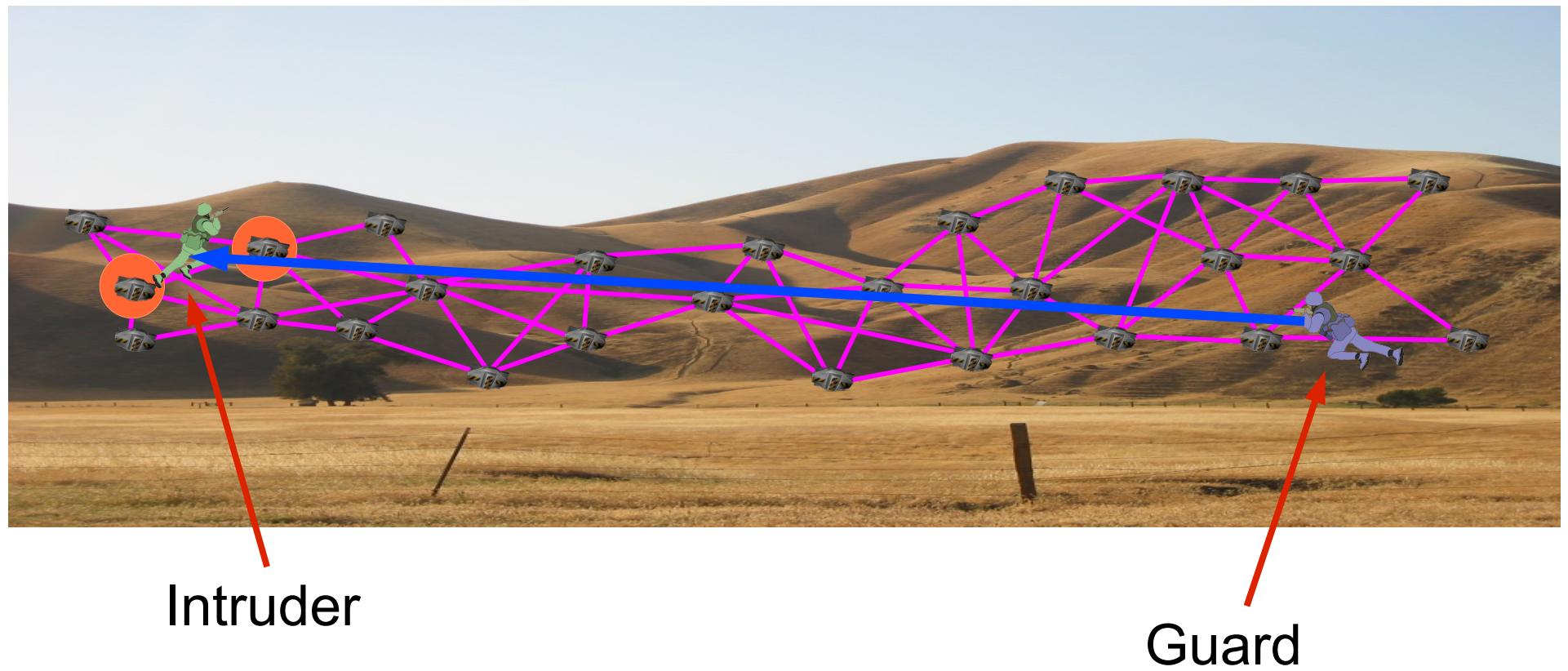
# Example: Target Tracking



# Example: Target Tracking



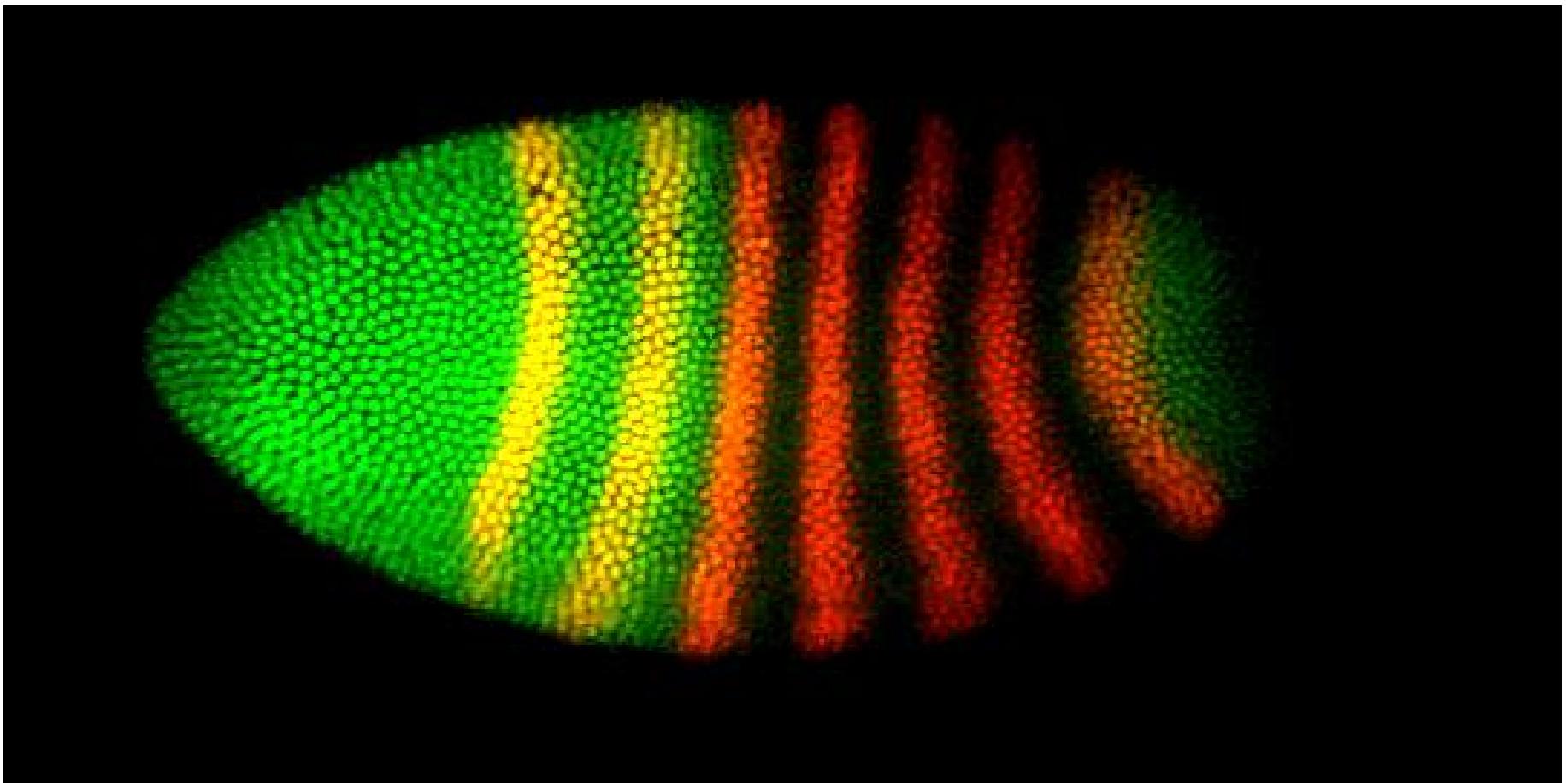
# Example: Target Tracking



# Example: Museum Guide



# Example: Morphogenesis



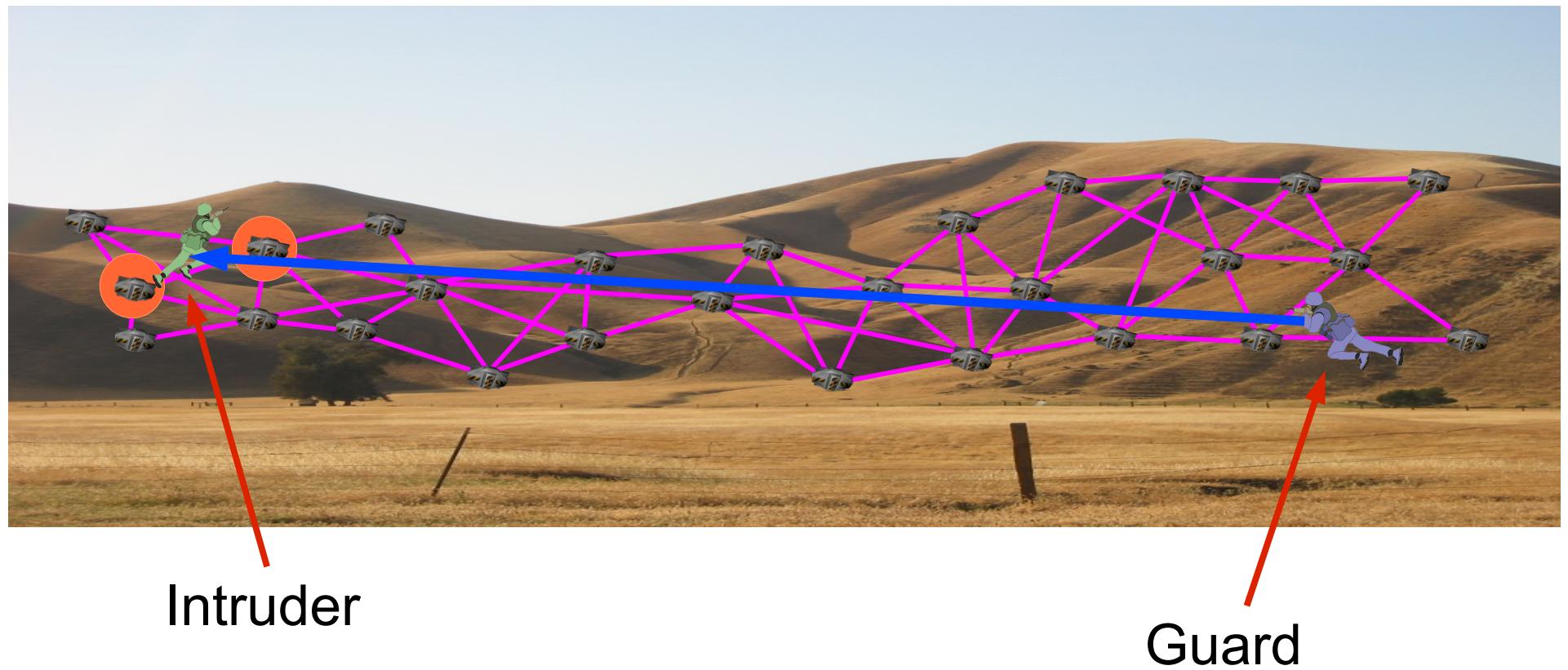
# How can we program these?

- Desiderata for approaches:
  - Simple, easy to understand code
  - Robust to errors, adapt to changing environment
  - Scalable to potentially vast numbers of devices
  - Take advantage of spatial nature of problems

# Outline

- What is Spatial Computing?
- **Global → Local → Global**
- From Space to Robustness & Scalability
- The Biological World

# Example: Target Tracking

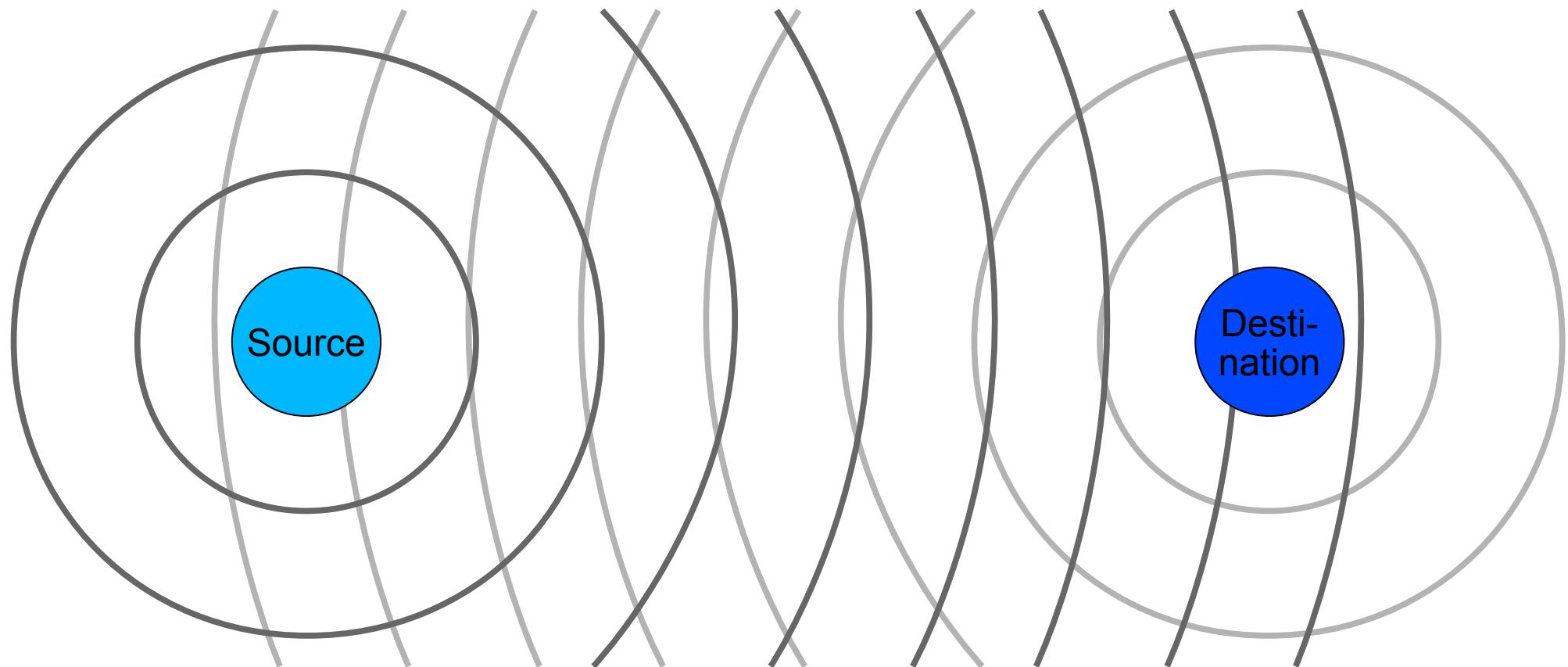


# Geometric Program: Channel



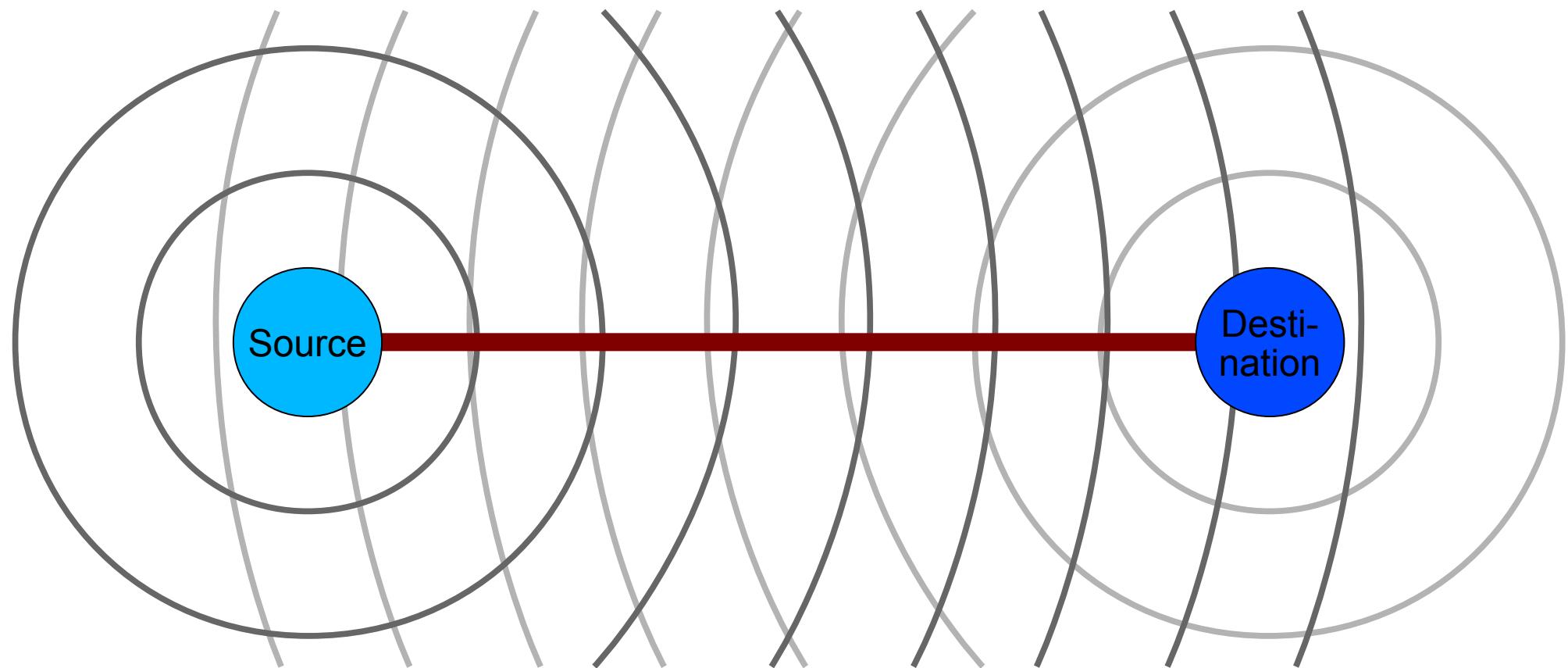
(cf. Butera)

# Geometric Program: Channel



(cf. Butera)

# Geometric Program: Channel



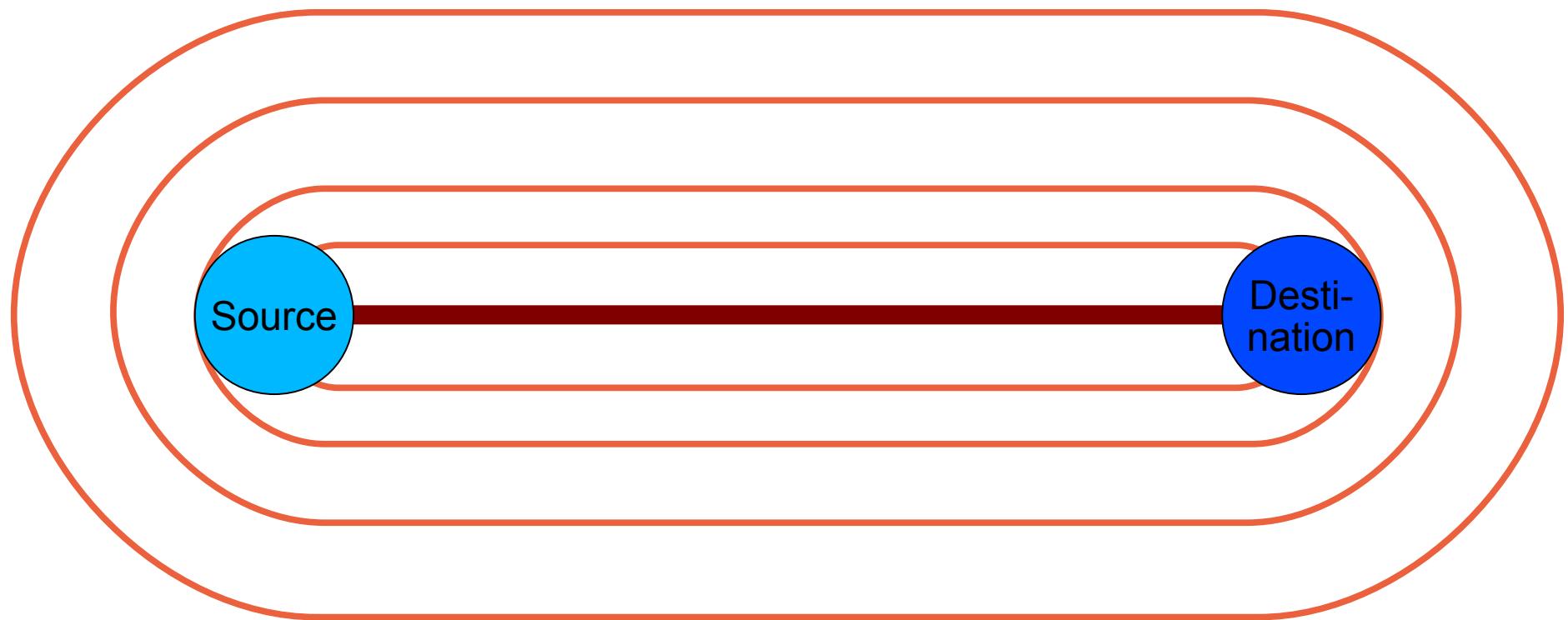
(cf. Butera)

# Geometric Program: Channel



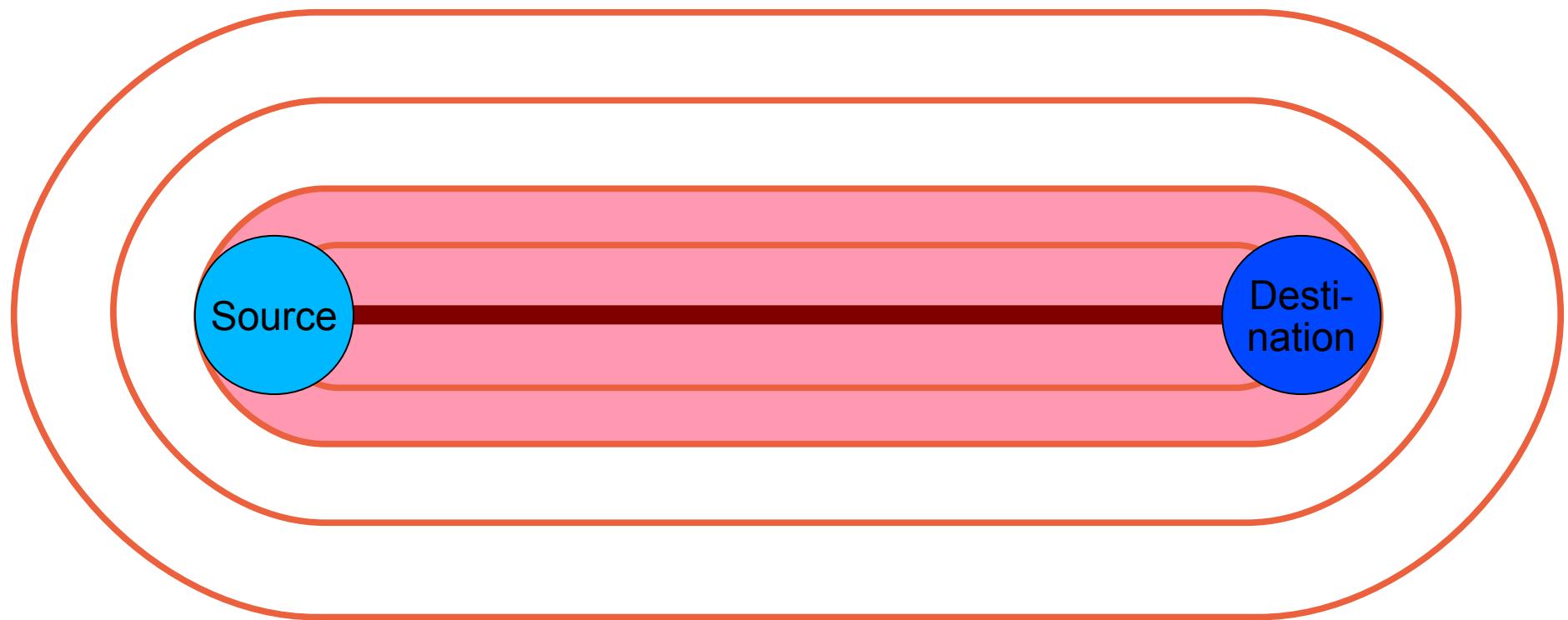
(cf. Butera)

# Geometric Program: Channel



(cf. Butera)

# Geometric Program: Channel



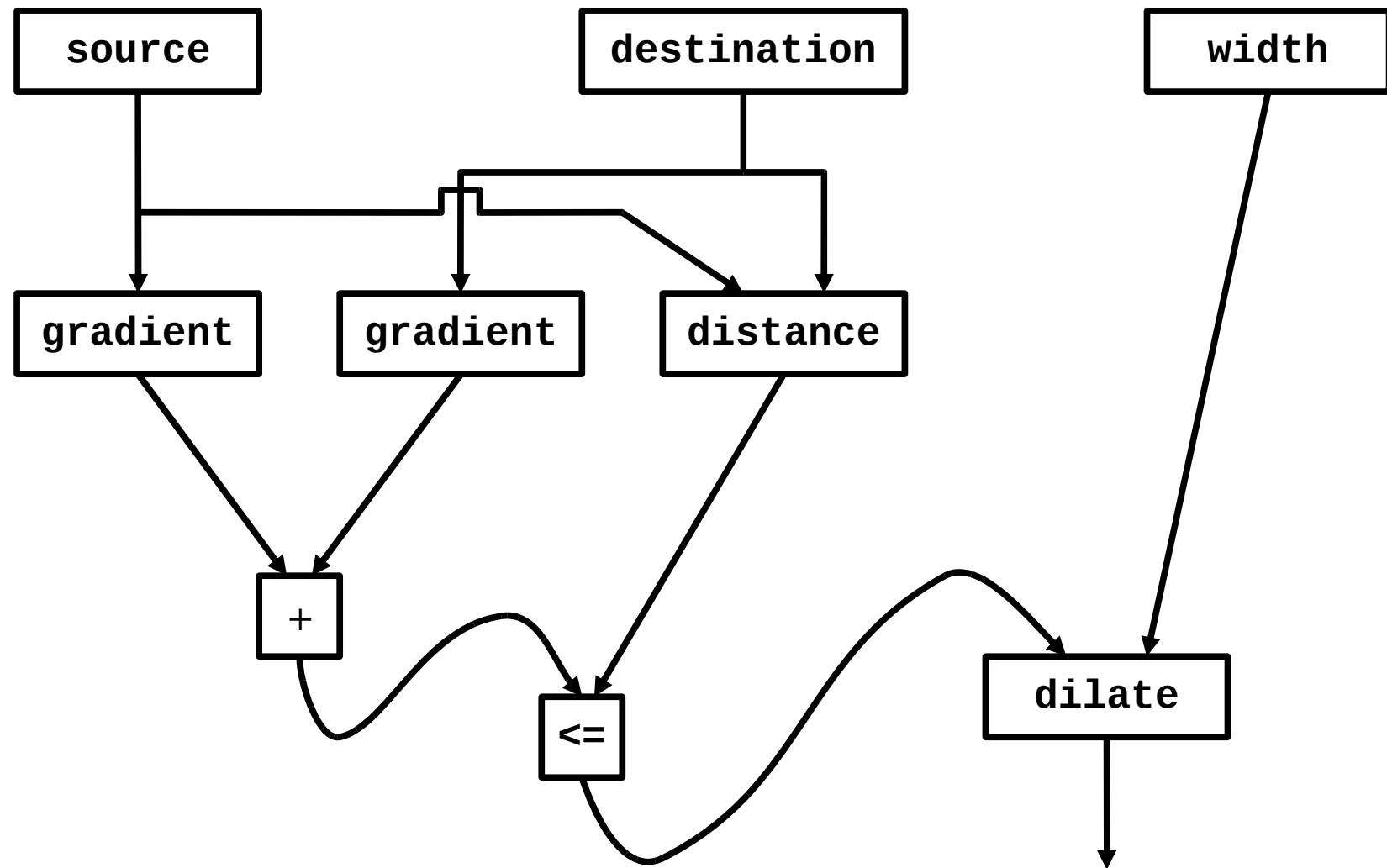
(cf. Butera)

# Geometric Program: Channel

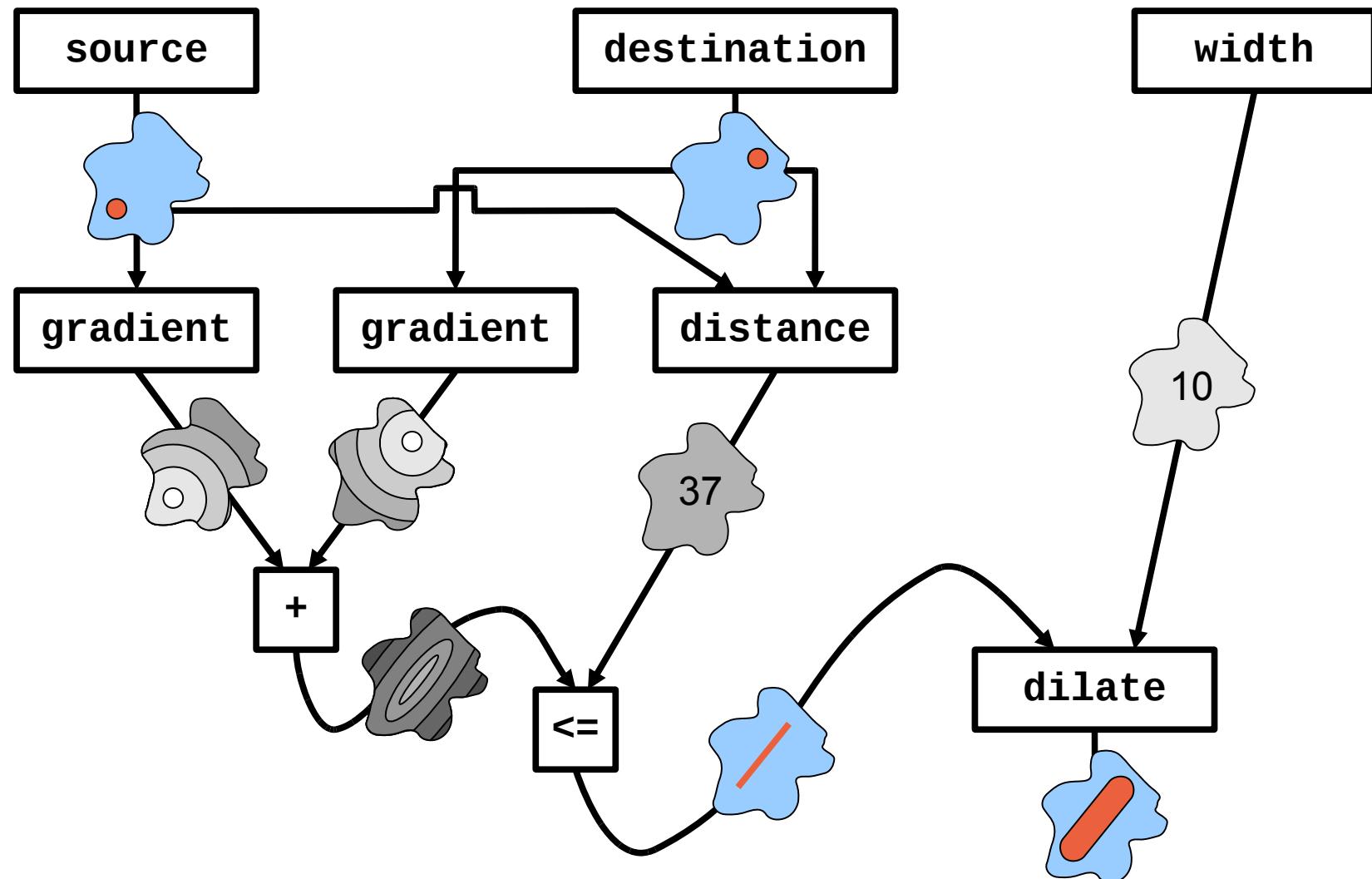


(cf. Butera)

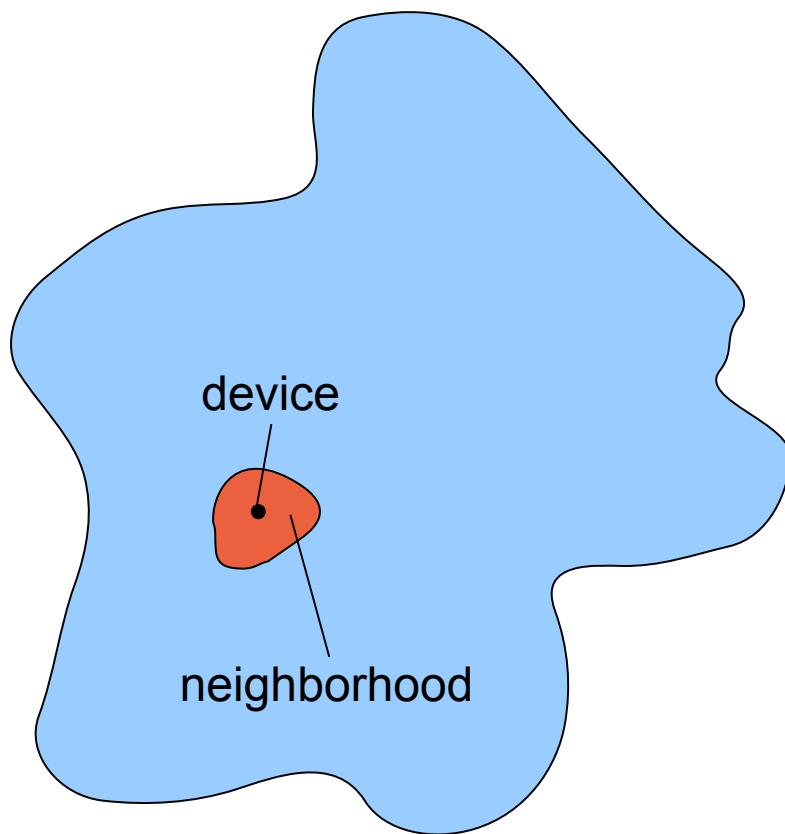
# Computing with fields



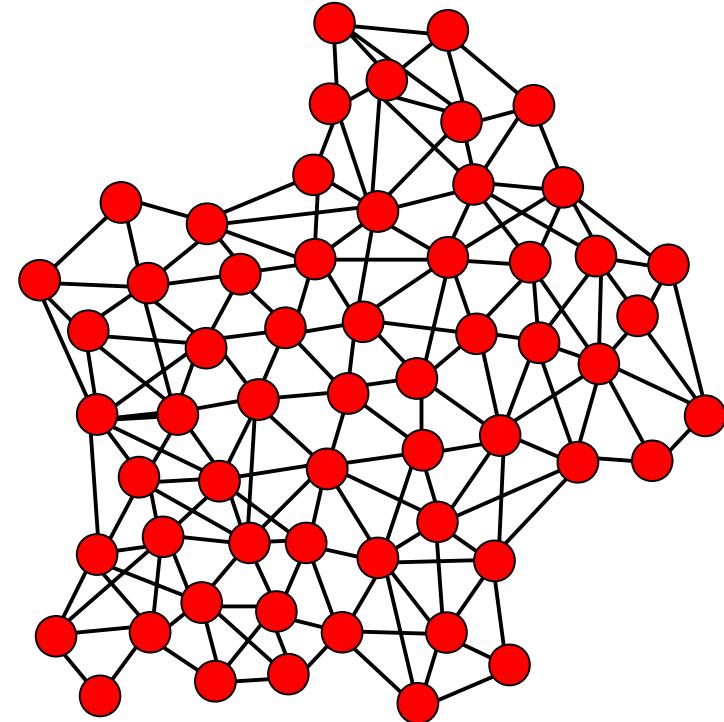
# Computing with fields



# Amorphous Medium



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



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

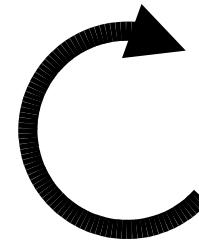
# Proto

```
(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)))
```

**platform  
specificity &  
optimization**

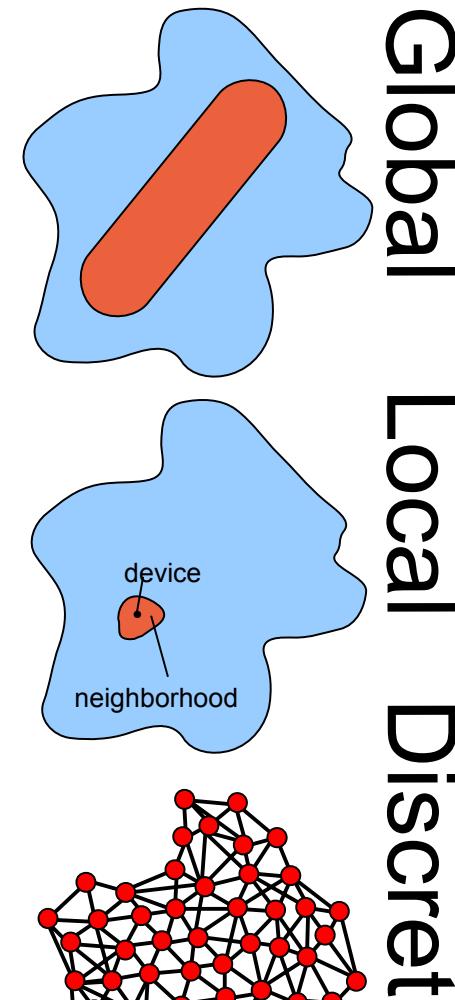
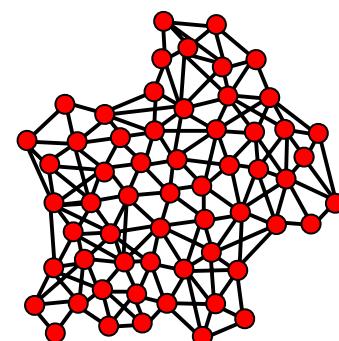
**evaluation** →

**global to local  
compilation**

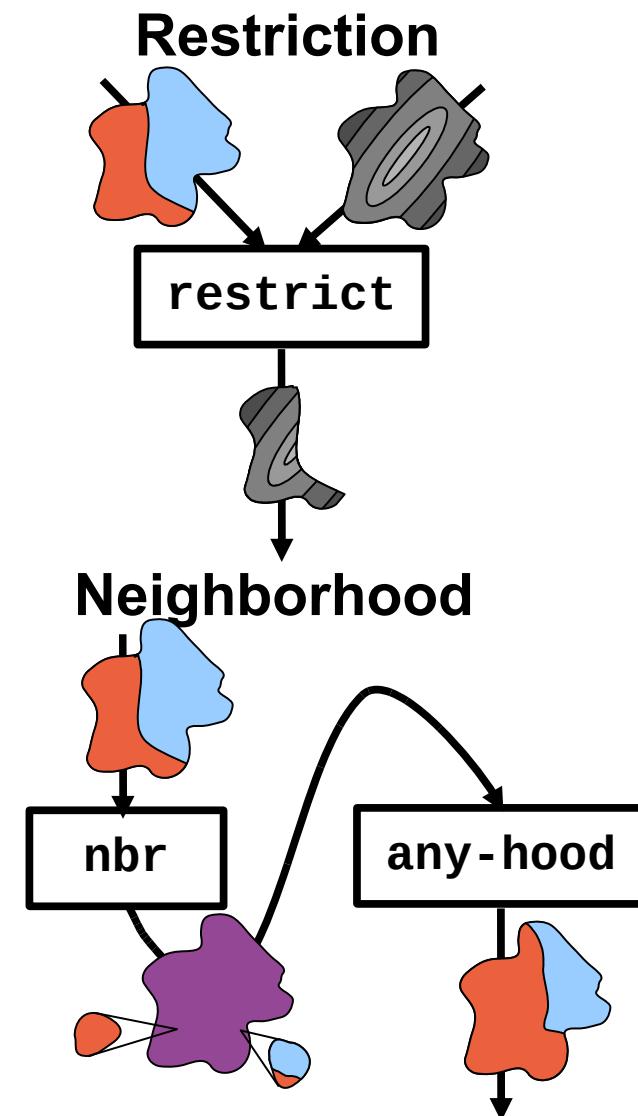
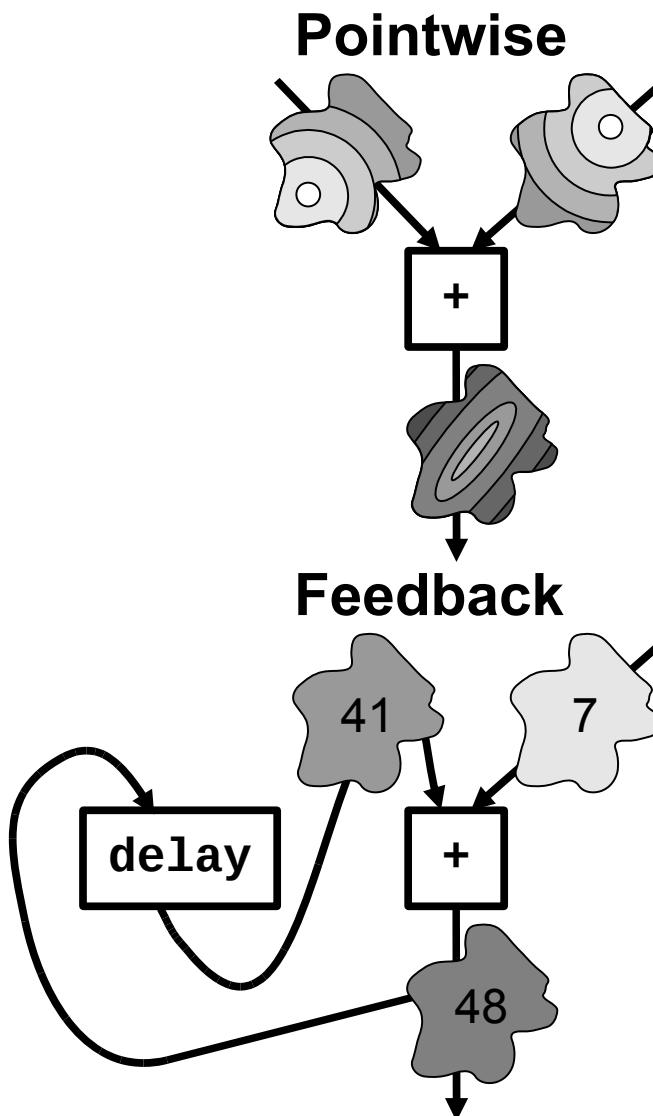


**discrete  
approximation**

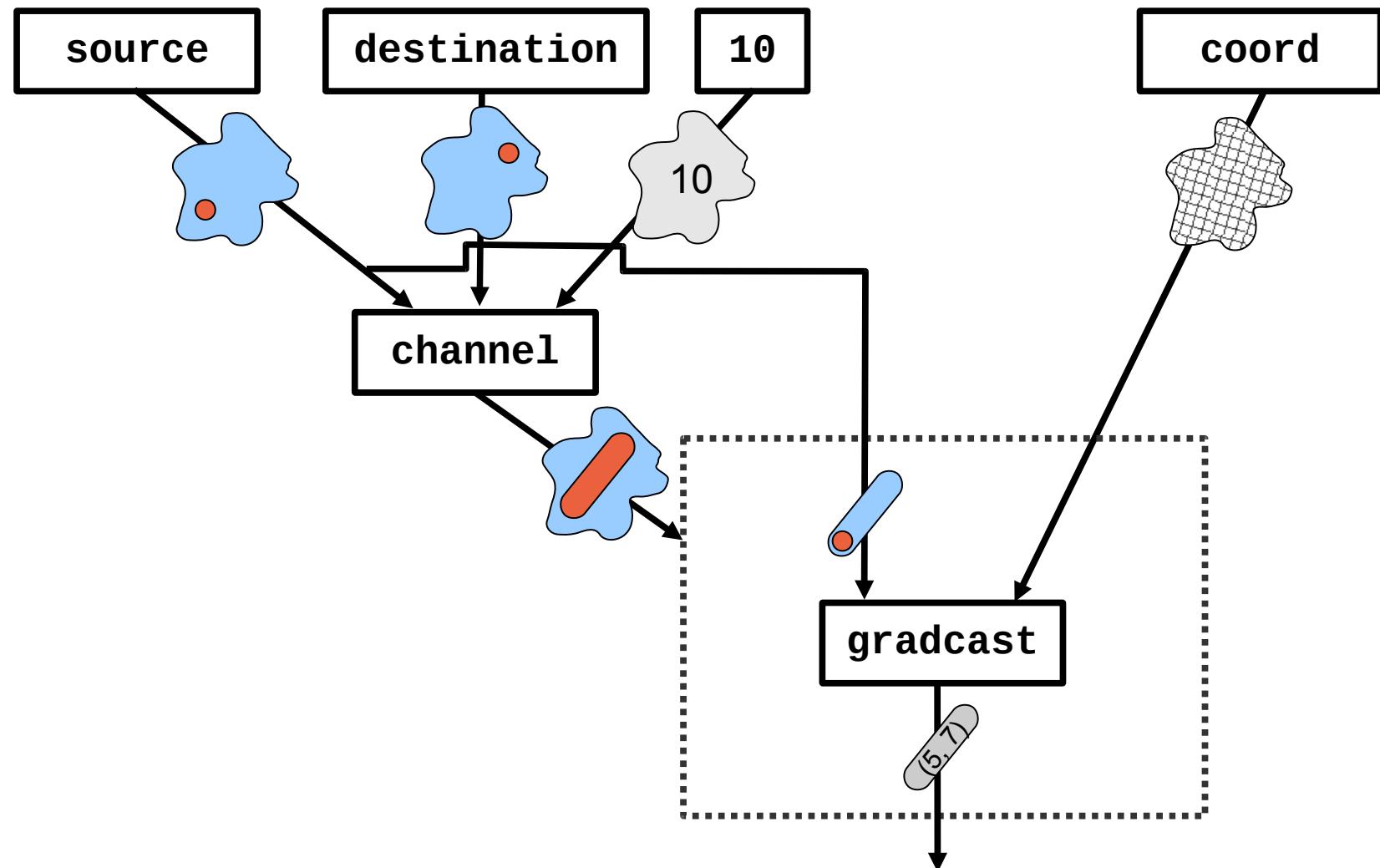
Device  
Kernel



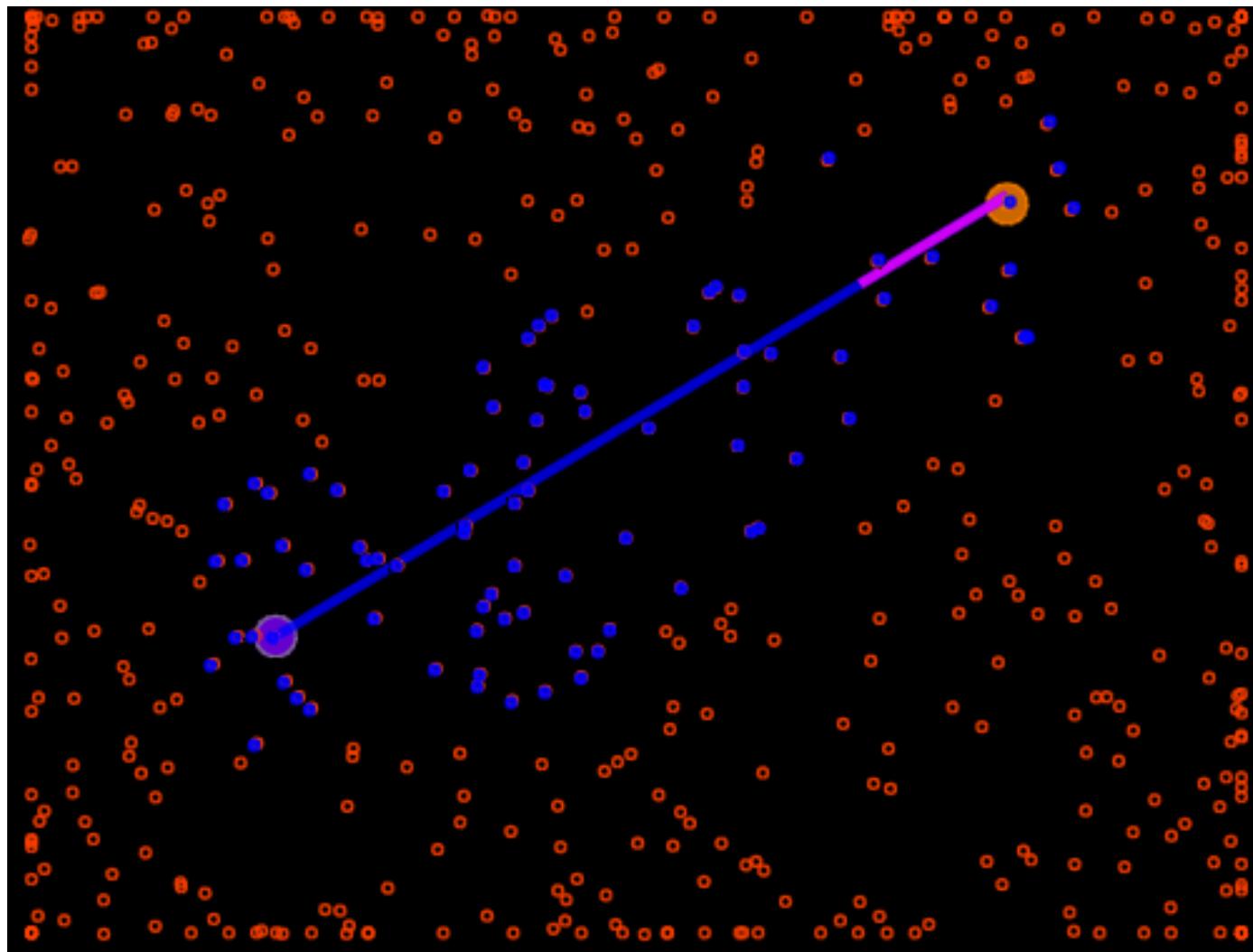
# Proto's Families of Primitives



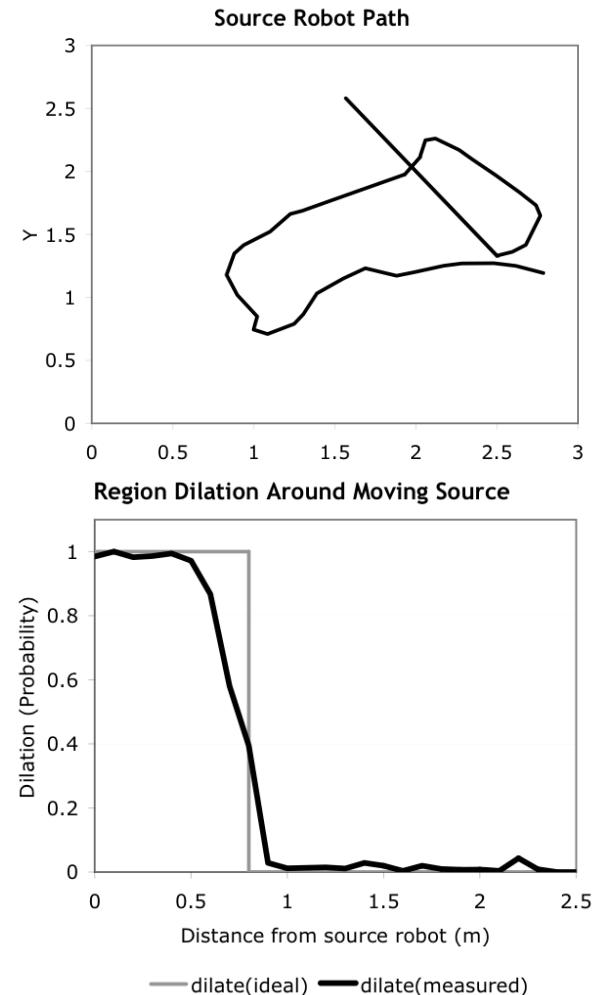
# Modulation by Restriction



# In simulation...

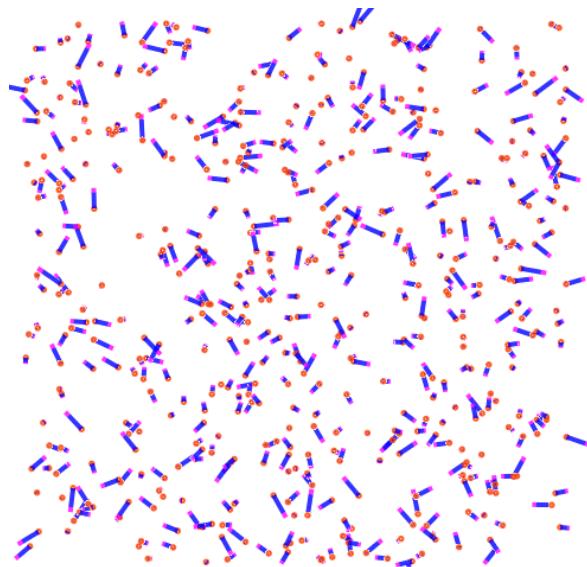


# Swarm Robots

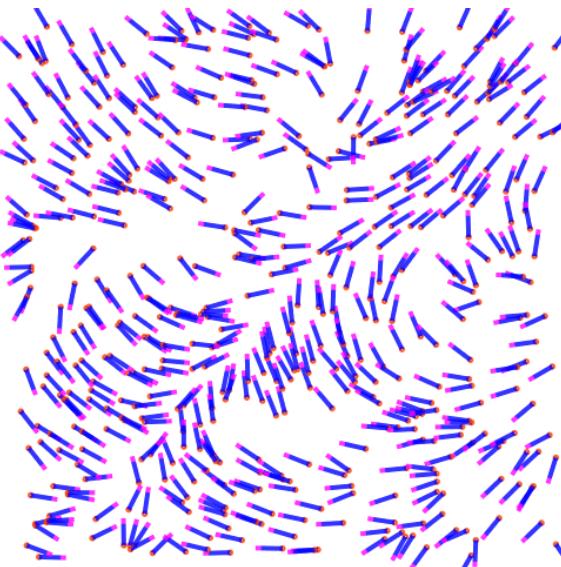


w. McLurkin, Bachrach, Correll

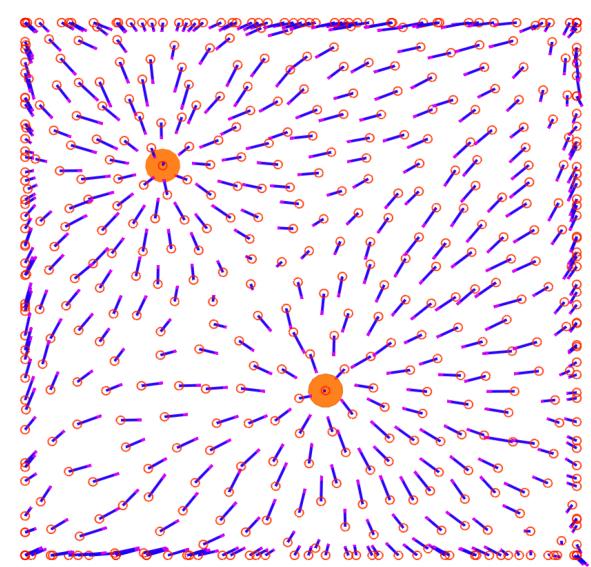
# Device Motion = Vector Fields



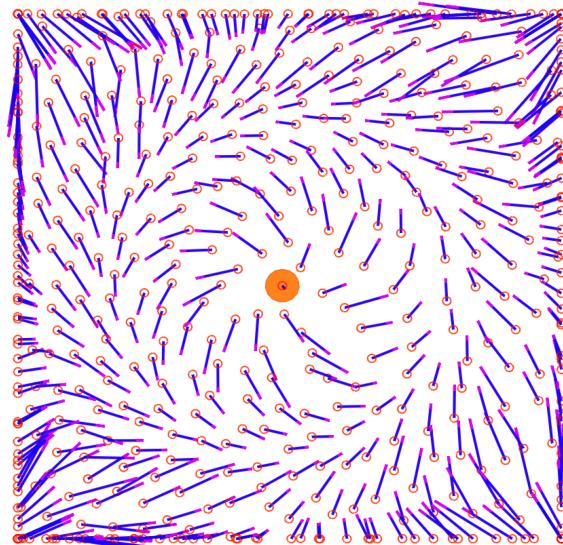
brownian



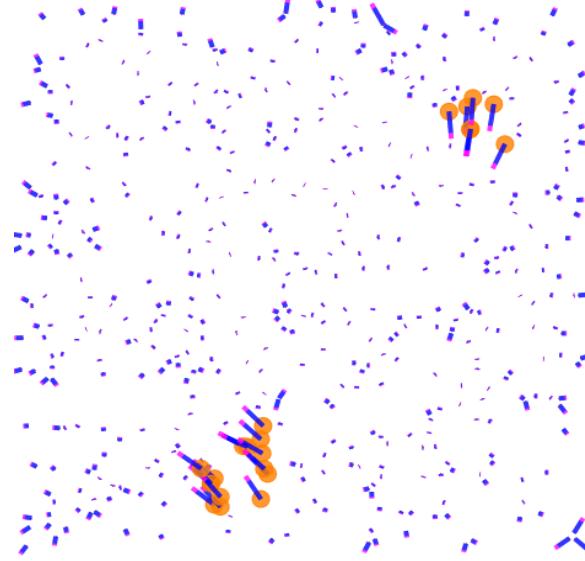
flock



cluster-to



contour-field



search-and-rescue

# Weaknesses

- Functional programming scares people
- Programmers can break the abstraction
- No dynamic allocation of processes
- No formal proofs available for quality of approximation in a composed program

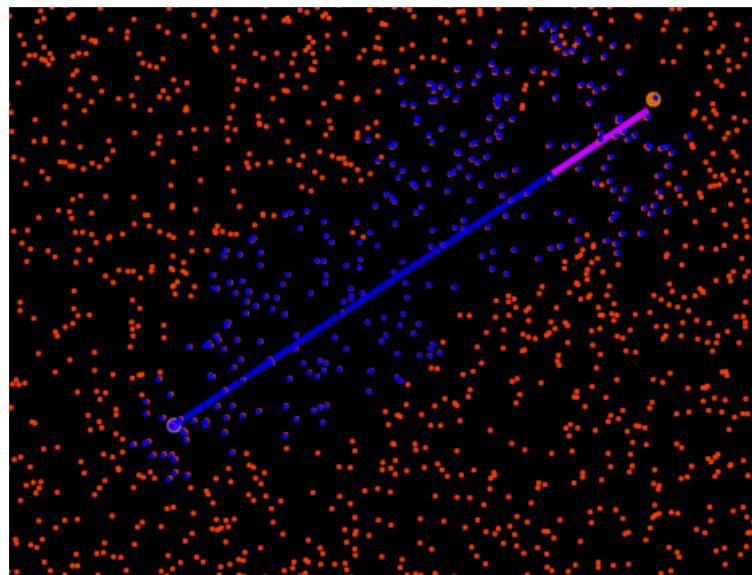
*(active research on last two)*

# Outline

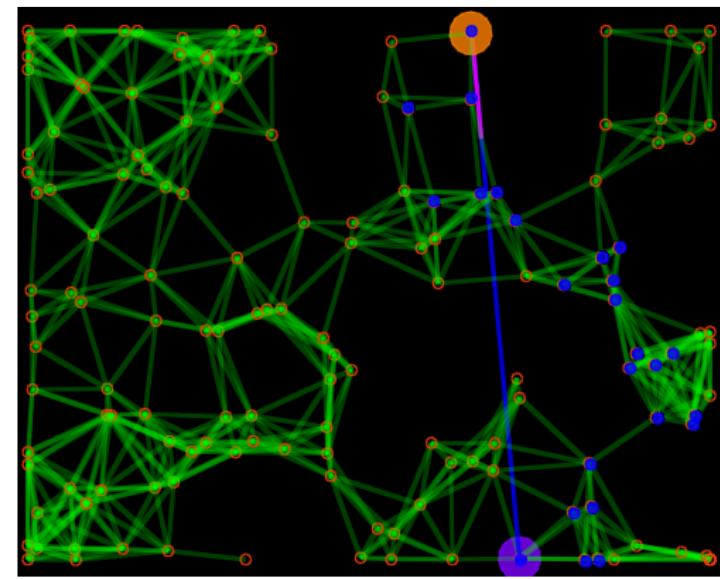
- What is Spatial Computing?
- Global → Local → Global
- **From Space to Robustness & Scalability**
- The Biological World

# Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness

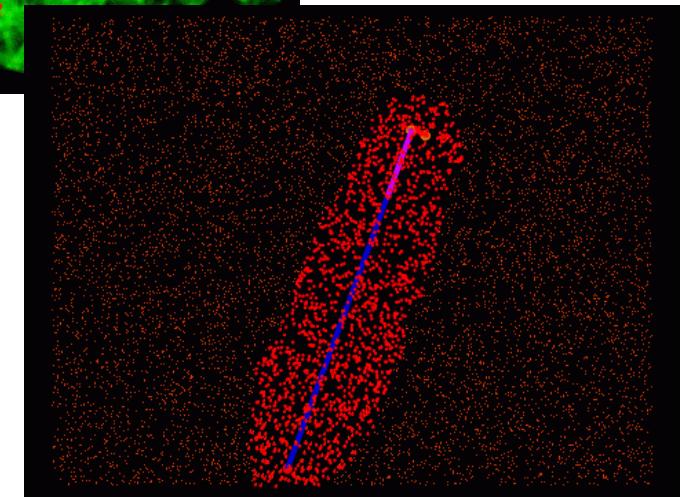
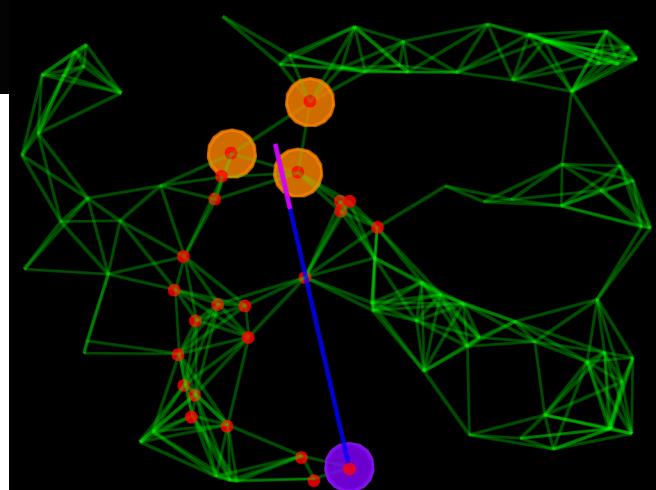
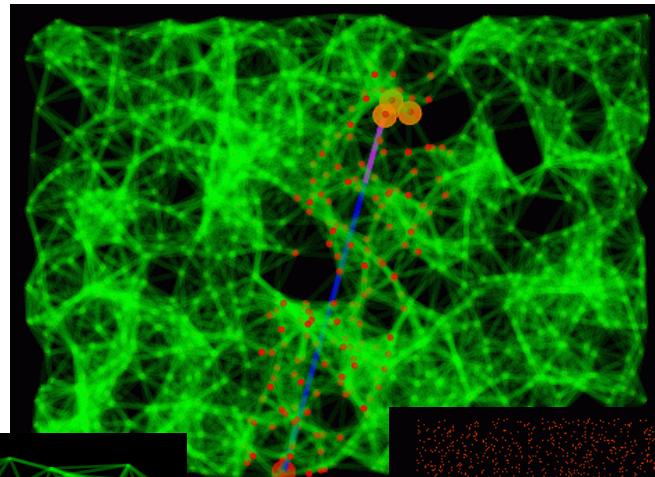
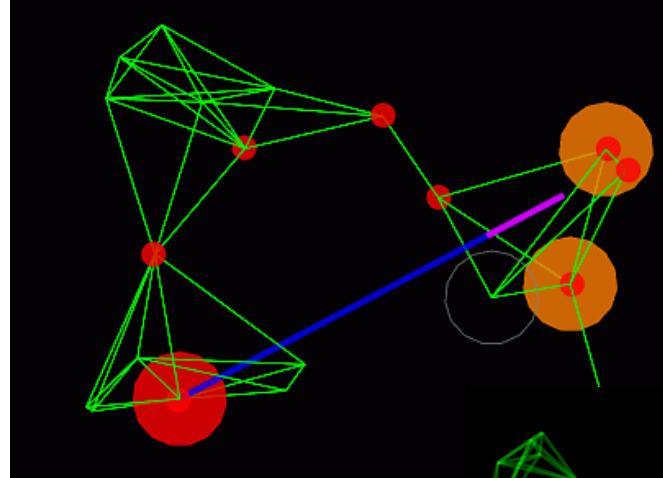


2000 devices



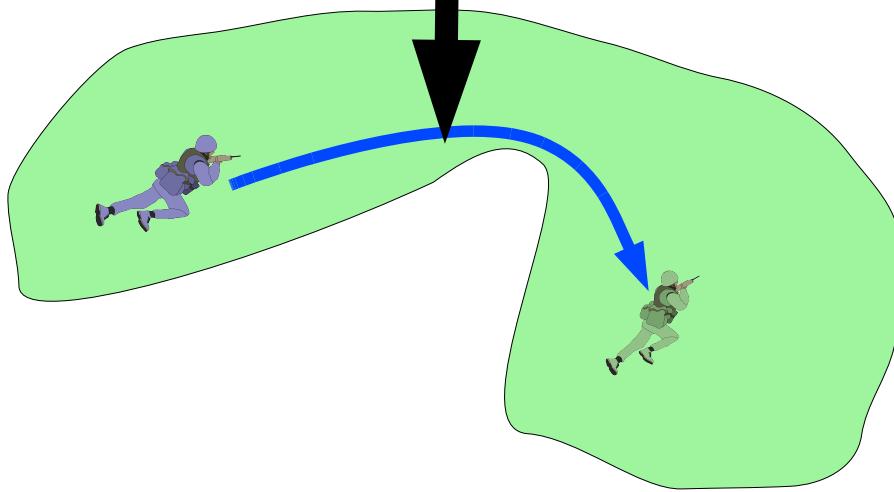
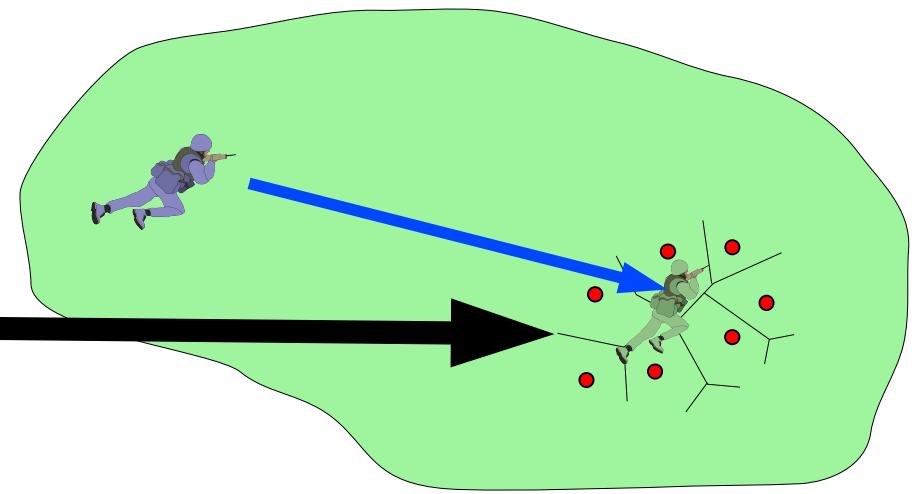
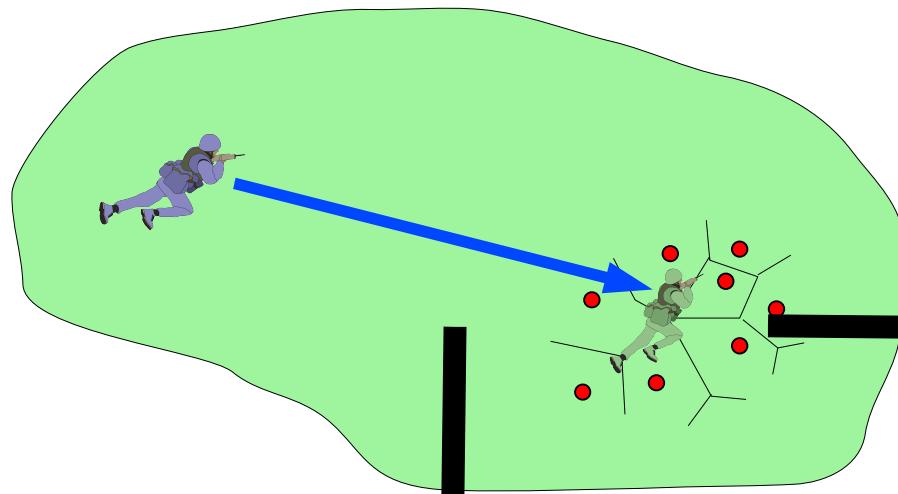
150 devices

# Continuous Programs → Self-Scaling



*Target tracking across three orders of magnitude*

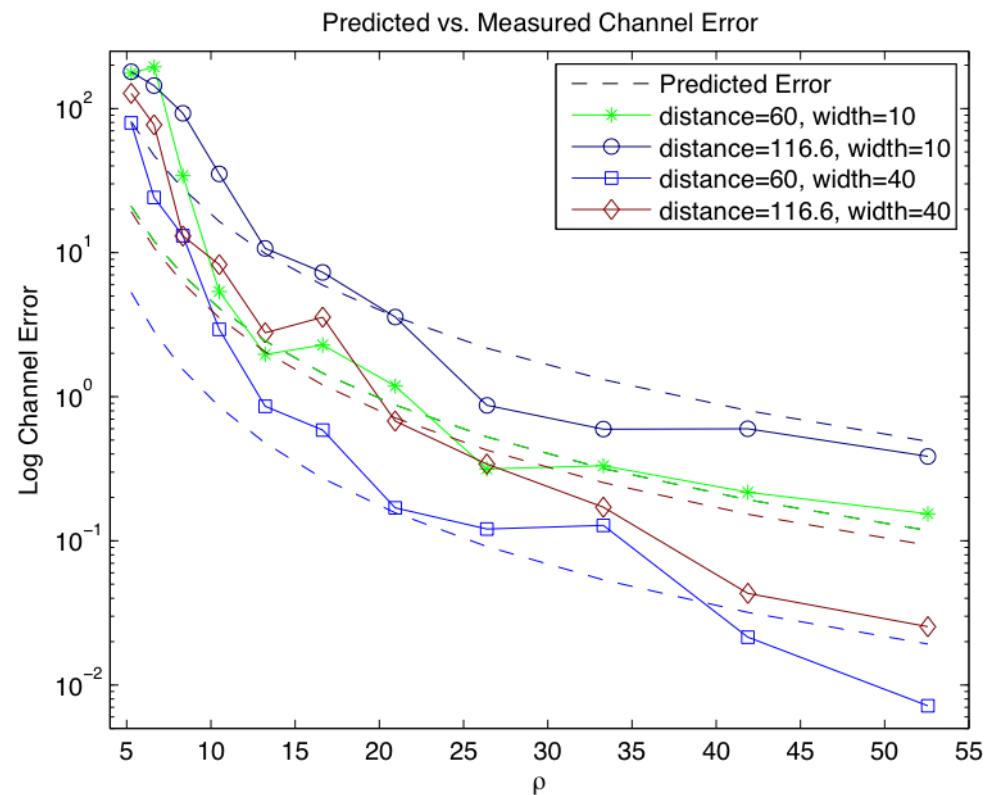
# Robustness



- Local change adapts in discrete approximation
- Global change adapts in manifold geometry

# Composition

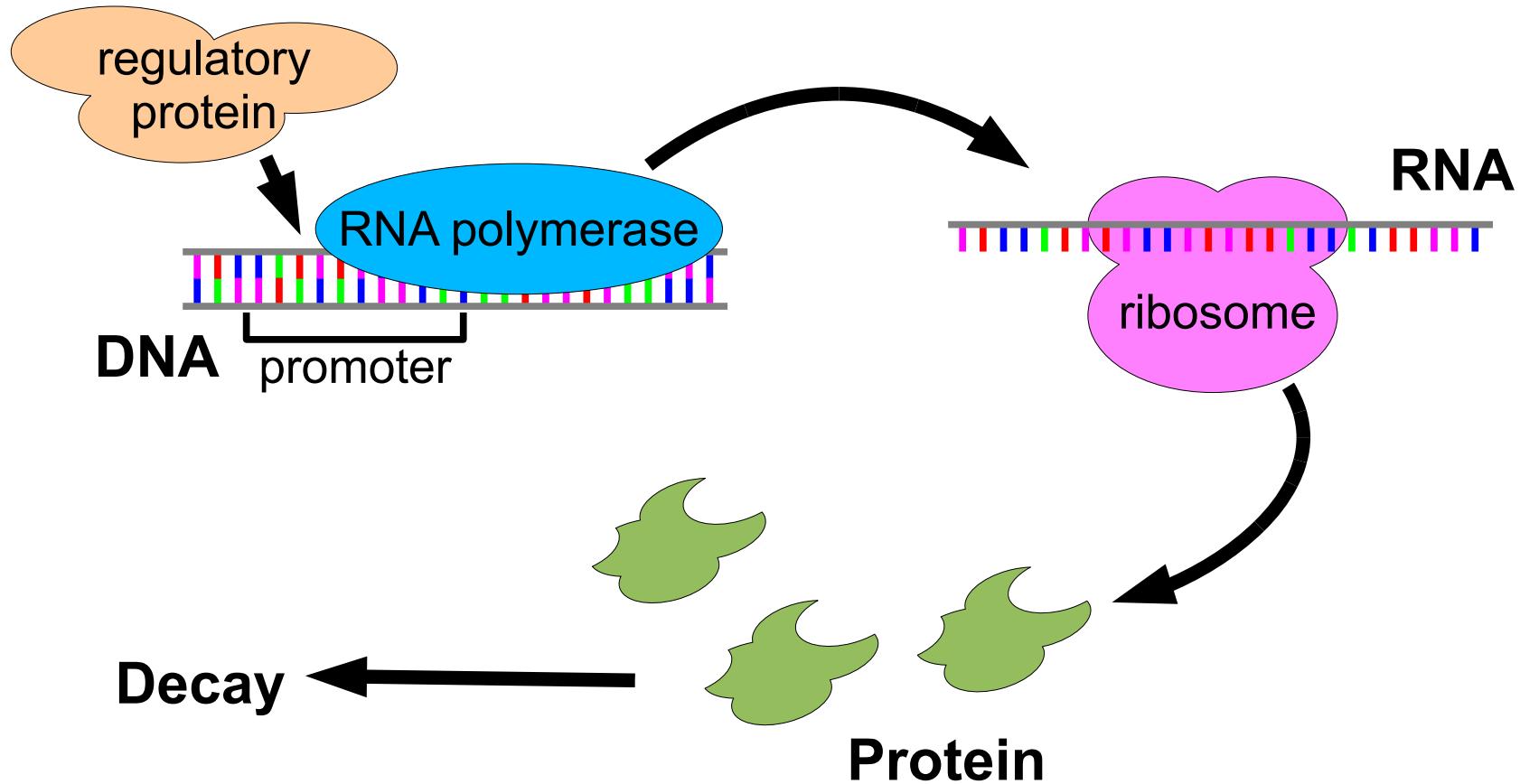
- Purely functional = simpler composition
- Self-stabilizing geometric algorithms can be composed feed-forward
- Approximation error can be predicted



# Outline

- What is Spatial Computing?
- Global → Local → Global
- From Space to Robustness & Scalability
- **The Biological World**

# Computation via Transcription Network



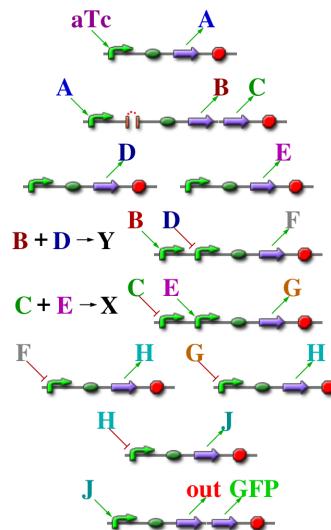
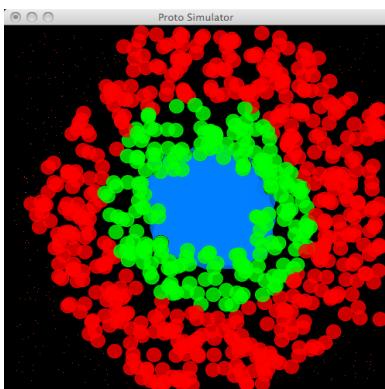
# Proto BioCompiler

## High-Level Language

```
(def band-detector (signal lo hi)
  (and (> signal lo)
       (< signal hi)))
(let ((v (diffuse (aTc) 0.8 0.05)))
  (green (band-detect v 0.2 1)))
```

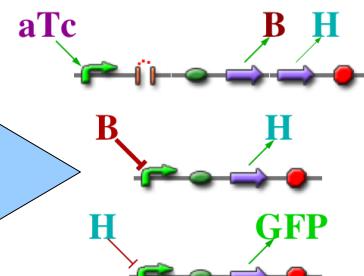
Compile

Simulate

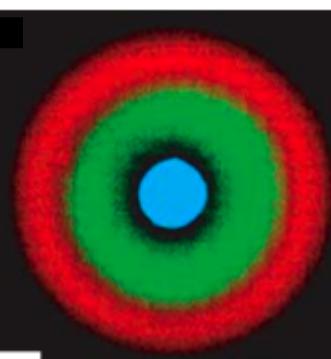


## Genetic Regulatory Network

Optimize



Assemble



w. Weiss

# Band detect: code

Proto

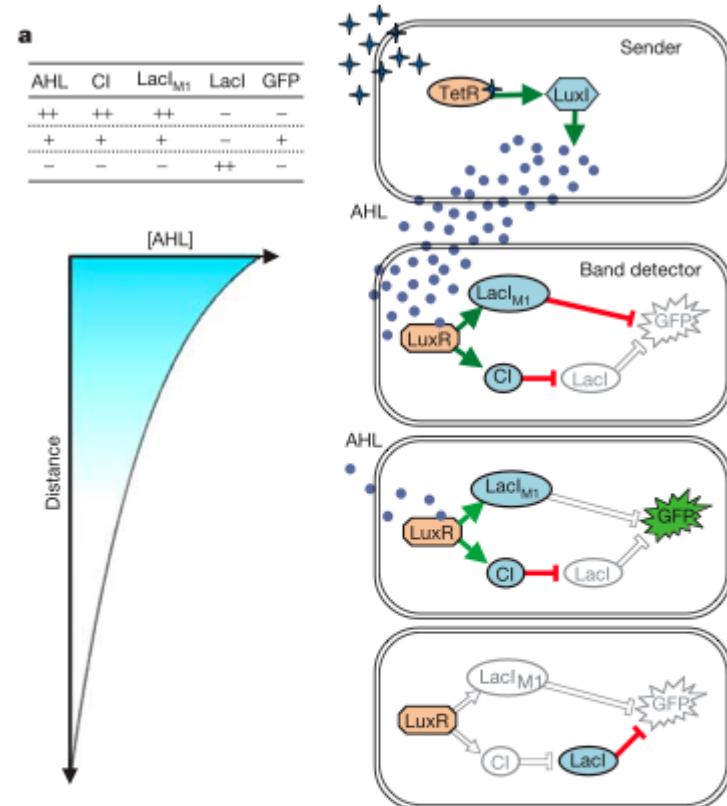
```
(def band-detector (signal lo hi)
  (and (> signal lo)
        (< signal hi)))
```

```
(let
  ((v (diffuse (aTc) 0.8 0.05)))
  (green (band-detect v 0.2 1)))
```

*simpler, more reusable*

[Beal & Bachrach, '08]

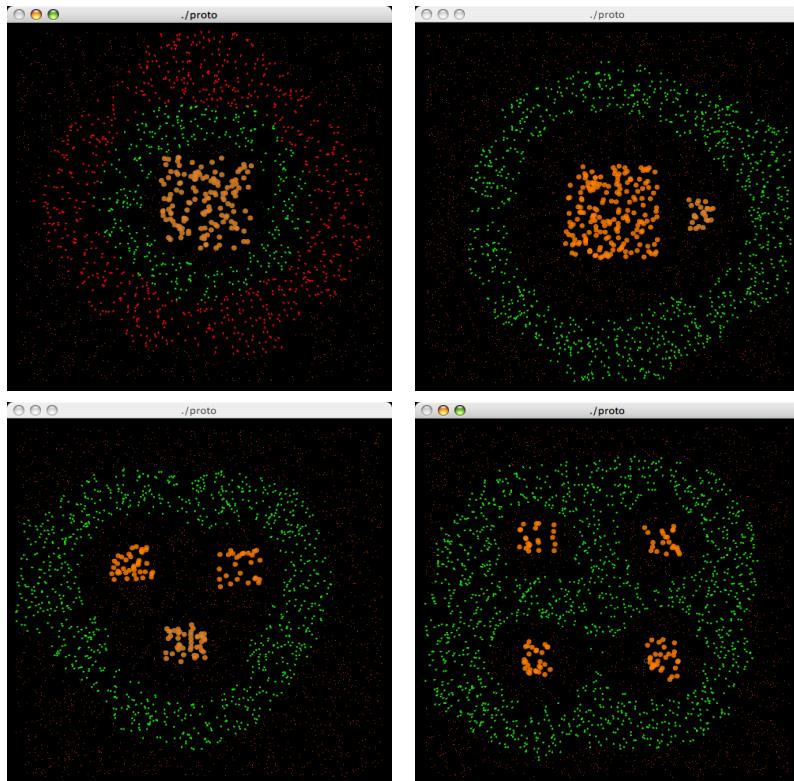
Engineered Bacteria



[Weiss '05]

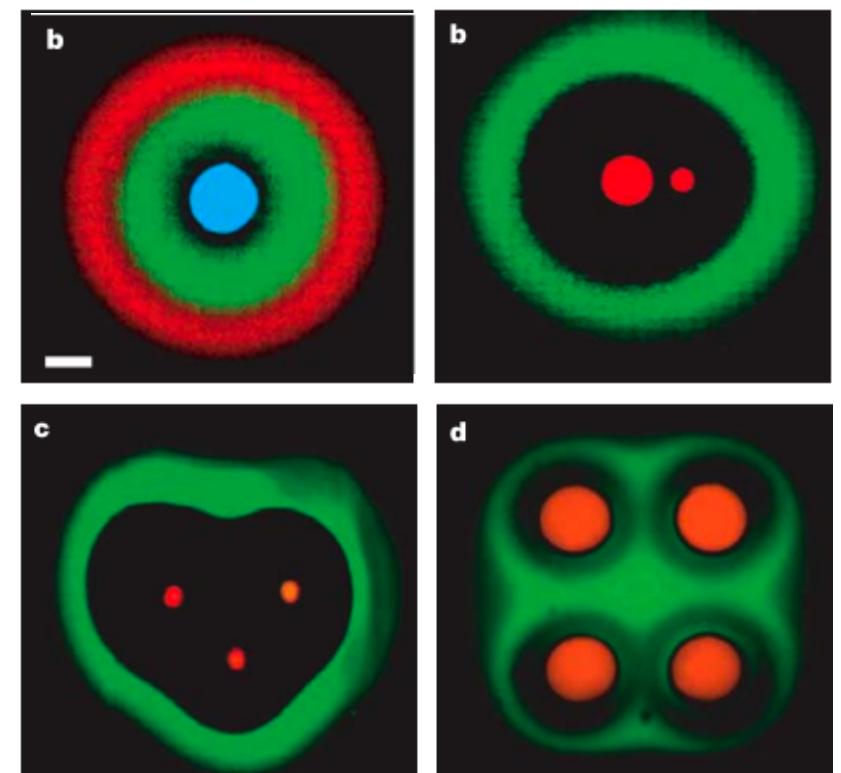
# Band detect: behavior

Proto



[Beal & Bachrach, '08]

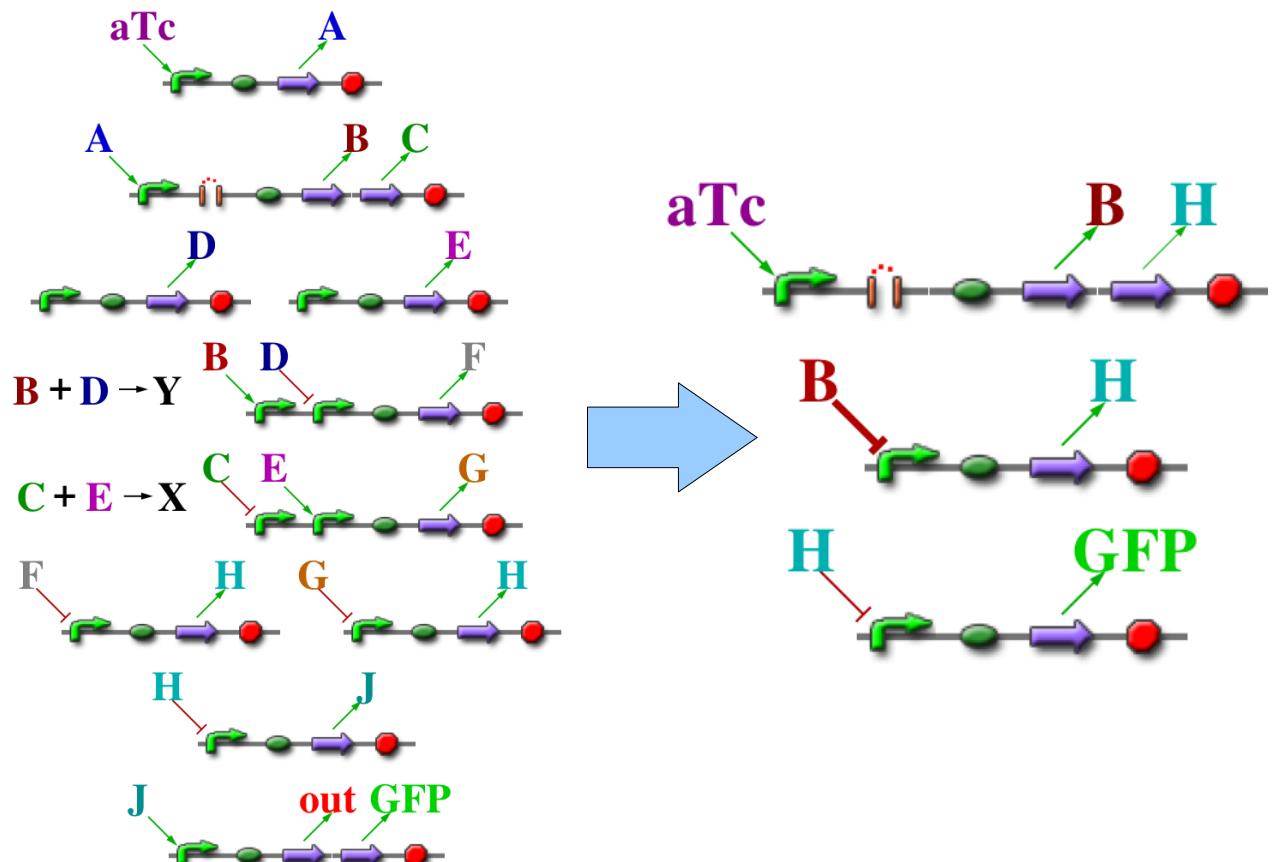
Engineered Bacteria



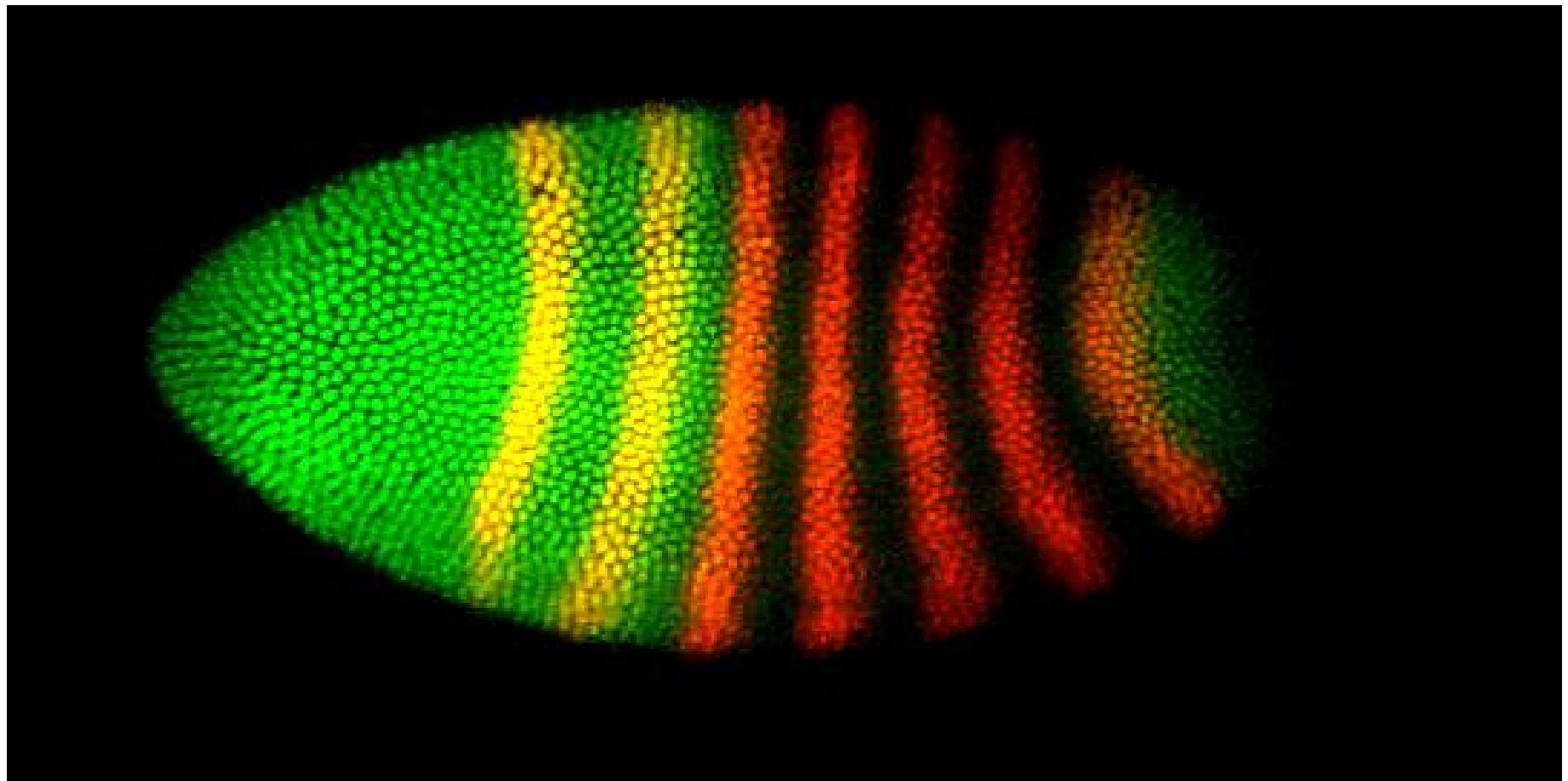
[Weiss '05]

# Classical Optimization can be Adapted

```
(def band-detector
  (signal lo hi)
  (and (> signal lo)
       (< signal hi)))
(let
  ((v (diffuse
            (aTc) 0.8 0.05)))
(green
  (band-detect v 0.2 1)))
```



# Morphogenetic Engineering



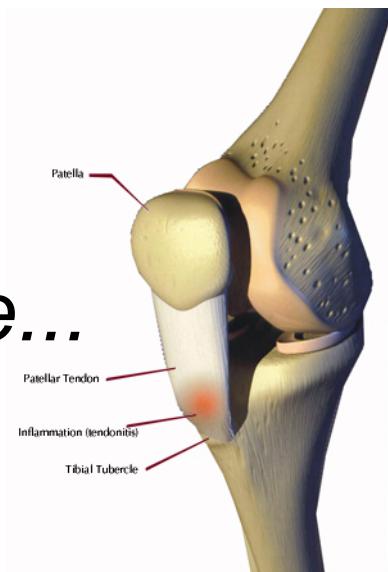
# Why doesn't growth injure animals?



Many interlinked systems

- Muscles, bones, blood, lungs, kidneys, etc...
- How is growth synchronized?
  - Not like building a house!

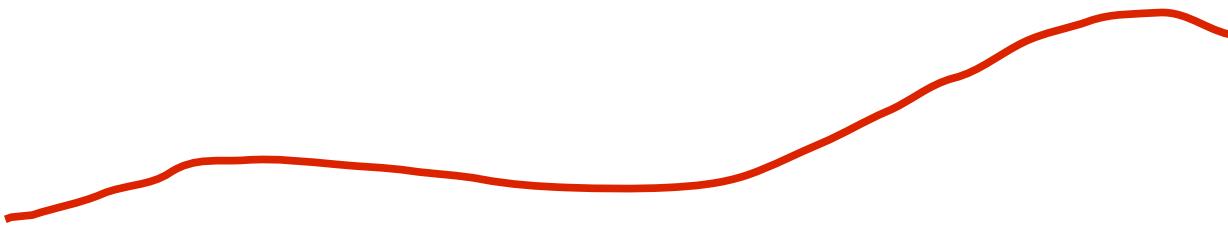
*Consider Osgood-Schlatter's disease...*



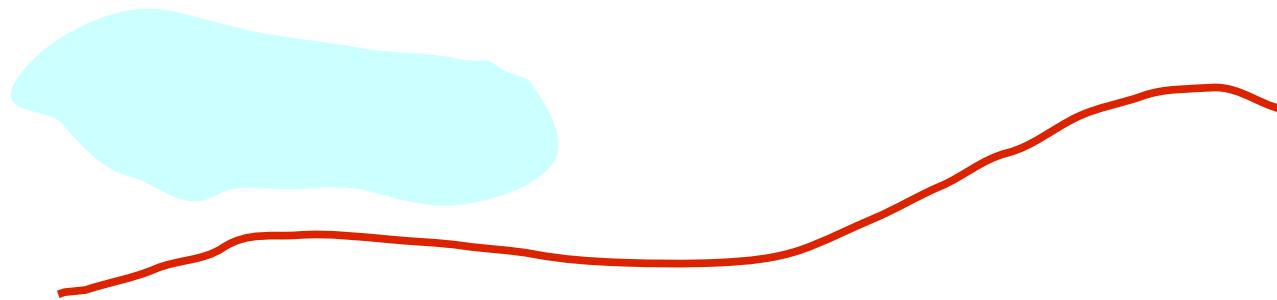
# Functional Blueprint

1. Functional behavior that degrades gracefully
2. Metric for degree and direction of stress
3. Incremental growth program for stress relief
4. Program to construct minimal initial system

# Example: Vascular System

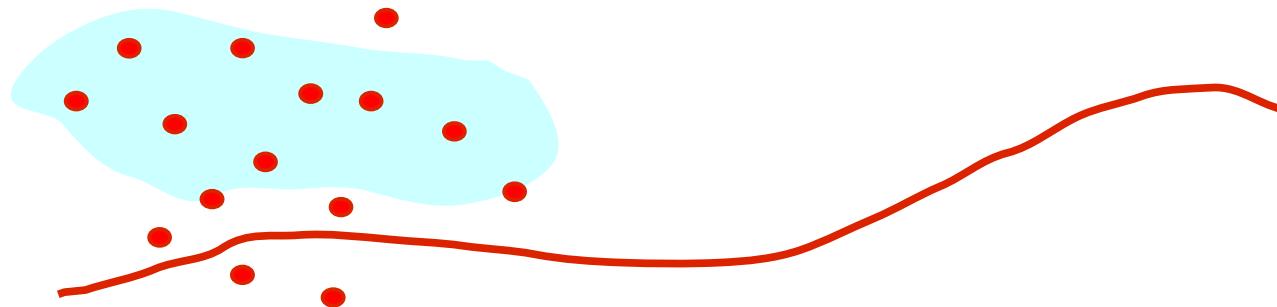


# Example: Vascular System



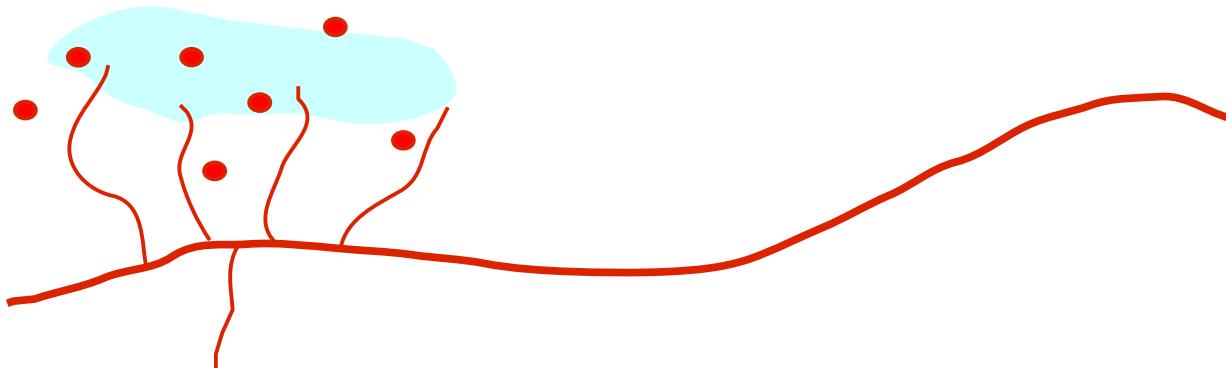
- Oxygen-starved cells signal capillary to leak

# Example: Vascular System



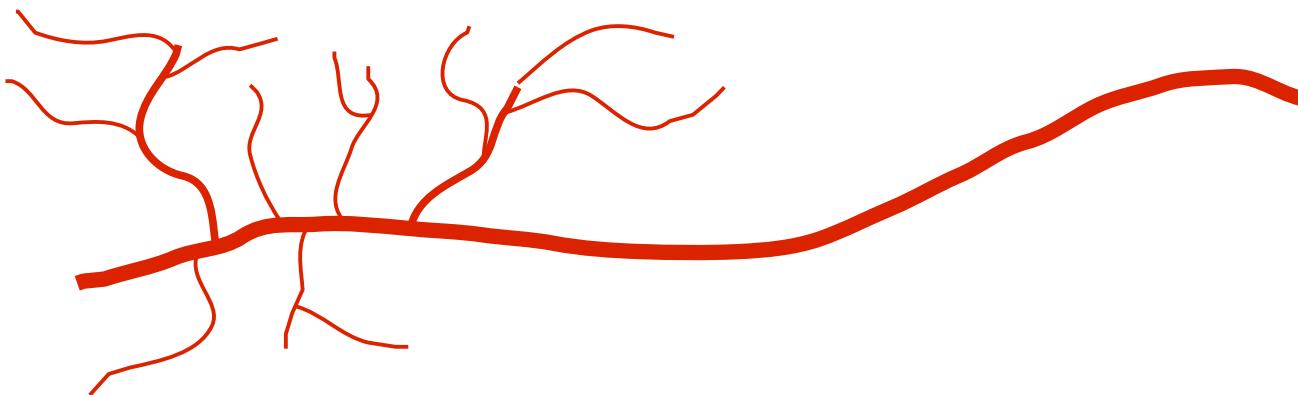
- Oxygen-starved cells signal capillary to leak

# Example: Vascular System



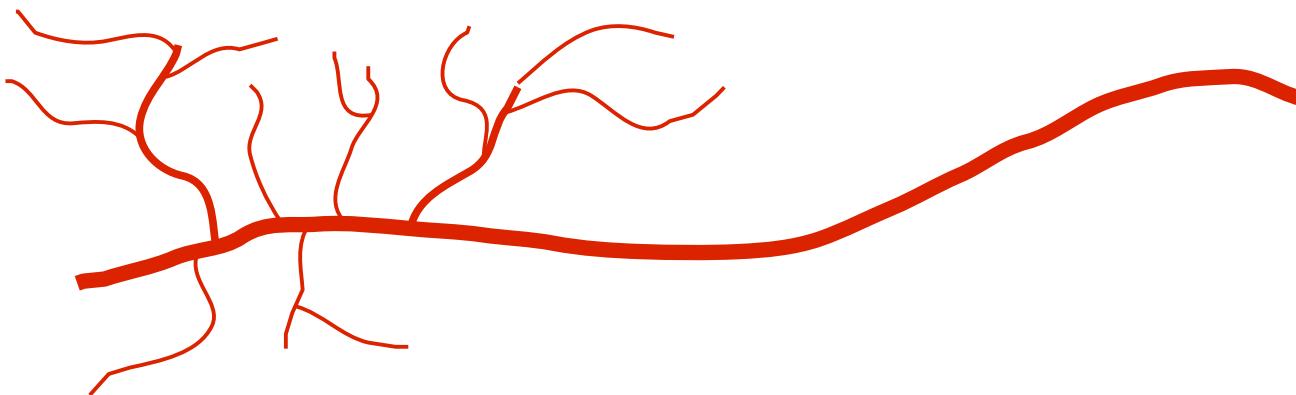
- Oxygen-starved cells signal capillary to leak
- New capillaries sprout

# Example: Vascular System



- Oxygen-starved cells signal capillary to leak
- New capillaries sprout
- Blood vessels are elastic: persistent stretch triggers growth; persistent slack shrinks

# Example: Vascular System

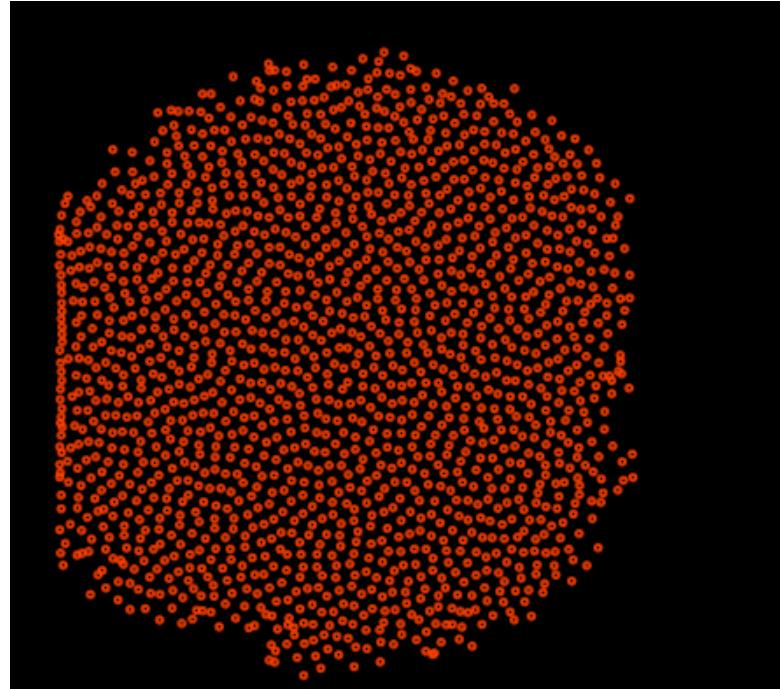


- Oxygen-starved cells signal capillary to leak
- New capillaries sprout
- Blood vessels are elastic: persistent stretch triggers growth; persistent slack shrinks

*Metric: oxygen, elastic stress*

*Homeostatic: leaking, vessel grow/shrink*

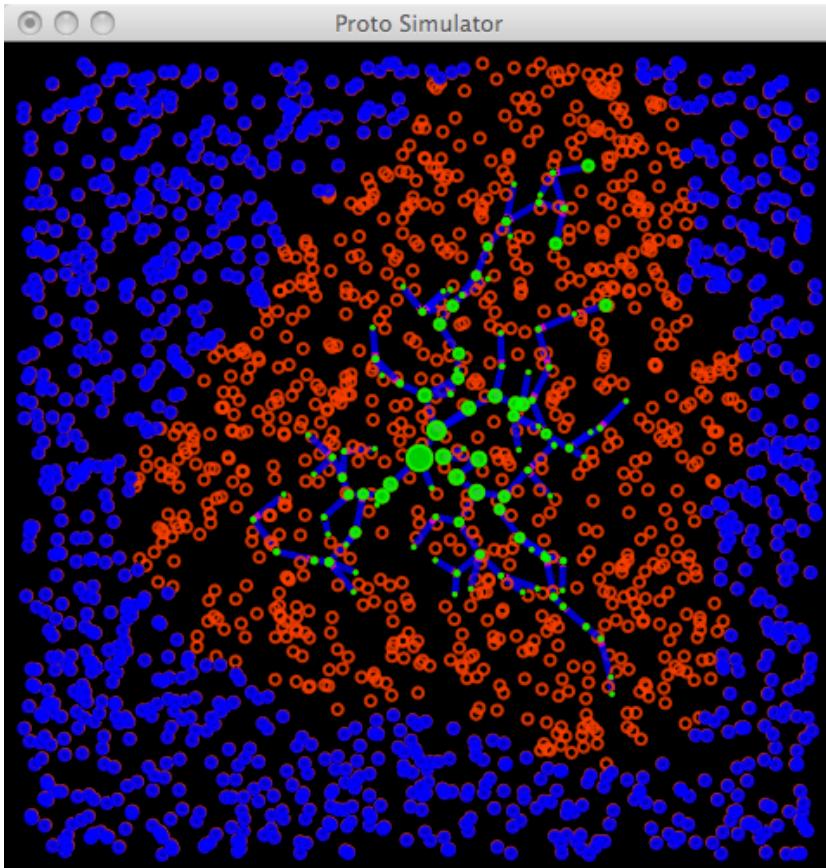
# Growth via Density Maintenance



```
(def simple-tissue ()  
  (let ((packing (num-nbrs)))  
    (clone (and (< packing 8) (< (rnd 0 1) 0.1)))  
    (die (and (> packing 12) (< (rnd 0 1) 0.1))))  
    (disperse 0.9)))
```

```
proto "(mov (simple-tissue))" -m -s 0.1 -dist-dim -25 -15 -5 5 -dim 500 500 -rad 2 -w
```

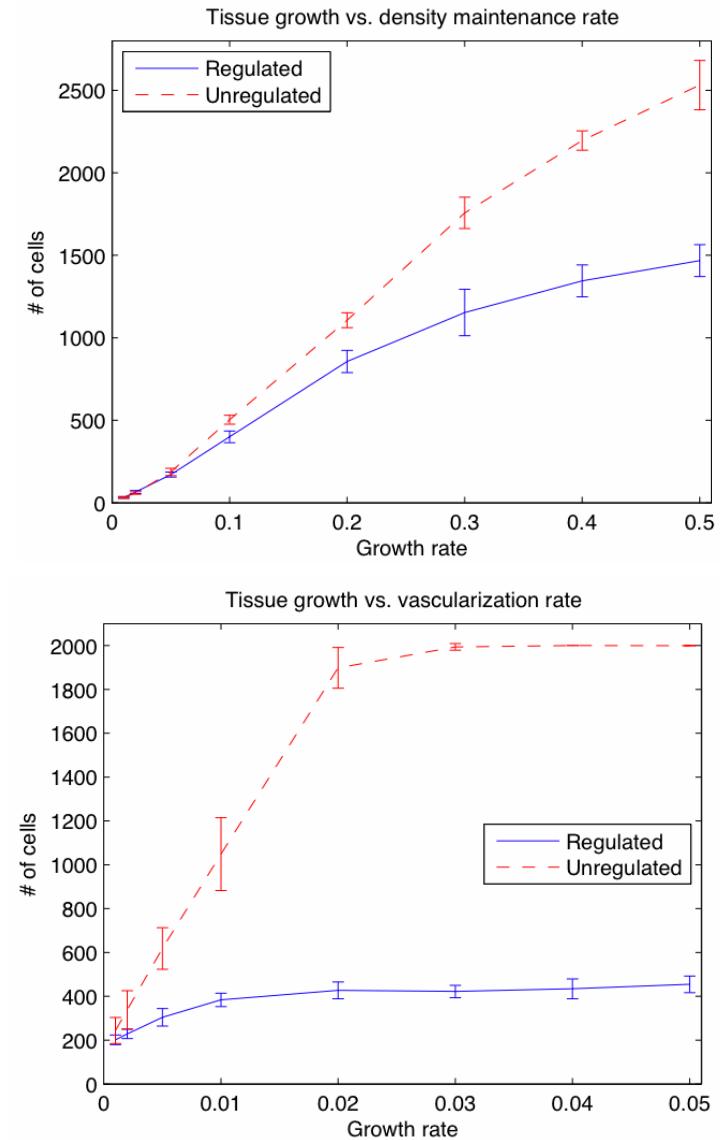
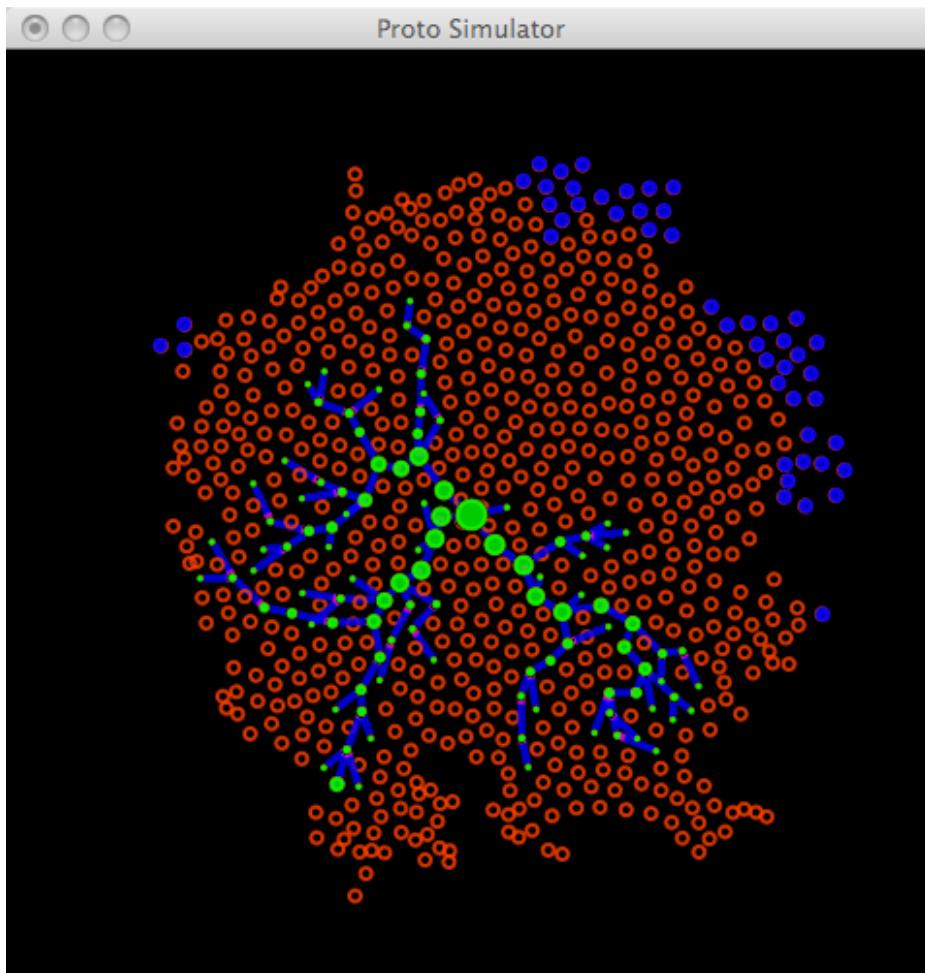
# Vascularization



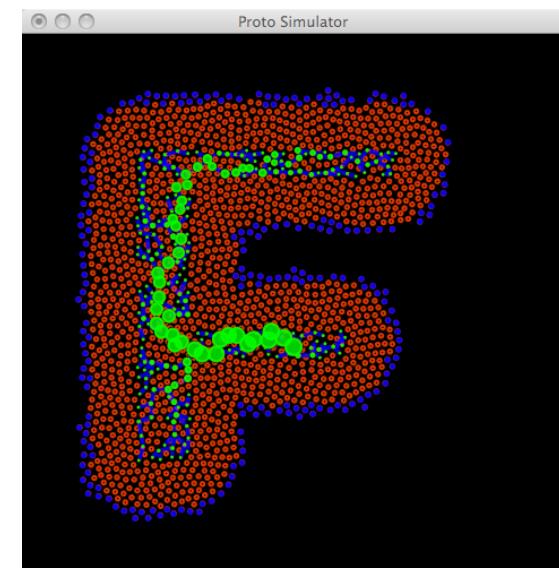
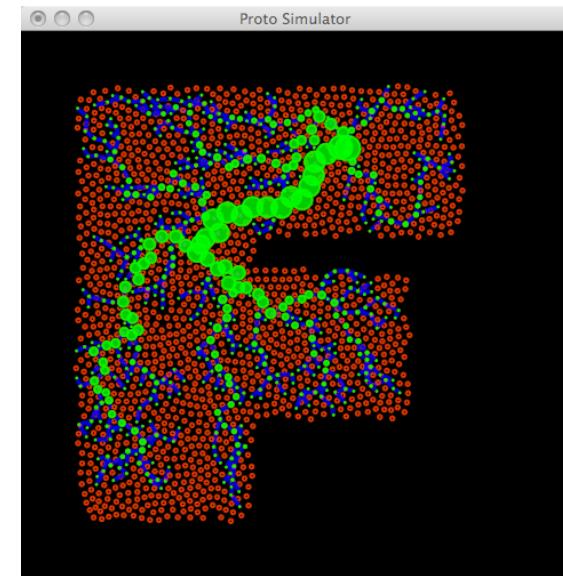
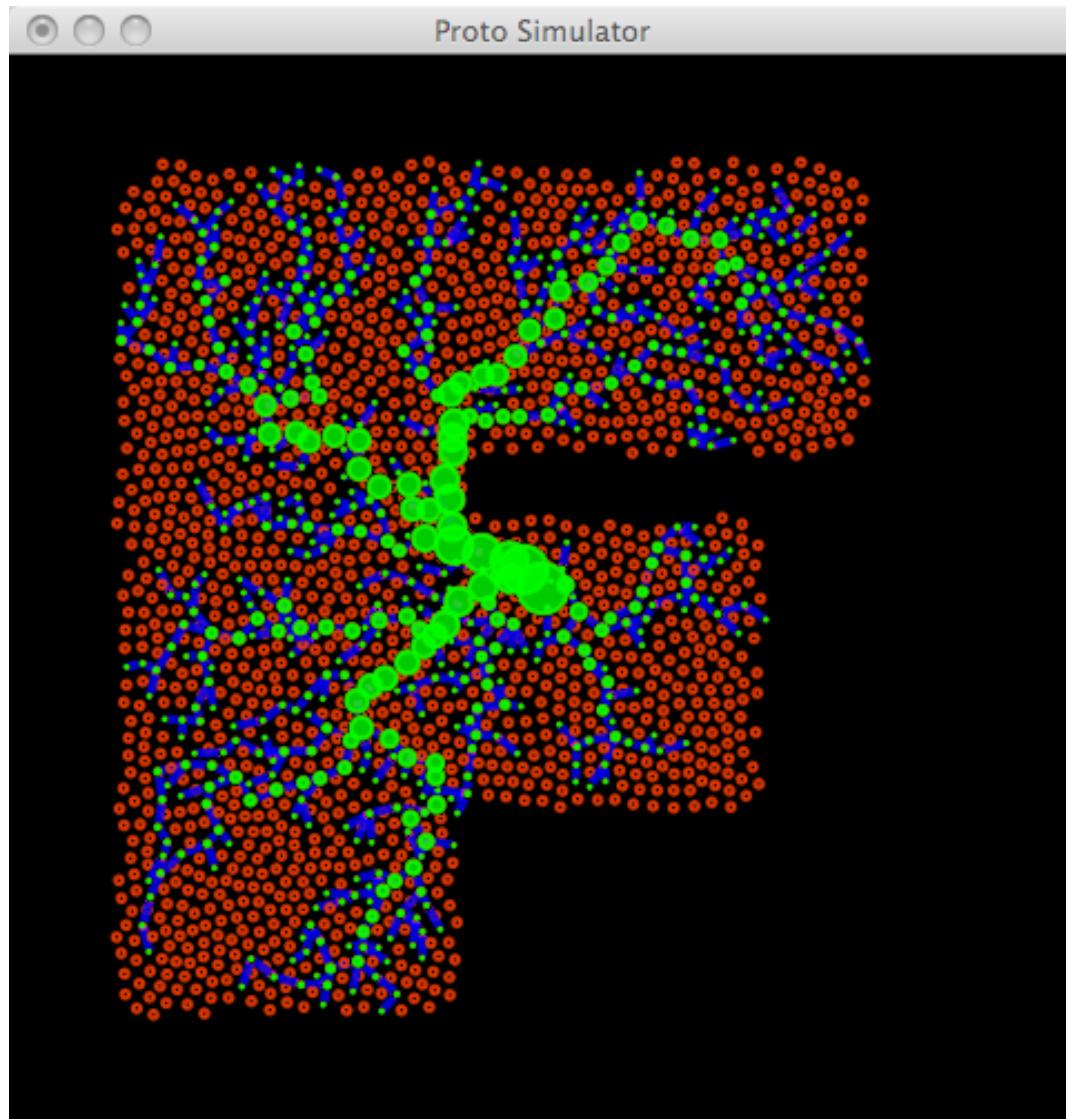
```
(def vascularize (source serv-range)
  (rep (tup vessel served parent)
    (tup source source (if source (mid) -1))
    (mux source
      (tup 1 1 -1)
      (let ((service (< (gradient vessel) serv-range)))
        (server (gradcast vessel (mid)))
        (children (sum-hood (= (mid) (nbr parent))))))
    (mux vessel
      (mux (or (muxand (any-hood (and (= (nbr (mid)) parent)
          (> (nbr children) 2)))
        (< (rnd 0 1) 0.1))
        (not (any-hood (= (nbr (mid)) parent))))
      (tup 0 1 -1) ; vessel is discarded
      (tup 1 1 (probe parent 0))) ; vessels stay fixed
    (mux (muxand (muxand (any-hood (nbr vessel))
      (dilate (not served) serv-range))
      (< (rnd 0 1) 0.02))
      (tup 1 1 server)
      (tup 0 service -1)))))))
```

```
proto "(let ((v (vascularize (sense 1) 50))) (green (1st v)) (blue (not (2nd v))))" -n 200 -l -s 0.1 -m
```

# Vascularization/Density Co-Regulation



# Modular Integration



# Summary

- The Amorphous Medium abstraction simplifies programming of scalable, robust behavior on space-filling networks
- Proto has four families of space and time operations, compiles global descriptions into local actions that approximate the global
- Geometric metaphors allow complex spatial computing problems to be solved with very short programs.
- Spatial abstractions enable imports from computation to biology and vice versa.

# Proto is available

**<http://stpg.csail.mit.edu/proto.html>**  
(or google “MIT Proto”)

- Includes libraries, compiler, kernel, simulator, platforms
- Licensed under GPL (w. libc-type exception)