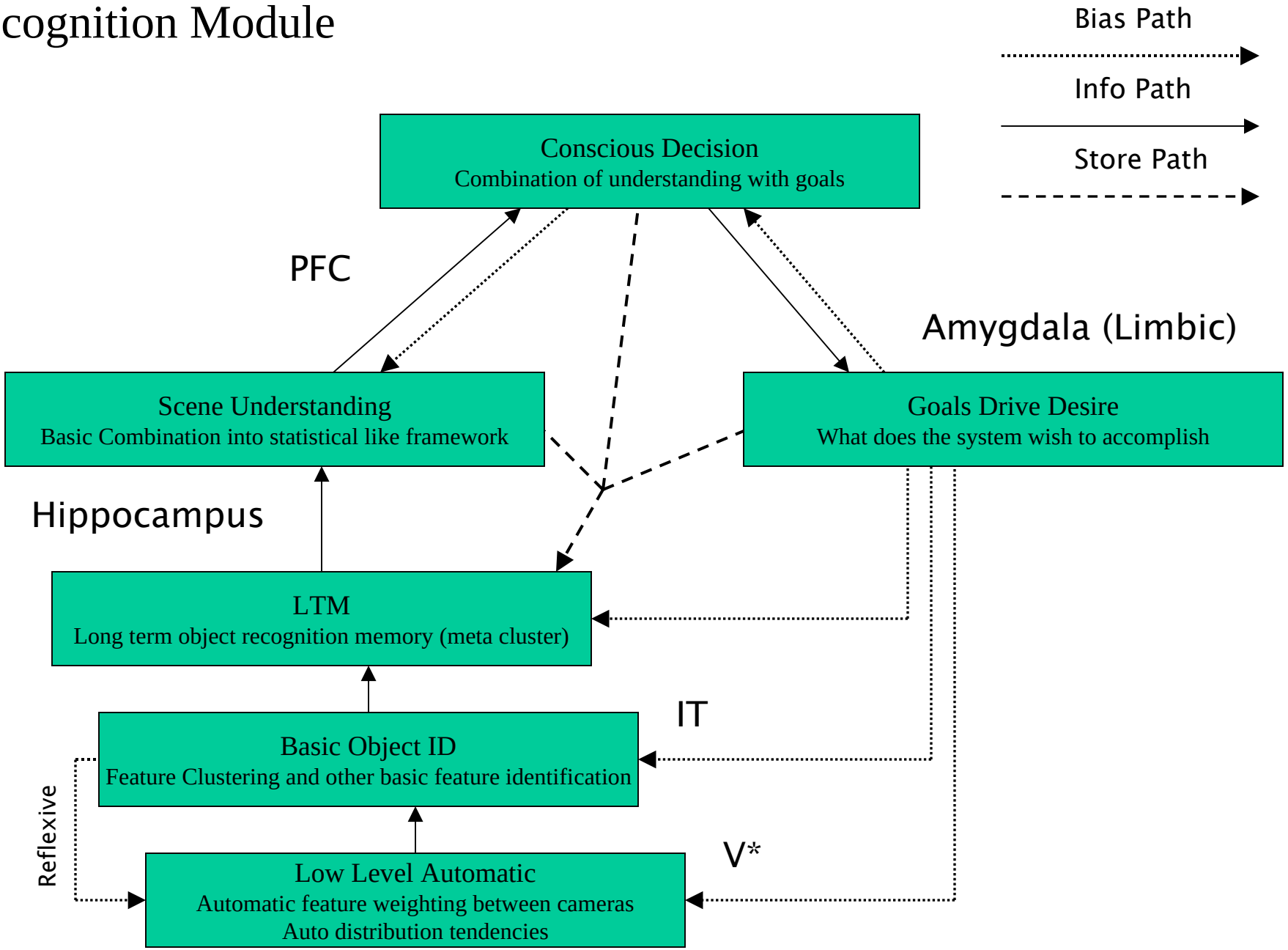
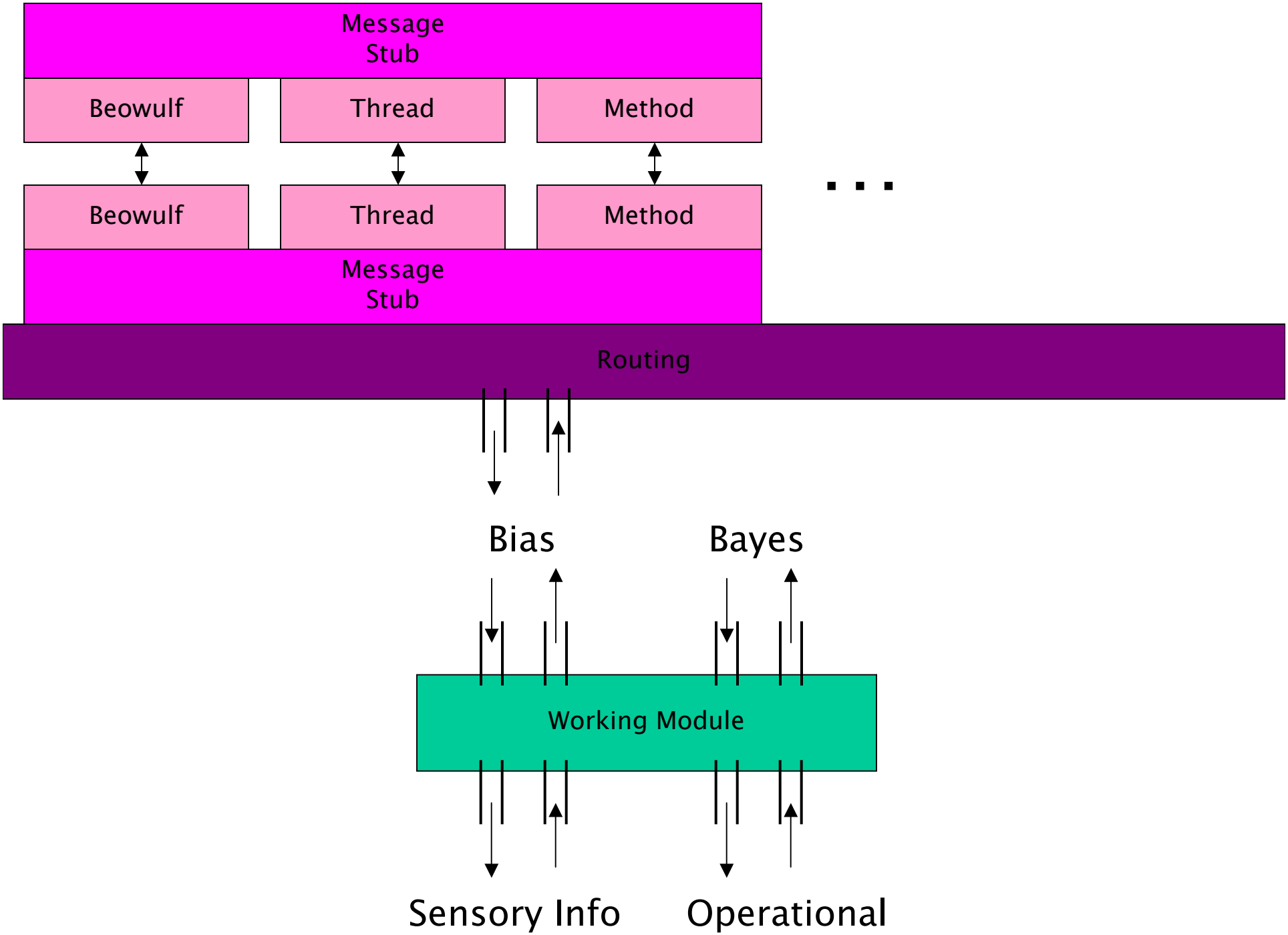
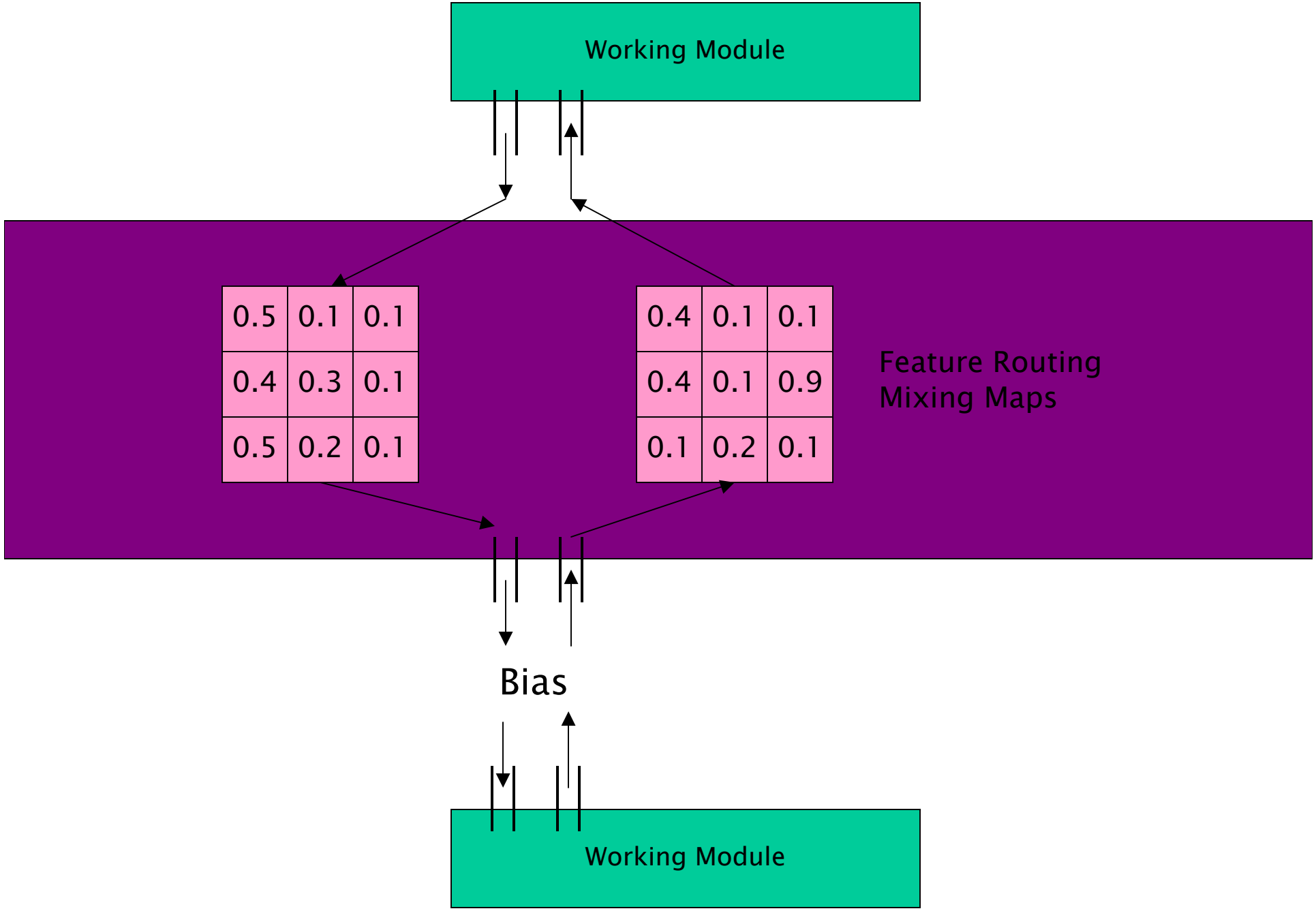
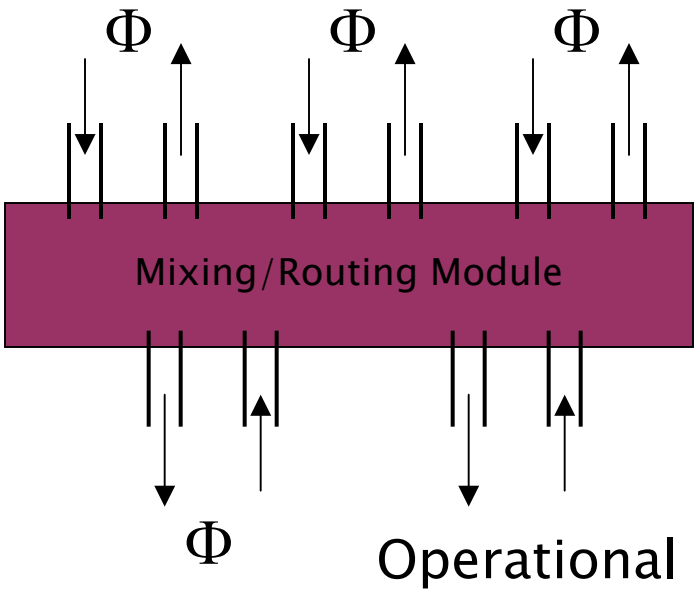
Recognition Module

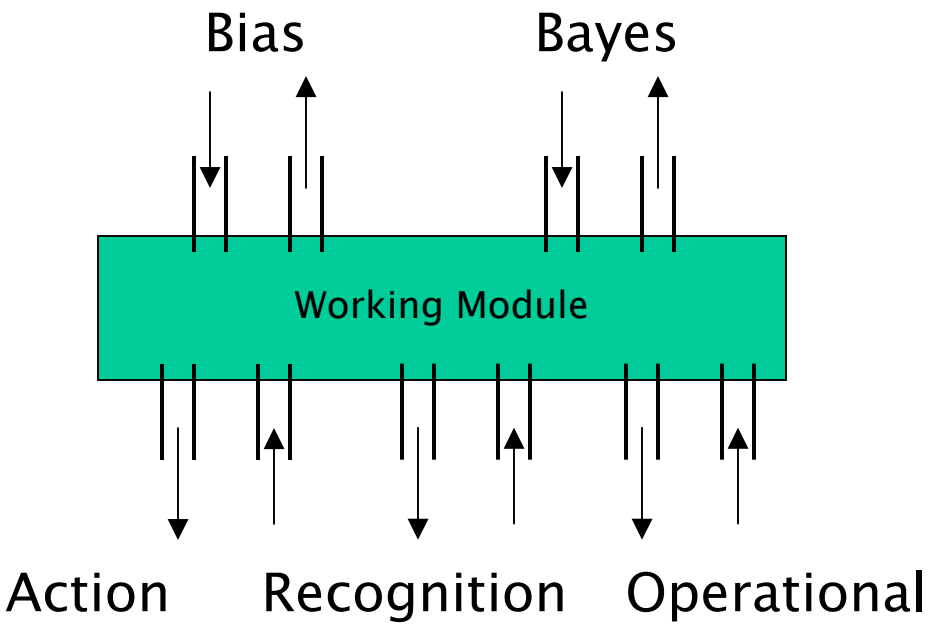


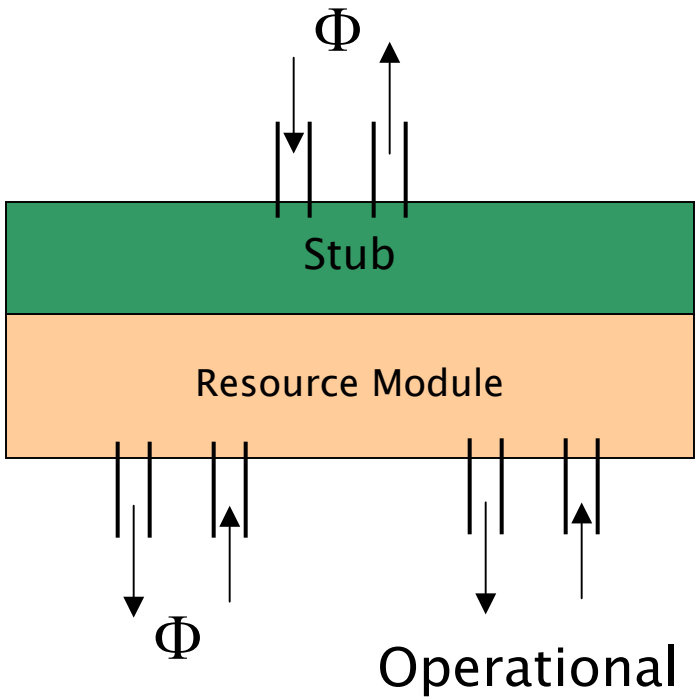
T Nathan Mundhenk

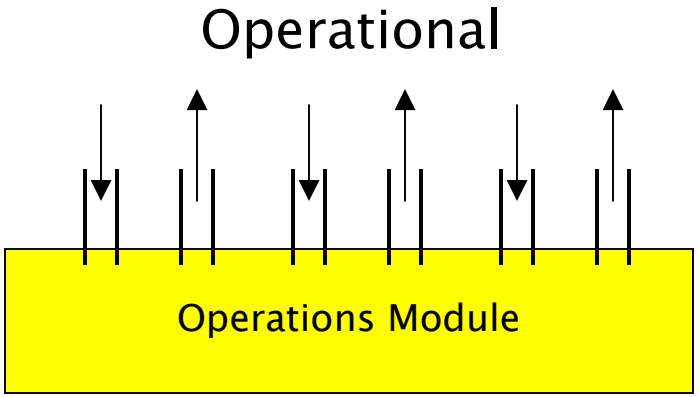

Message
Stub
Beowulf
Thread
Method
Beowulf
Thread
Method
Message
Stub
Routing
Bias
Bayes
Working Module
Sensory Info
Operational


Working Module
0.5 0.1 0.1
0.4 0.3 0.1
0.5 0.2 0.1
0.4 0.1 0.1
0.4 0.1 0.9
0.1 0.2 0.1
Feature Routing
Mixing Maps
Bias
Working Module


Φ
Φ
Φ
Mixing/Routing Module
Φ
Operational

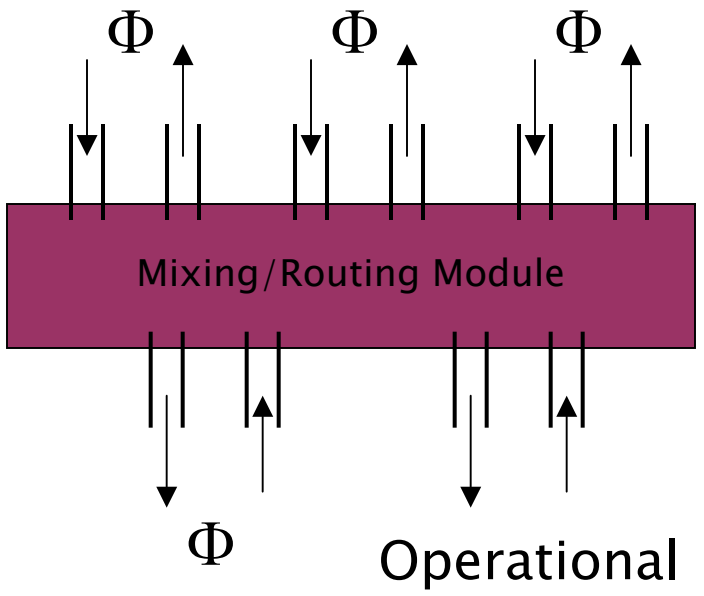

Bias
Bayes
Working Module
Action
Recognition
Operational


Φ
Stub
Resource Module
Φ
Operational


Operational
Operations Module

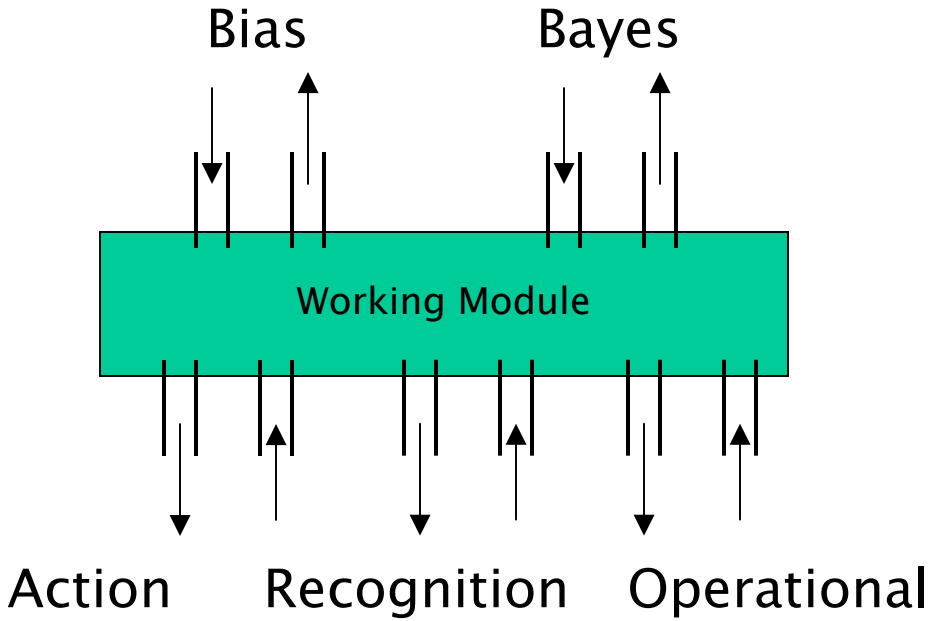
Mixing/Routing Module

- Defines how outputs are routed to and from different modules
- Can instantiate linear and non-linear matrix mixing
- Is derived for the different data types from a parent class
- Is designed to be a super mux/de-mux
- May mix different data types so long as it is logical and symple
  - Otherwise this must be a working module



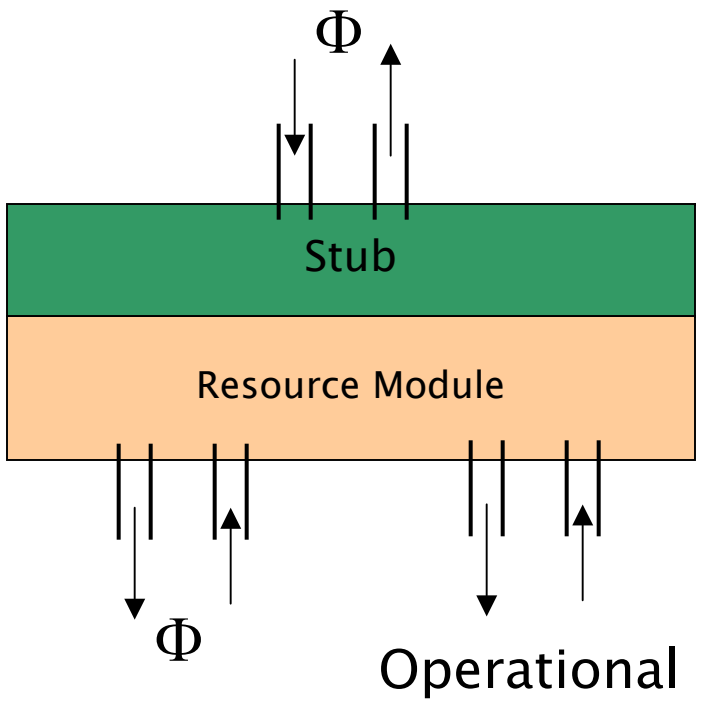
Working Module

- This is a standardized wrapper for complex functions.
- We will define a set of all information types to be passed.
- Emphasis will be on the fact that all working modules can be biased and must understand Bayesian workings.
- Recognition input will most likely derive from the image class
- Are linked explicitly by Mixing/Routing modules

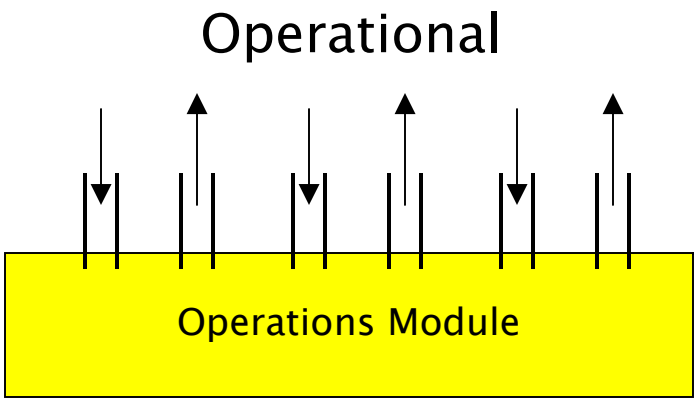


Resource Module

- Connects other modules to a variable resource.
- Example: The output from a working module may need to be sent to a thread or another Beowulf cluster
- Operational input may be only optional since the input Φ may include it's preference for resource usage.
- A working example may be the grabber class
- The stub should be generic.



Operations Module (*non-brain*)



- Provides control interface between the brain and other non-brain resources such as hardware (via resource module) and user interfaces.
- Is properly defined as a traditional control mechanism (contains rules)

Bias Channel

- Is a column vector per output of all biases as floating points. Is a set of vectors for multiple outputs.
- Are mixed and routed via matrix methods.
- Bias weights should be normalized between modules (from 1 to –1).

Bayes Channel

- Contains Eigen vector/ values that define probabilities
- Contain global or local labels for probabilities (e.g. the label is only shared between two modules or many or all)
- All probabilities are normalized
- Defines the similarity of all labels. That is how much alike are different measures.
  - This allows Prob's to be generalized and morphed
  - In essence defines relationships between probability measures.

Recognition Channel

- Uses the image class as a basis ,but information may be abstracted beyond a recognizable image
- Outputs single frames of data

Action Channel

- Contains spatially relevant action commands.
- Action commands must be influencable
- Commands should be ego centric

Operations Channel

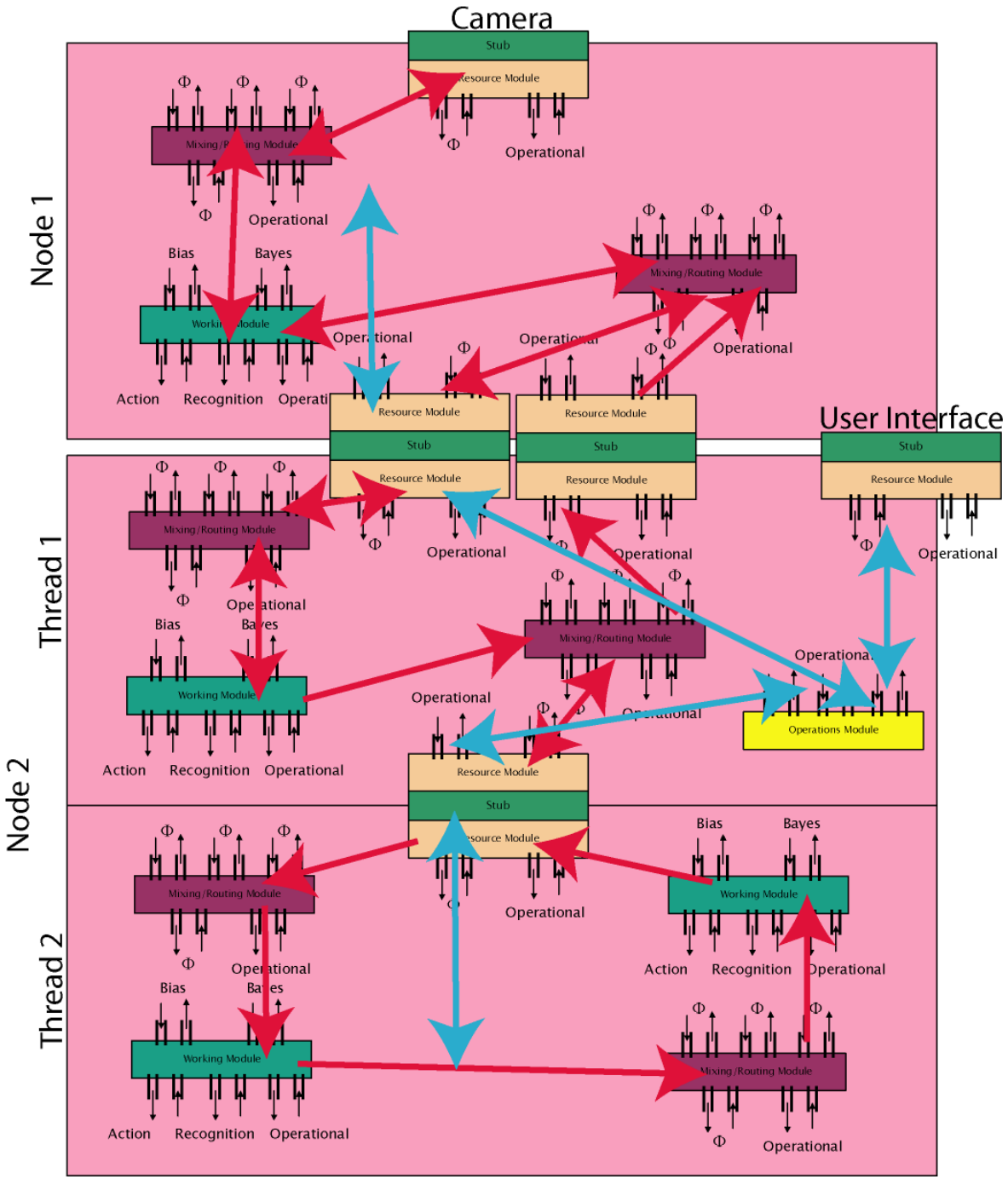
- Contains lists of parameters and their corresponding values
- Each parameter should correspond to a global set of known parameters.
- Should be instantiated by the Model Manager
- Is most likely already implemented. Just needs some augmentation to interact with resource models?
- Should this contain a synchronization channel between resource modules or should that be another channel?

Other channels

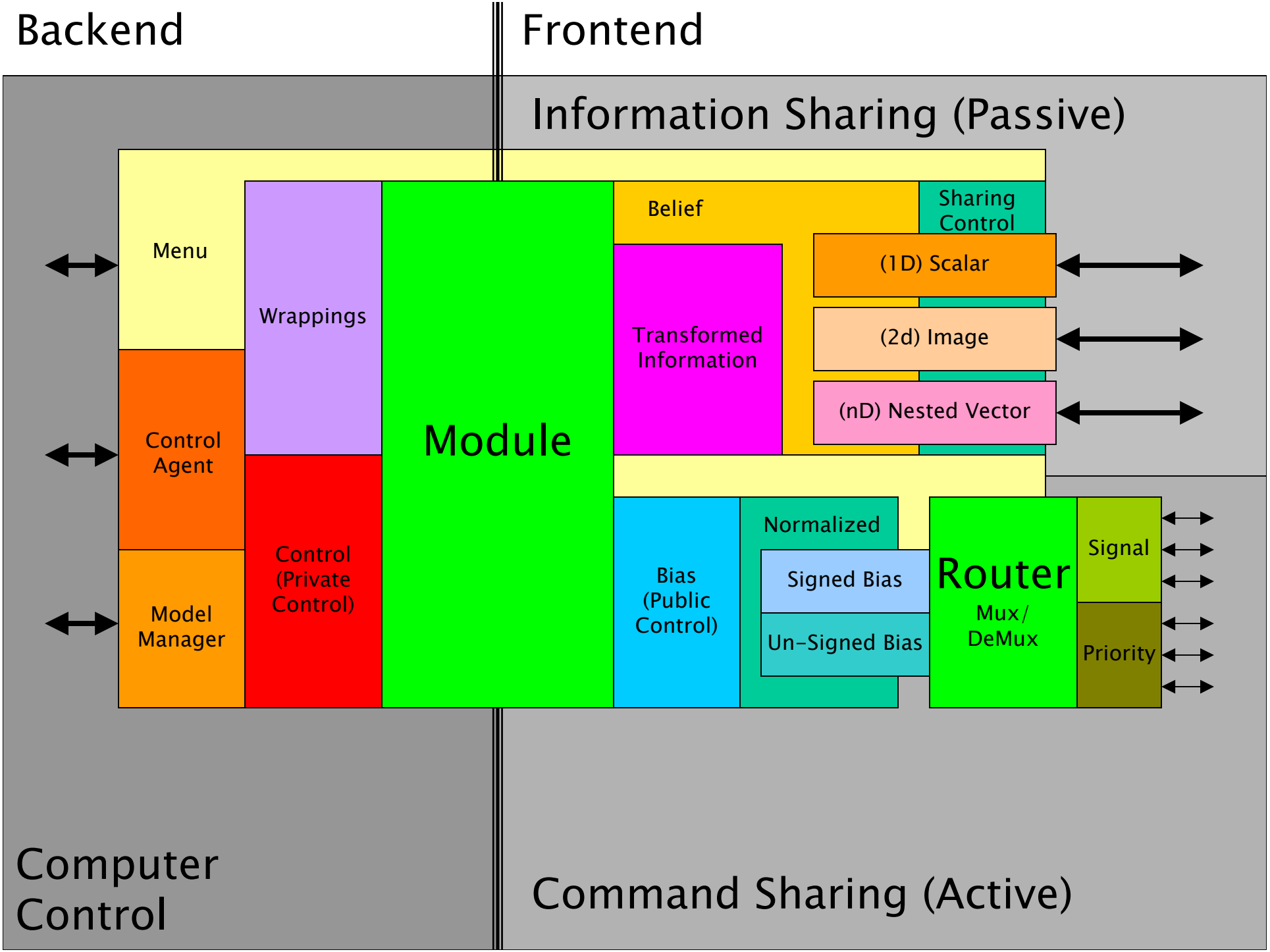
- Other channels may be defined so long as they are generic and cannot be logically accounted for with the other channels. As such EVERY module must be augmentable if for no other reason than to add a stub.

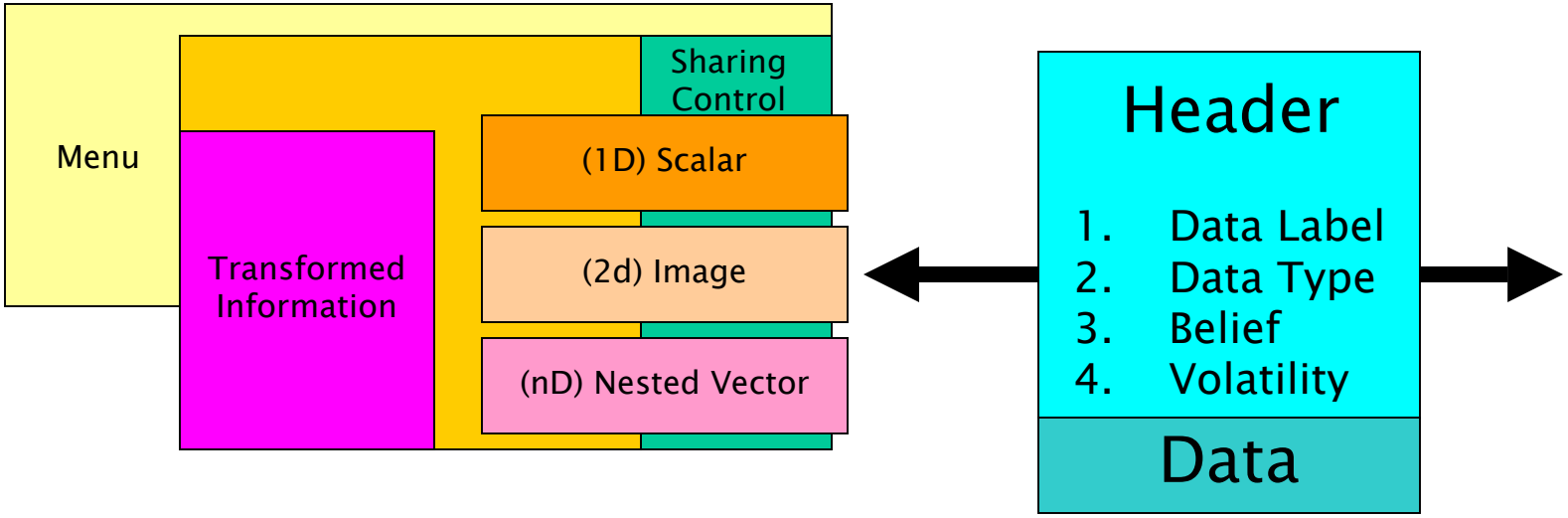
Connection Agent

- Instantiation of wrappings which links modules?
- It is assumed that all information behind a resource module is finite state (thus, threads must communicate via a resource module).
  - All information at this level is managed via copy on write

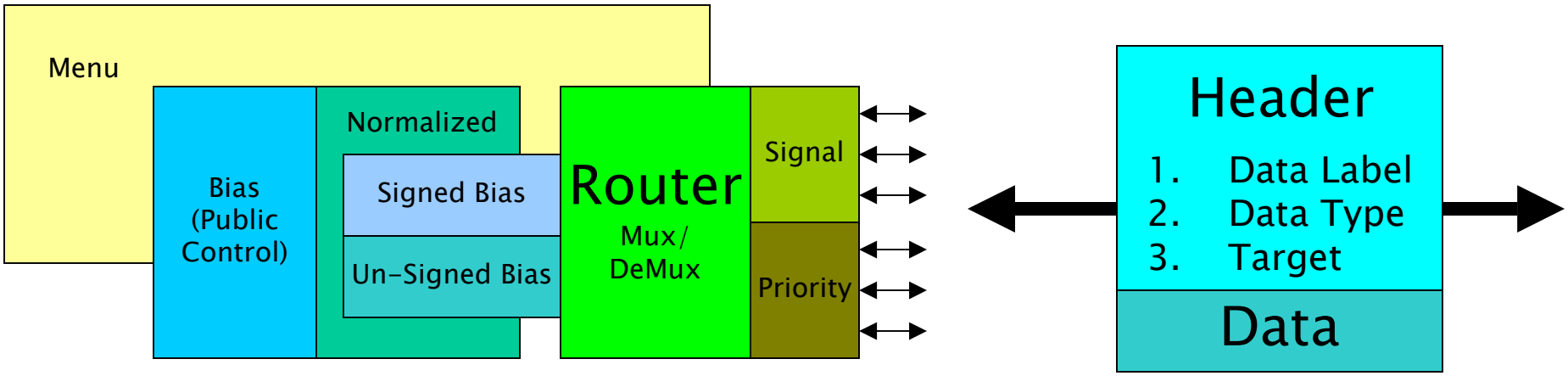

Camera
Stub
Resource Module
Operational
Node 1
Mixing/Routing Module
Bias
Bayes
Working Module
Action
Recognition
Operational
Thread 1
Thread 2
Node 2
User Interface
Operations Module

iRoom expert module standard
IEMS

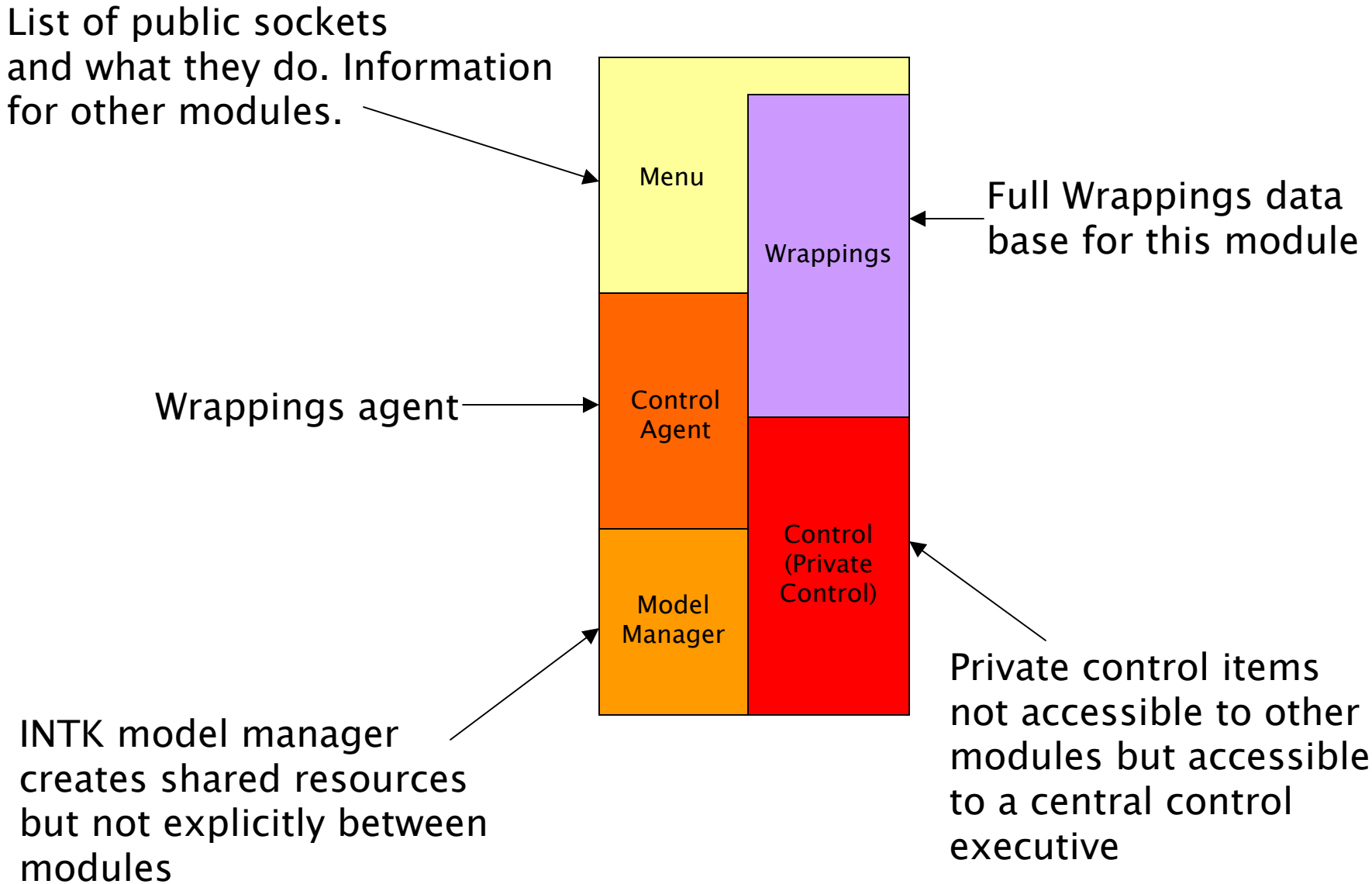

Backend
Frontend
Information Sharing (Passive)
Menu
Wrappings
Module
Belief
Sharing Control
(1D) Scalar
(2d) Image
(nD) Nested Vector
Transformed Information
Control Agent
Control (Private Control)
Model Manager
Bias (Public Control)
Normalized
Signed Bias
Un-Signed Bias
Router
Mux/ DeMux
Signal
Priority
Computer Control
Command Sharing (Active)



- Data Label – This is what the data called in the menu, for instance “saliency map”
- Data Type – This defines the dimension of the data. All scalars are doubles, all 2D data are images and all nD data are nested vectors.
- Belief – This is the certainty that the data is correct. It attaches a probability to all results.
- Volatility/Sharing – This is the resource management pointer. For each instance where information is pointed to but not copied, the receiver must be finished before the data is altered.



- The bias is the way in which modules “command” each other.
- Biases are either 0 to 1 or -1 to 1. Thus, they are normalized signed or unsigned. All biases are doubles.
- All biases are routed. This allows more than one module to bias another module on one channel. The router mixes and unmixes the bias. The mixing method is defined by the module author.
- Inputs and outputs for biases are separated into two parts. The amount of bias (signal) and the priority of the bias. This is like the difference between what is said and how loud it is said.
- The Bias mixing can be biased as well.


List of public sockets
and what they do. Information
for other modules.
Menu
Wrappings
Full Wrappings data
base for this module
Wrappings agent
Control
Agent
Control
(Private
Control)
Model
Manager
INTK model manager
creates shared resources
but not explicitly between
modules
Private control items
not accessible to other
modules but accessible
to a central control
executive