Simplifying Neurorobotic Development with NCSTools

C.M. Thibeault[1,2,3], J. Hegie[2], L. Jayet Bray[2,3], and F.C. Harris Jr.[1,2,3]

[1] Department of Electrical and Biomedical Engineering, University of Nevada, Reno. Reno, NV.
[2] Department of Computer Science & Engineering, University of Nevada, Reno. Reno, NV.
[3] Brain Computation Lab, University of Nevada, Reno. Reno, NV.

UNR Brain Lab

Faculty:

Laurence Jayet Bray

Sergiu Dascalu

Frederick Harris Jr.



UNR Brain Lab

Faculty:

Laurence Jayet Bray

Sergiu Dascalu

Frederick Harris Jr.



Graduate Students:

Gareth Ferneyhough

Roger V. Hoang


UNR Brain Lab

Faculty:

Laurence Jayet Bray

Sergiu Dascalu

Frederick Harris Jr.



Graduate Students:

Gareth Ferneyhough

Roger V. Hoang


Undergraduate Students:

Emily Barker

Kimberly Perry

Nathan Jordan

Nitish Narala

UNR Brain Lab

Faculty:

Laurence Jayet Bray

Sergiu Dascalu

Frederick Harris Jr.



Graduate Students:

Gareth Ferneyhough

Roger V. Hoang


Undergraduate Students:

Emily Barker

Kimberly Perry

Nathan Jordan

Nitish Narala

Project Collaborators :

Corey M. Thibeault

Joshua Hegie

UNR Brain Lab

Faculty:

Laurence Jayet Bray

Sergiu Dascalu

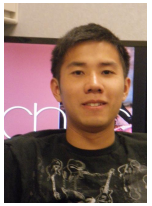
Frederick Harris Jr.



Graduate Students:

Gareth Ferneyhough

Roger V. Hoang

Undergraduate Students:

Emily Barker

Kimberly Perry

Nathan Jordan

Nitish Narala

Project Collaborators :

Corey M. Thibeault

Joshua Hegie

Facts:

- Founded in 2001
- Joint Research Center

Funding:

- ONR
- HRL Labs

Outline

- Introduction
  - Spiking Neural Models
  - Neurorobotics and VNR
  - NCS
- NCSTools
  - Motivation
  - Design
- Examples
  - Reward-Based Learning
  - Trust the Intent Recognition
- Questions?

Outline

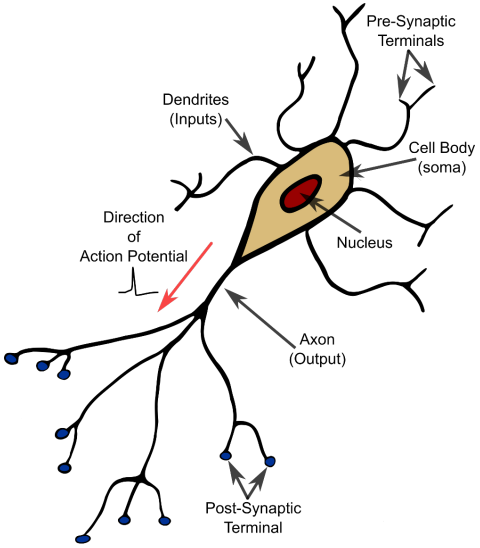
- Introduction
  - Spiking Neural Models
  - Neurorobotics and VNR
  - NCS

Introduction

- What is Computational Neuroscience?

Introduction

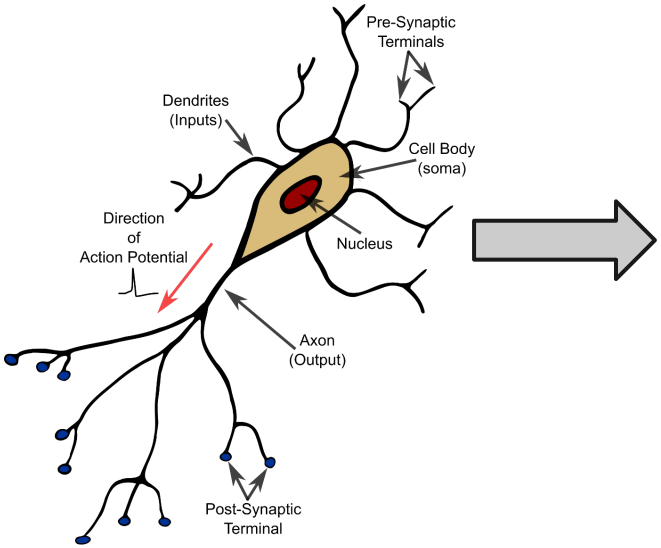
- What is Computational Neuroscience?



Introduction

- What is Computational Neuroscience?

$$I = C_M \frac{dV}{dT} + \bar{g}_K n^4 (V - V_K) + \bar{g}_{Na} m^3 h (V - V_{Na}) + \bar{g}_l (V - V_l)$$



$$\frac{dn}{dt} = \alpha_n (1 - n) - \beta_n n$$

$$\frac{dm}{dt} = \alpha_m (1 - m) - B_m m$$

$$\frac{dh}{dt} = \alpha_h (1 - h) - \beta_h h$$

$$\alpha_n = \frac{0.01(V + 10)}{e^{\frac{V - 10}{10}} - 1}$$

$$\beta_n = 0.125 e^{(V/80)}$$

$$\alpha_m = \frac{0.1(V + 25)}{e^{\frac{V + 25}{10}} - 1}$$

$$\beta_m = 4 e^{(V/18)}$$

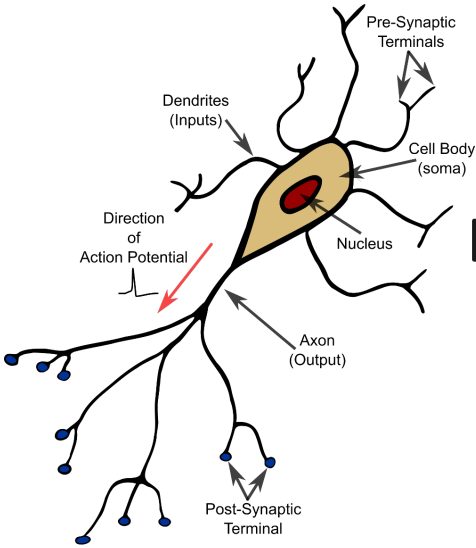
$$\alpha_h = 0.07 e^{\frac{V}{20}}$$

$$\beta_h = \frac{1}{e^{\frac{V + 30}{10}} + 1}$$

Introduction

- What is Computational Neuroscience?

$$I = C_M \frac{dV}{dT} + \bar{g}_K n^4 (V - V_K) + \bar{g}_{Na} m^3 h (V - V_{Na}) + \bar{g}_l (V - V_l)$$



$$\frac{dn}{dt} = \alpha_n (1 - n) - \beta_n n$$

$$\frac{dm}{dt} = \alpha_m (1 - m) - B_m m$$

$$\frac{dh}{dt} = \alpha_h (1 - h) - \beta_h h$$

$$\alpha_n = \frac{0.01(V + 10)}{e^{\frac{V - 10}{10}} - 1}$$

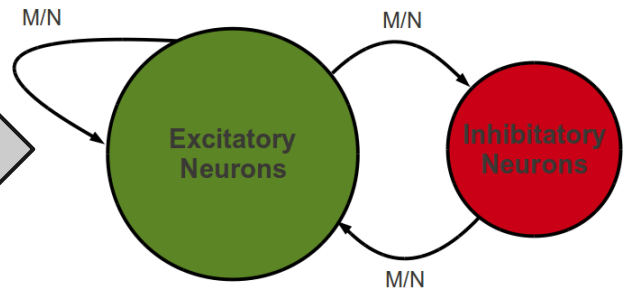
$$\beta_n = 0.125 e^{(V/80)}$$

$$\alpha_m = \frac{0.1(V + 25)}{e^{\frac{V + 25}{10}} - 1}$$

$$\beta_m = 4 e^{(V/18)}$$

$$\alpha_h = 0.07 e^{\frac{V}{20}}$$

$$\beta_h = \frac{1}{e^{\frac{V + 30}{10}} + 1}$$



Introduction

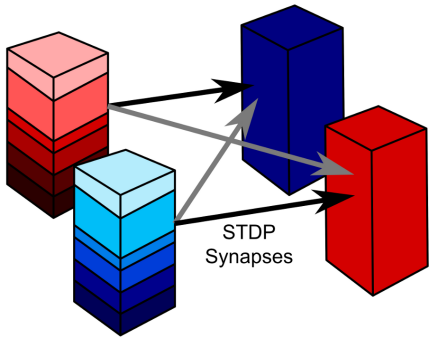
- Artificial Intelligence

Introduction

- Artificial Intelligence
- Embodied Modeling

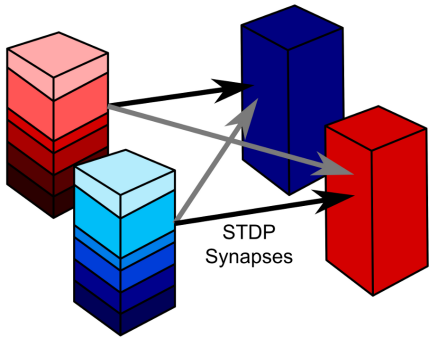
Introduction

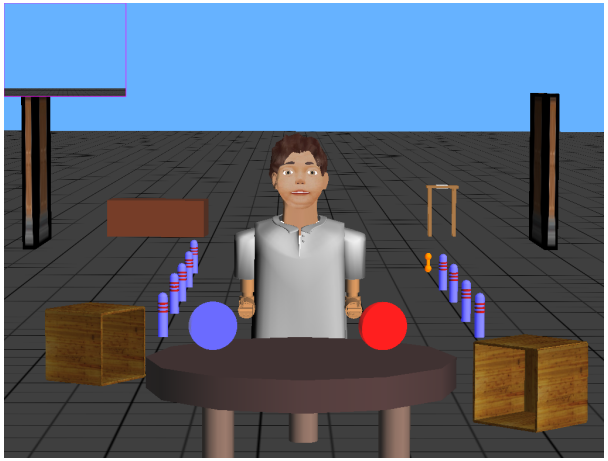
- Artificial Intelligence
- Embodied Modeling



Introduction

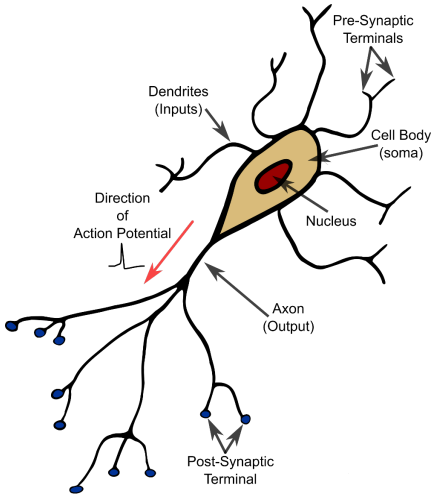
- Artificial Intelligence
- Embodied Modeling





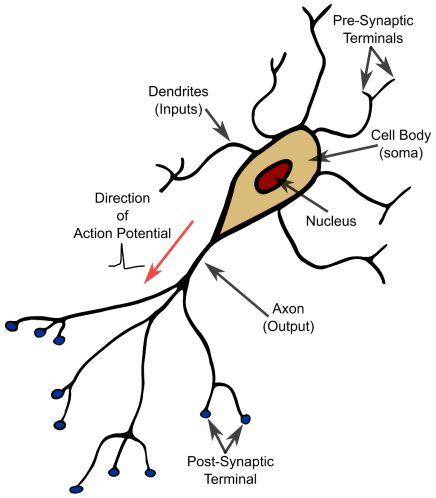
Introduction:
Spiking Neural Models

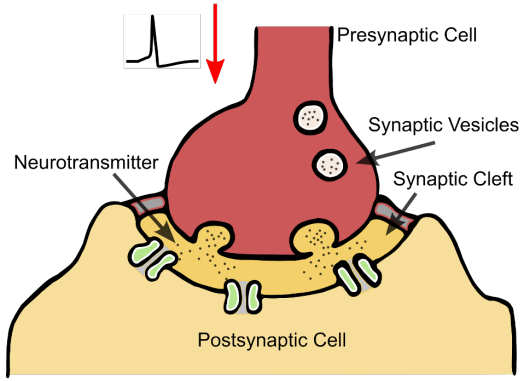
- Neurons are excitable cells that fire an action potential once a threshold voltage has been reached.



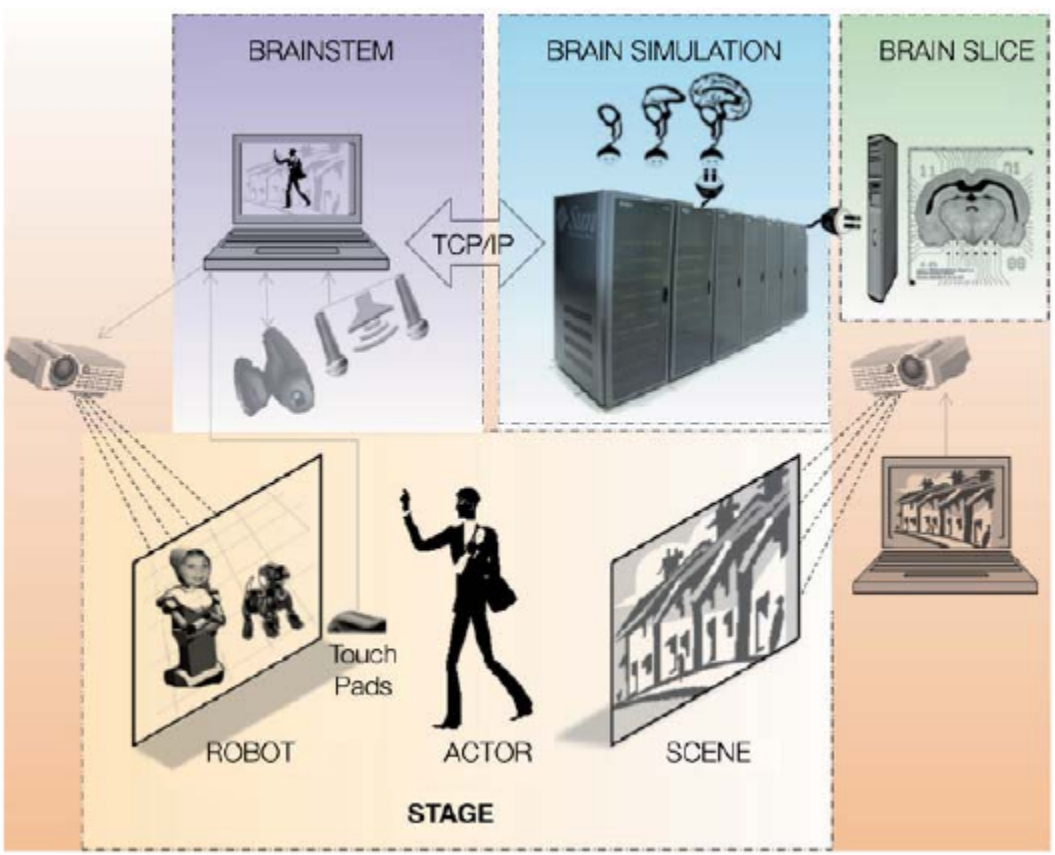
Introduction:
Spiking Neural Models

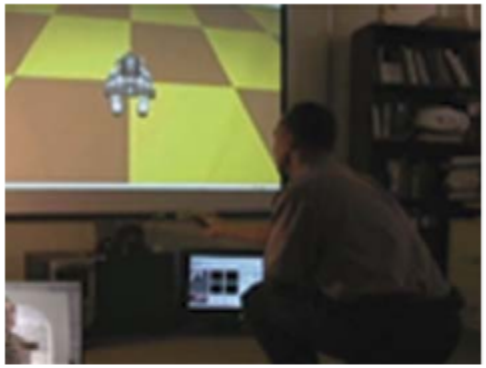
- Neurons are excitable cells that fire an action potential once a threshold voltage has been reached.
- This spike of electrical activity initiates the communication between neurons. Its effect is felt by all neurons connected to the one that spiked.

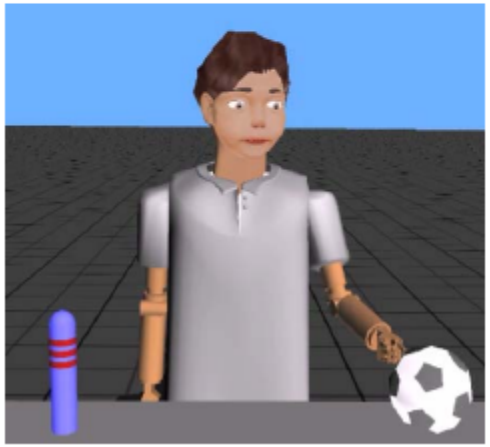




Introduction:
Neurorobotics and VNR







Introduction:
Neocortical Simulator

Features:

Introduction:
Neocortical Simulator

Features:

- High-performance MPI-based parallel architecture

Introduction:
Neocortical Simulator

Features:

- High-performance MPI-based parallel architecture
- Leaky Integrate-and-fire (LIF) neurons with conductance based synapses and Hodgkin-Huxley channels.

Introduction:
Neocortical Simulator

Features:

- High-performance MPI-based parallel architecture
- Leaky Integrate-and-fire (LIF) neurons with conductance based synapses and Hodgkin-Huxley channels.
- Hebbian synaptic learning with Short-term plasticity, augmentation, and spike-timing dependent plasticity (STDP).

Introduction:
Neocortical Simulator

Features:

- High-performance MPI-based parallel architecture
- Leaky Integrate-and-fire (LIF) neurons with conductance based synapses and Hodgkin-Huxley channels.
- Hebbian synaptic learning with Short-term plasticity, augmentation, and spike-timing dependent plasticity (STDP).
- Unique network mechanism for getting spiking information out and stimulus information in.

Outline

- Introduction
  - Spiking Neural Models
  - Neurorobotics and VNR
  - NCS
- NCSTools
  - Motivation
  - Design
- Examples
  - Reward-Based Learning
  - Trust the Intent Recognition
- Questions?

Outline

- NCSTools
  - Motivation
  - Design

NCSTools

Motivation

- Neurorobotics is a combination of very complex disciplines.

NCSTools

Motivation

- Neurorobotics is a combination of very complex disciplines.
- Previous tools required intimate knowledge of both neuroscience and engineering.

NCSTools

Motivation

- Neurorobotics is a combination of very complex disciplines.
- Previous tools required intimate knowledge of both neuroscience and engineering.
- Changes in either the agent or neural model required extensive modifications of both.

NCSTools

Motivation

- Neurorobotics is a combination of very complex disciplines.
- Previous tools required intimate knowledge of both neuroscience and engineering.
- Changes in either the agent or neural model required extensive modifications of both.
- Previous solutions lacked reusability.

NCSTools Design:

Software Engineering

Developed in C++ as a replacement for a package named Brainstem.

NCSTools Design:

Software Engineering

Developed in C++ as a replacement for a package named Brainstem.

Non-functional requirements

NCSTools Design:
Software Engineering

Developed in C++ as a replacement for a package named Brainstem.

Non-functional requirements

- Usability

NCSTools Design:

Software Engineering

Developed in C++ as a replacement for a package named Brainstem.

Non-functional requirements

- Usability
- Extensibility

NCSTools Design:

Software Engineering

Developed in C++ as a replacement for a package named Brainstem.

Non-functional requirements

- Usability
- Extensibility
- Robustness

NCSTools Design:

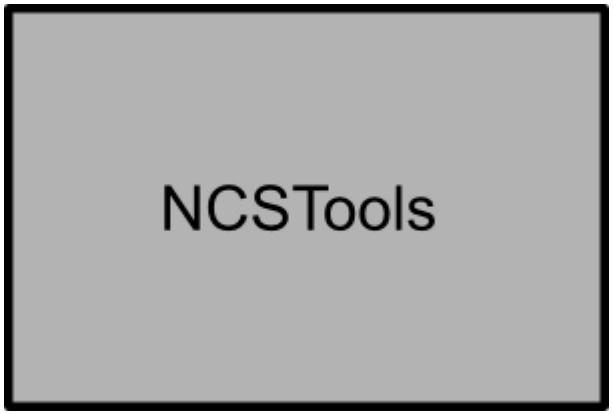
Software Engineering

Developed in C++ as a replacement for a package named Brainstem.

Non-functional requirements

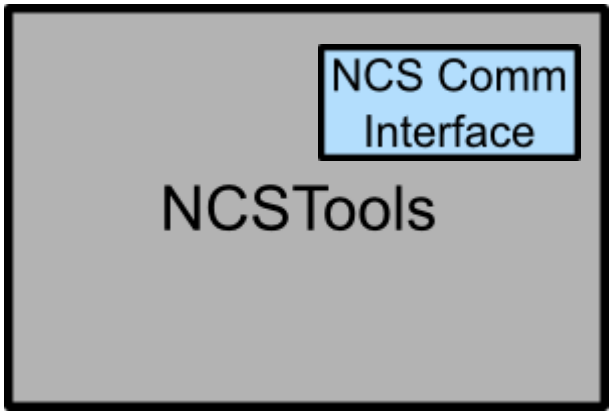
- Usability
- Extensibility
- Robustness
- Reliability

NCSTools Design:
Components



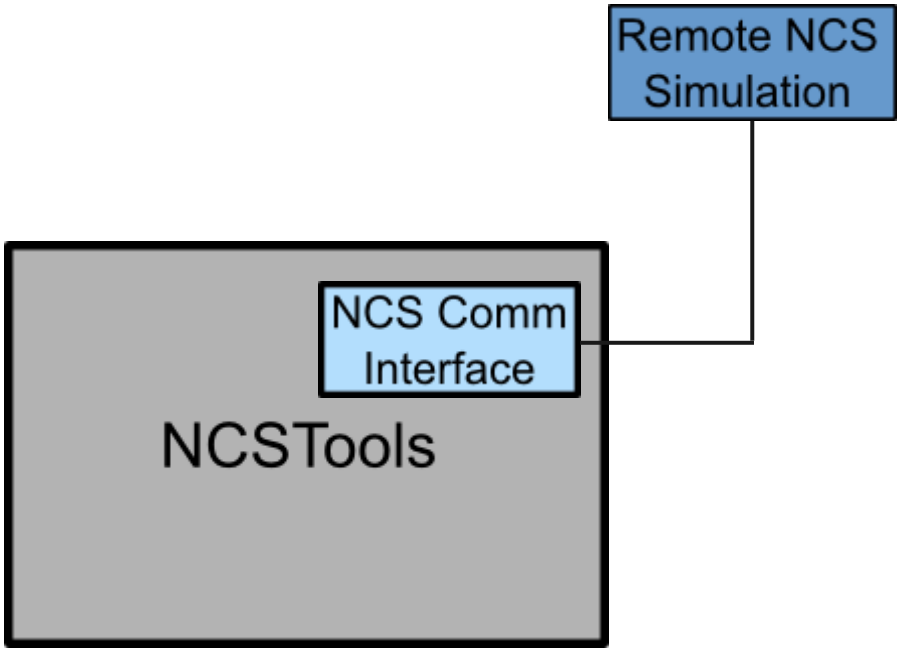
NCSTools Design:

Components



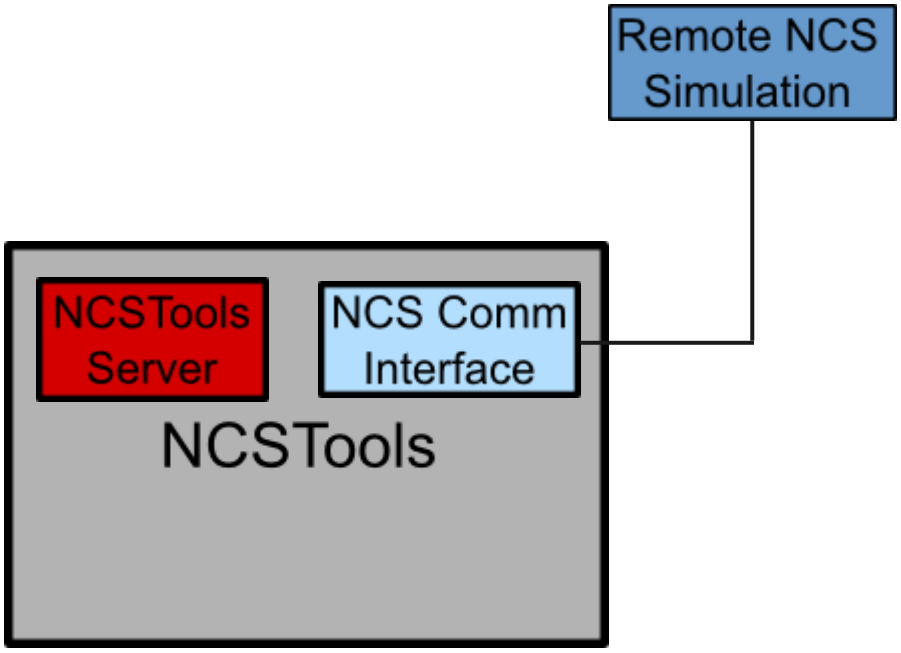
NCSTools Design:

Components



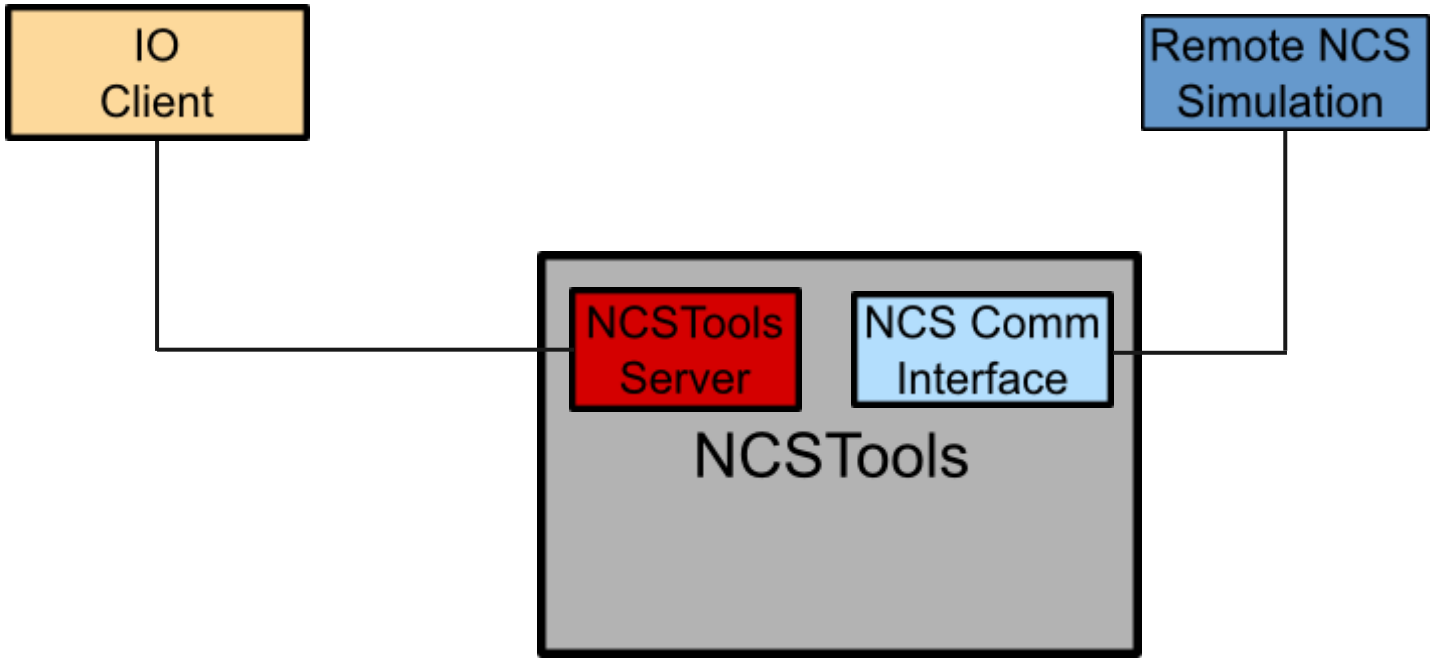
NCSTools Design:

Components



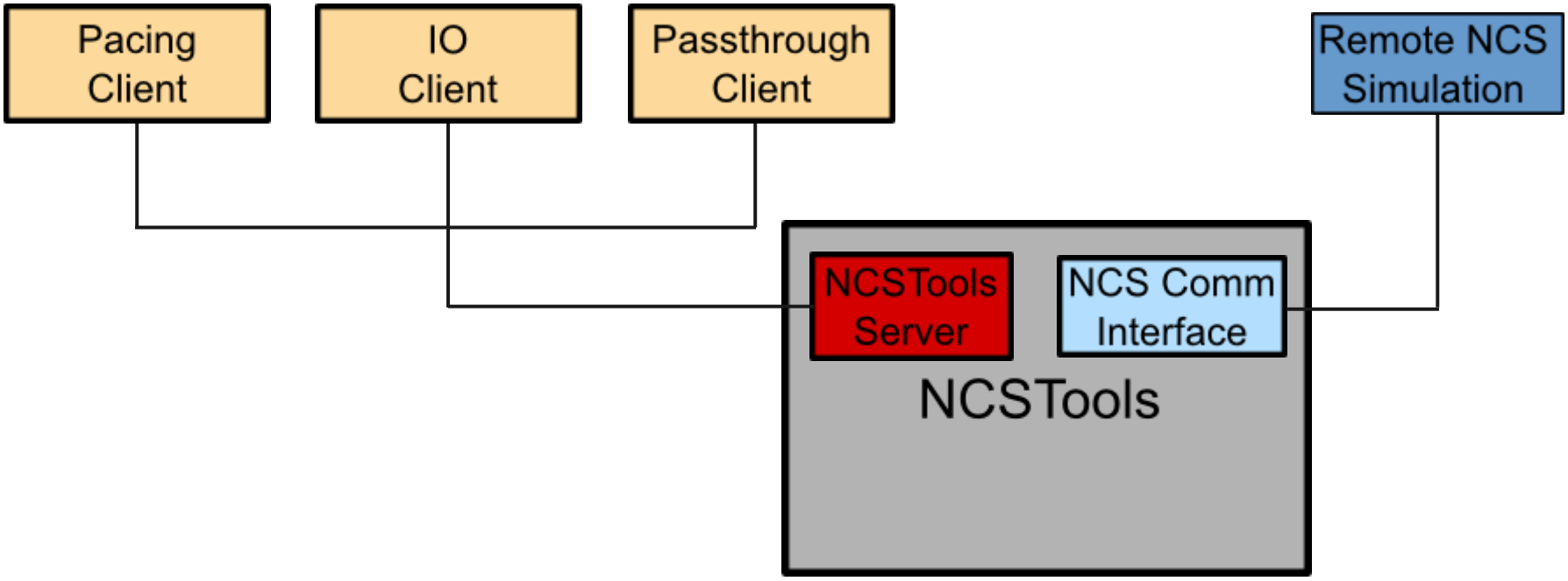
NCSTools Design:

Components



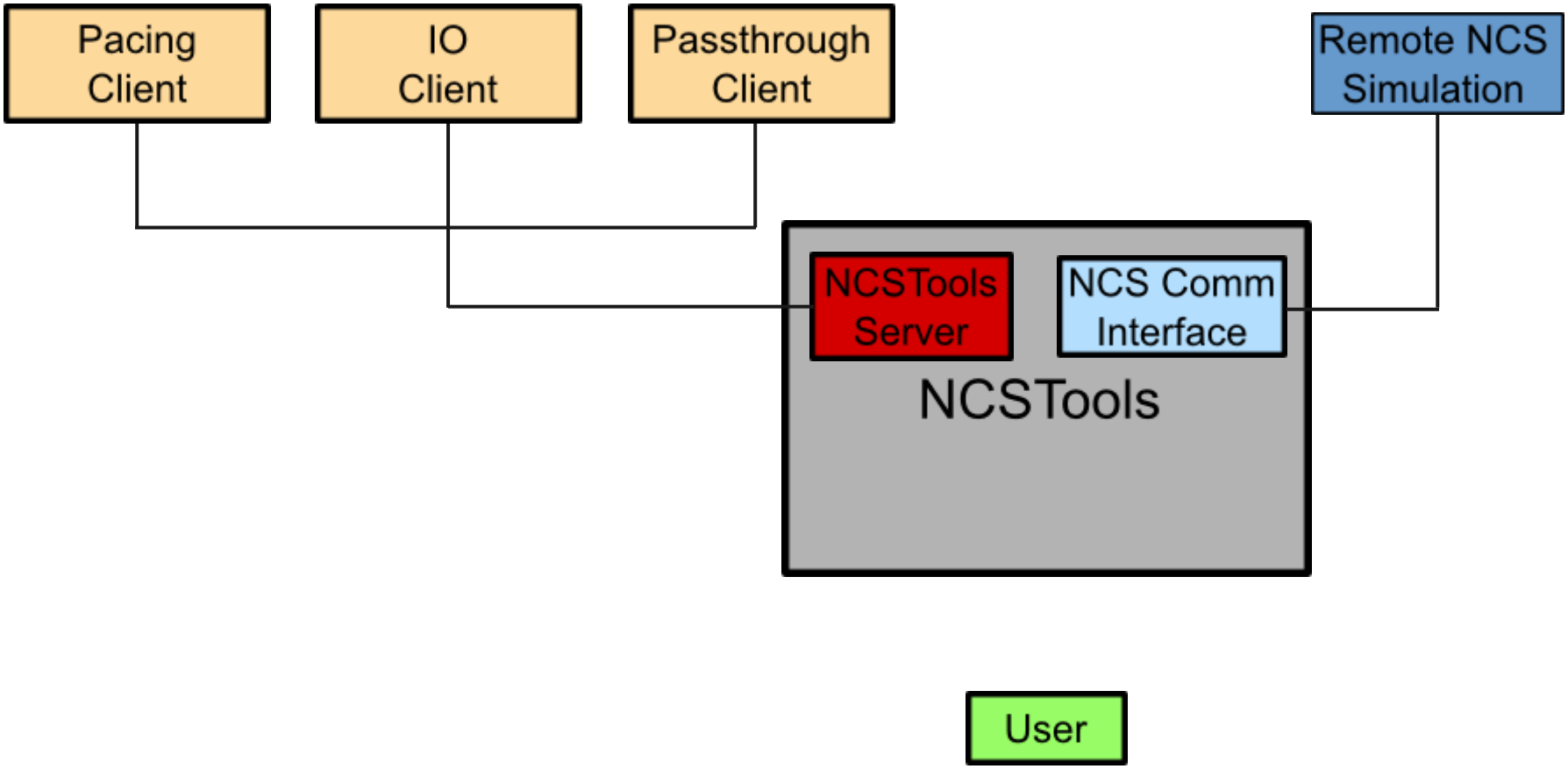
NCSTools Design:

Components



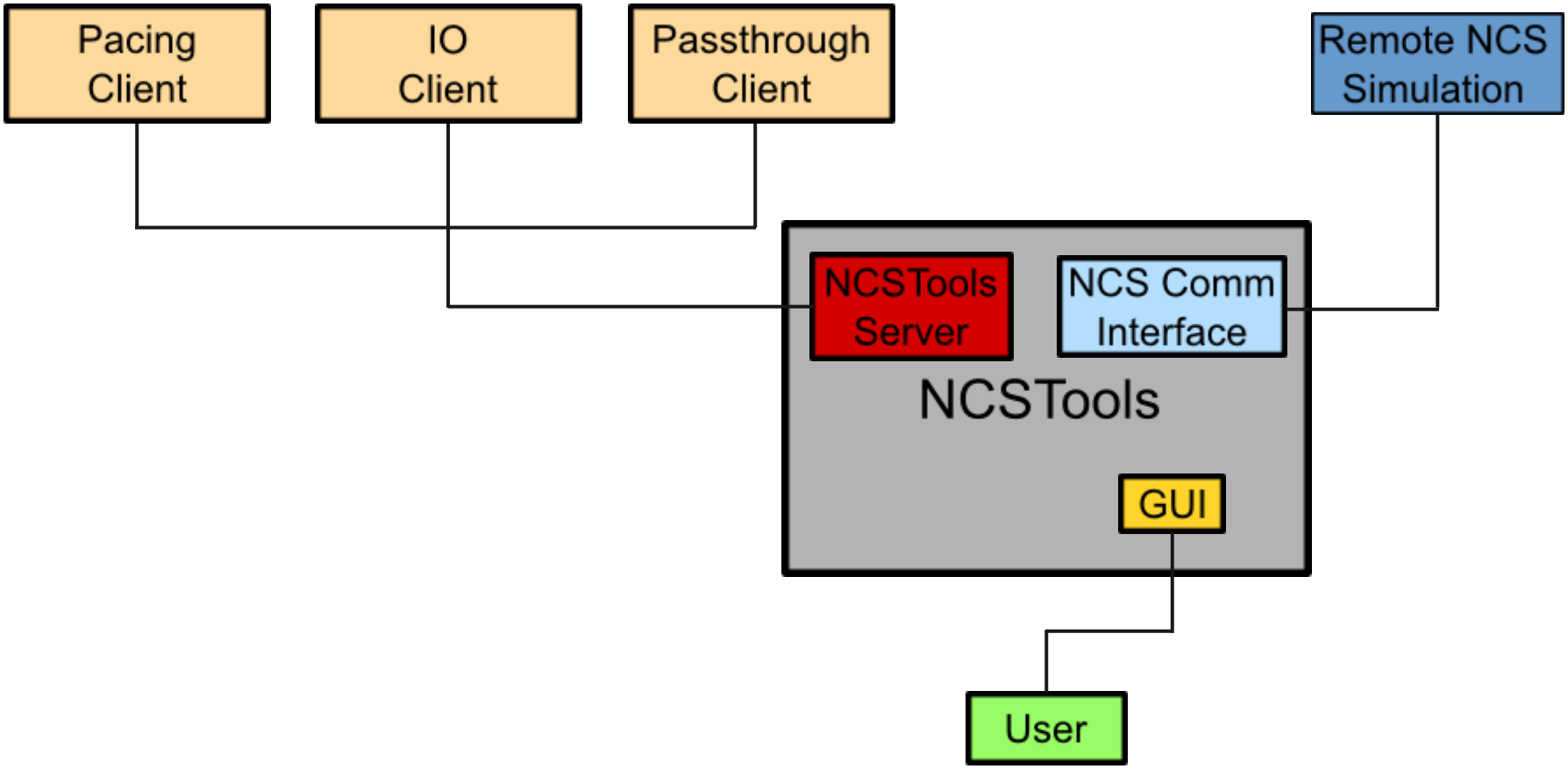
NCSTools Design:

Components



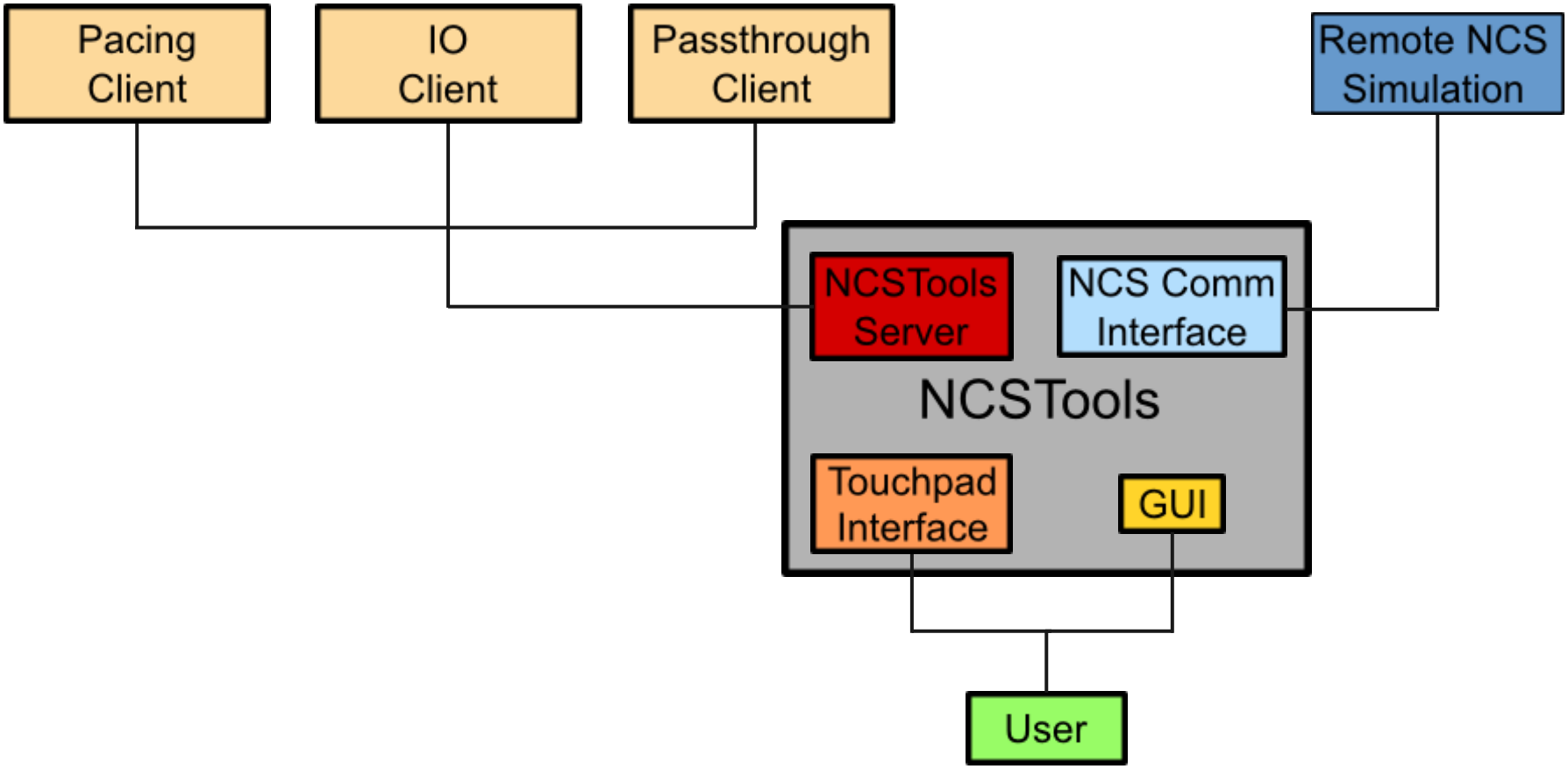
NCSTools Design:

Components



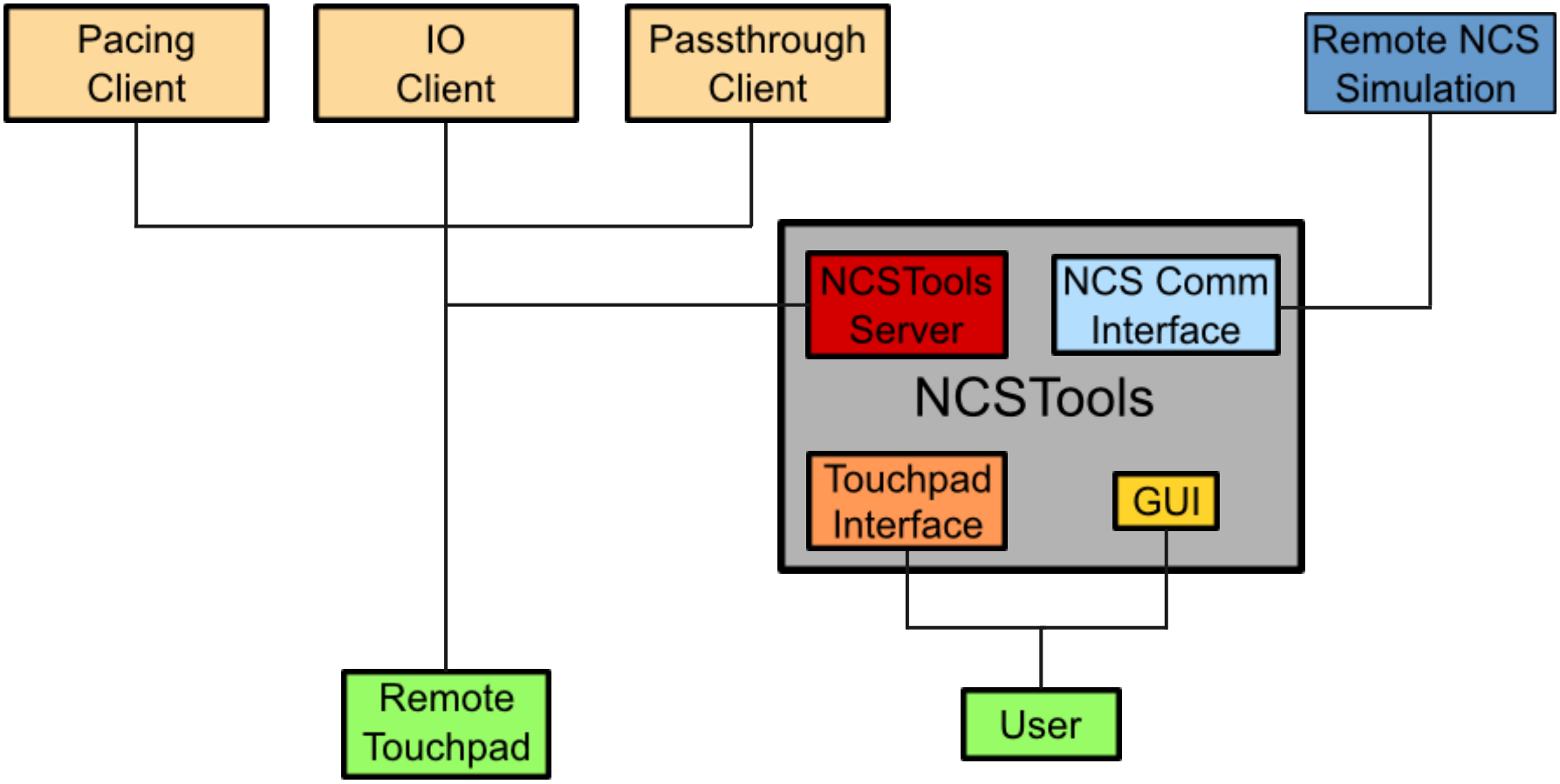
NCSTools Design:

Components



NCSTools Design:

Components



NCSTools Design:
Configuration Language

The flexibility of NCSTools lies in its configuration language.

NCSTools Design:
Configuration Language

The flexibility of NCSTools lies in its configuration language.

This includes:

- Communication "Language"

NCSTools Design:
Configuration Language

The flexibility of NCSTools lies in its configuration language.

This includes:

- Communication "Language"
- GUI

NCSTools Design:
Configuration Language

The flexibility of NCSTools lies in its configuration language.

This includes:

- Communication "Language"
- GUI
- User IO

NCSTools Design:
Configuration Language

The flexibility of NCSTools lies in its configuration language.

This includes:

- Communication "Language"
- GUI
- User IO
- Data Processing

NCSTools Design:

Configuration Language

```
input:
{
  NCSReportCollection1: {
    num_reports = 500;
    # The total period of counting, includes
    # report collection and recovery.
    period = 500;
    # The number of reports.
    NCSReports = 2;
    type = "STANDARD";
    NCSReport1: {
      connection = "to_PMC1";
      command = "point_left";
      };
    NCSReport2: {
      connection = "to_PMC2";
      command = "point_right";
    };
    # Setup the plots for this group.
    plot = "YES";
    plotType = "BAR";
    plotname = "Motor Activity";
    tabIndex = 1;
    plotIndex = 1;
  };
};
```

NCSTools Design:
Configuration Language

```
output:
{
  NCSStim1: {
    type = "TIMED_OUTPUT";
    command = "saw_red";
    connection = "from_VC1";
    # The static output to send.
    output = "0.2000";
    # How many ticks between stim inputs.
    frequency = 50;
    # The number of times to repeat the input.
    num_outputs = 10;;
  };
  NCSStim2: {
    type = "TIMED_OUTPUT";
    command = "saw_blue";
    connection = "from_childbot_VC1";
    output = "0.0000";
    frequency = 50;
    num_outputs = 10;;
  };
};
```

NCSTools Design:

Inputs

Inputs are signals coming from NCS.

NCSTools Design:

Inputs

Inputs are signals coming from NCS.

- Spike Counts

NCSTools Design:

Inputs

Inputs are signals coming from NCS.

- Spike Counts
- Voltages

NCSTools Design:

Inputs

Inputs are signals coming from NCS.

- Spike Counts
- Voltages
- Synaptic Weights

NCSTools Design:

Inputs

Inputs are signals coming from NCS.

- Spike Counts
- Voltages
- Synaptic Weights

Two Types of Inputs

NCSTools Design:

Inputs

Inputs are signals coming from NCS.

- Spike Counts
- Voltages
- Synaptic Weights

Two Types of Inputs

- Input Containers

NCSTools Design:
Inputs

Inputs are signals coming from NCS.

- Spike Counts
- Voltages
- Synaptic Weights

Two Types of Inputs

- Input Containers
- Windowed Input Containers

NCSTools Design:
Outputs

Outputs define the signals traveling to NCS.

NCSTools Design:
Outputs

Outputs define the signals traveling to NCS.

These can be:

- Static

NCSTools Design:
Outputs

Outputs define the signals traveling to NCS.

These can be:

- Static
- Dynamic

NCSTools Design:
Outputs

Outputs define the signals traveling to NCS.

These can be:

- Static
- Dynamic
- Timed

NCSTools Design:

Network Communication

Manages communication to and from NCS.

NCSTools Design:

Network Communication

Manages communication to and from NCS.

Provides a socket server for client communication.

NCSTools Design:

Network Communication

Manages communication to and from NCS.

Provides a socket server for client communication.

Client classes are provided in C/C++, python and MATLAB/Java.

NCSTools Design:

Network Communication

Manages communication to and from NCS.

Provides a socket server for client communication.

Client classes are provided in C/C++, python and MATLAB/Java.

There are three types of client connections:

NCSTools Design:

Network Communication

Manages communication to and from NCS.

Provides a socket server for client communication.

Client classes are provided in C/C++, python and MATLAB/Java.

There are three types of client connections:

- IO

NCSTools Design:

Network Communication

Manages communication to and from NCS.

Provides a socket server for client communication.

Client classes are provided in C/C++, python and MATLAB/Java.

There are three types of client connections:

- IO
- Pacing

NCSTools Design:

Network Communication

Manages communication to and from NCS.

Provides a socket server for client communication.

Client classes are provided in C/C++, python and MATLAB/Java.

There are three types of client connections:

- IO
- Pacing
- Passthrough

NCSTools Design:

User Interface

Configurable GUI for plotting simulation information.

NCSTools Design:

User Interface

Configurable GUI for plotting simulation information.

Written in Qt for cross-compatibility.

NCSTools Design:

User Interface

Configurable GUI for plotting simulation information.

Written in Qt for cross-compatibility.

Provides three plot types

NCSTools Design:

User Interface

Configurable GUI for plotting simulation information.

Written in Qt for cross-compatibility.

Provides three plot types

- Bar

NCSTools Design:

User Interface

Configurable GUI for plotting simulation information.

Written in Qt for cross-compatibility.

Provides three plot types

- Bar
- Line

NCSTools Design:

User Interface

Configurable GUI for plotting simulation information.

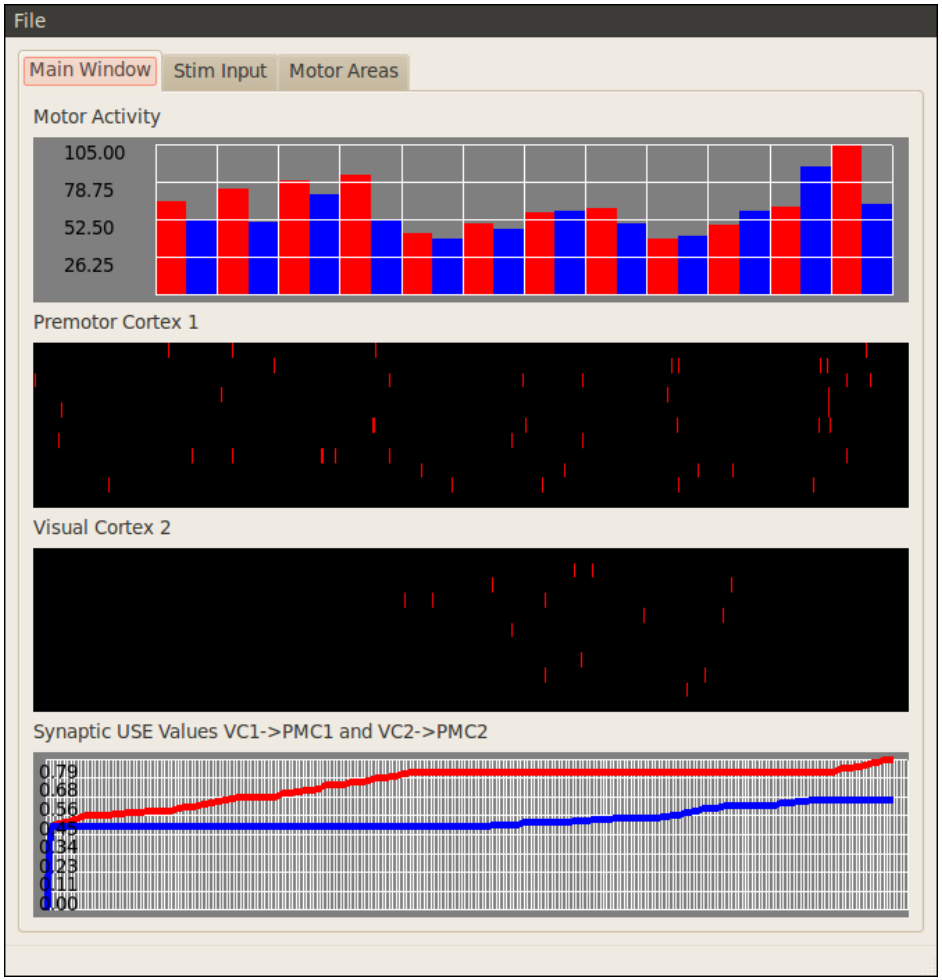
Written in Qt for cross-compatibility.

Provides three plot types

- Bar
- Line
- Raster

NCSTools Design:

User Interface



NCSTools Design:

User Interface

Command line (touchpad) interface.

NCSTools Design:

User Interface

Command line (touchpad) interface.

Users can bind single keys or new-line terminated strings to stimulus and control signals.

NCSTools Design:

User Interface

Command line (touchpad) interface.

Users can bind single keys or new-line terminated strings to stimulus and control signals.

Signals can be sent to NCS, its control interface or to connected agents.

NCSTools Design:

User Interface

Command line (touchpad) interface.

Users can bind single keys or new-line terminated strings to stimulus and control signals.

Signals can be sent to NCS, its control interface or to connected agents.

There are three types of touchpad connections:

NCSTools Design:

User Interface

Command line (touchpad) interface.

Users can bind single keys or new-line terminated strings to stimulus and control signals.

Signals can be sent to NCS, its control interface or to connected agents.

There are three types of touchpad connections:

- Instant

NCSTools Design:

User Interface

Command line (touchpad) interface.

Users can bind single keys or new-line terminated strings to stimulus and control signals.

Signals can be sent to NCS, its control interface or to connected agents.

There are three types of touchpad connections:

- Instant
- Repeated

NCSTools Design:

User Interface

Command line (touchpad) interface.

Users can bind single keys or new-line terminated strings to stimulus and control signals.

Signals can be sent to NCS, its control interface or to connected agents.

There are three types of touchpad connections:

- Instant
- Repeated
- Timed

Outline

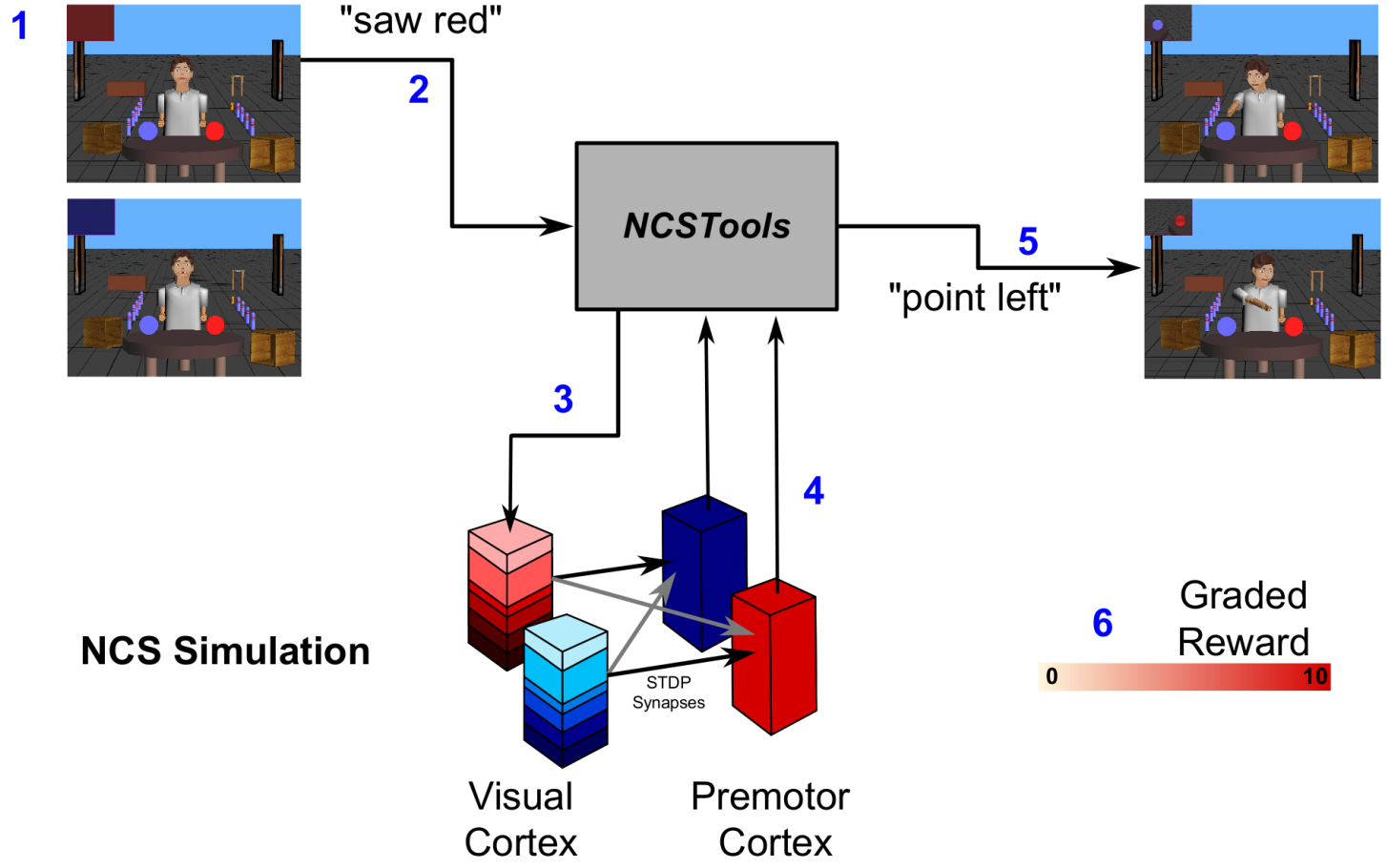
- Introduction
  - Spiking Neural Models
  - Neurorobotics and VNR
  - NCS
- NCSTools
  - Motivation
  - Design
- Examples
  - Reward-Based Learning
  - Trust the Intent Recognition
- Questions?

Outline

- Examples
  - Reward-Based Learning
  - Trust the Intent Recognition

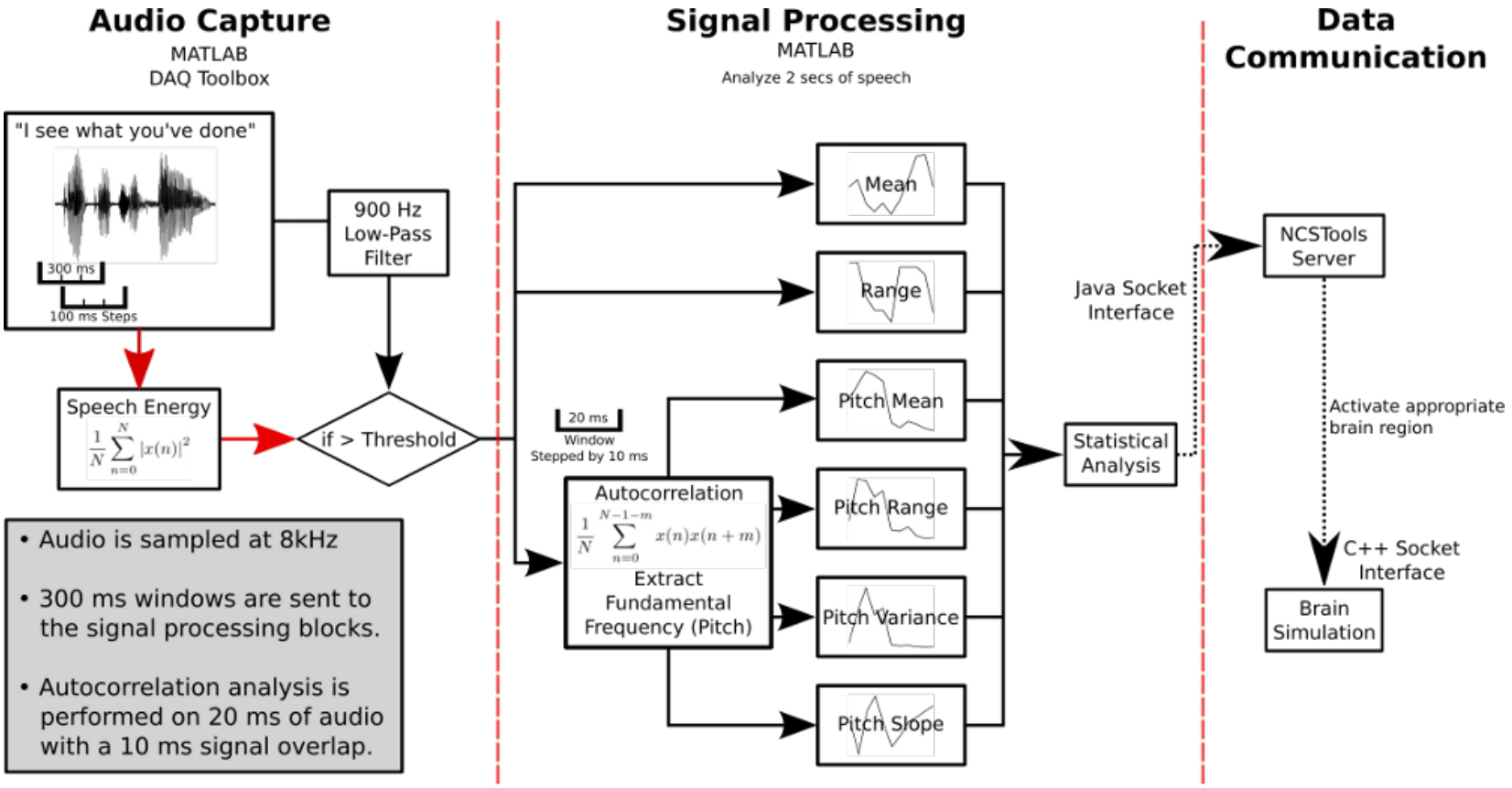
Example

Reward-Based Learning



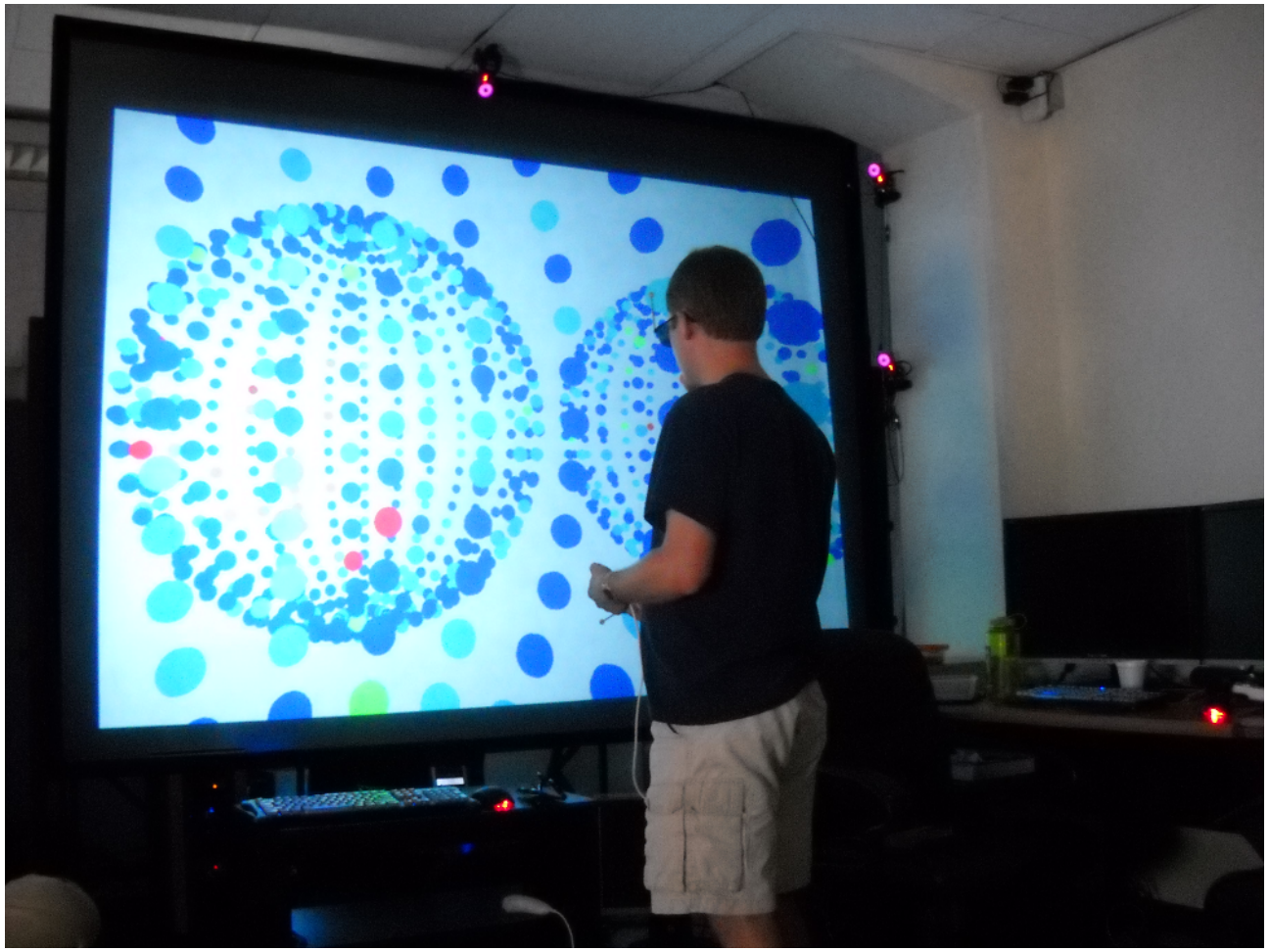
Example

Emotional Speech Processing



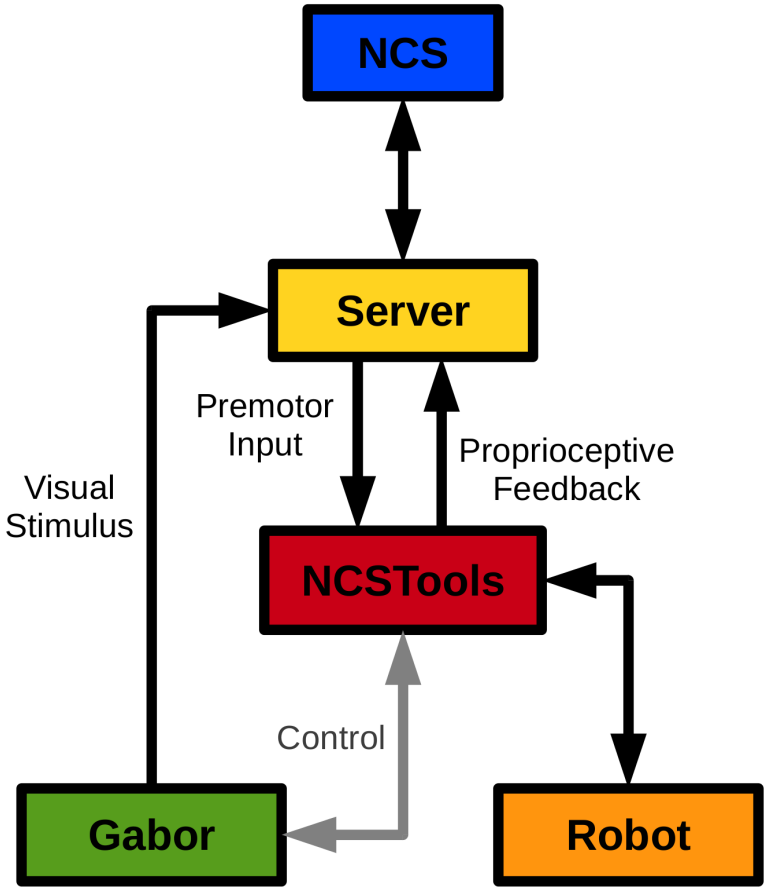
Example

Virtual Reality



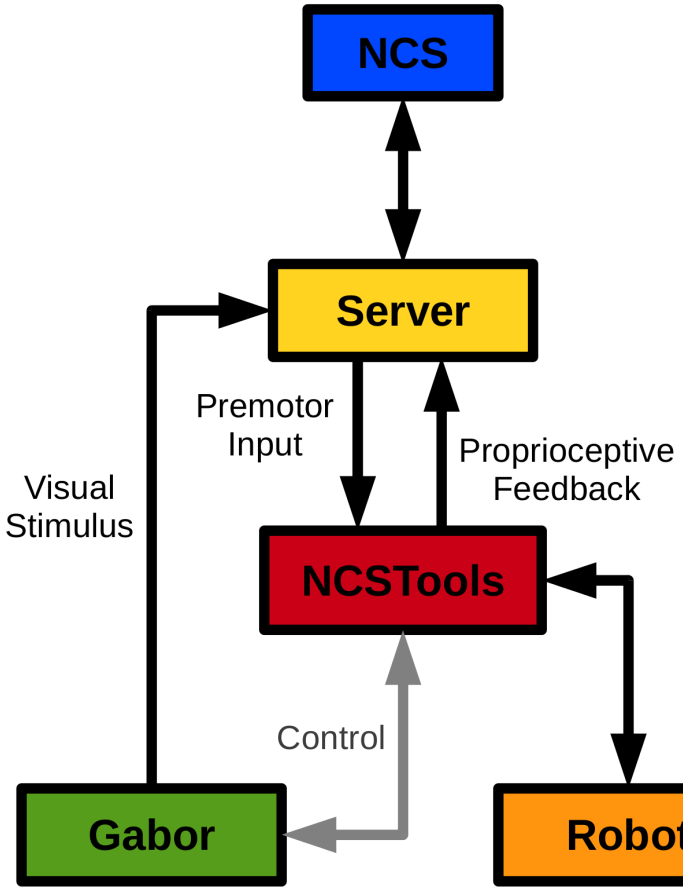
Example

Trust the Intent

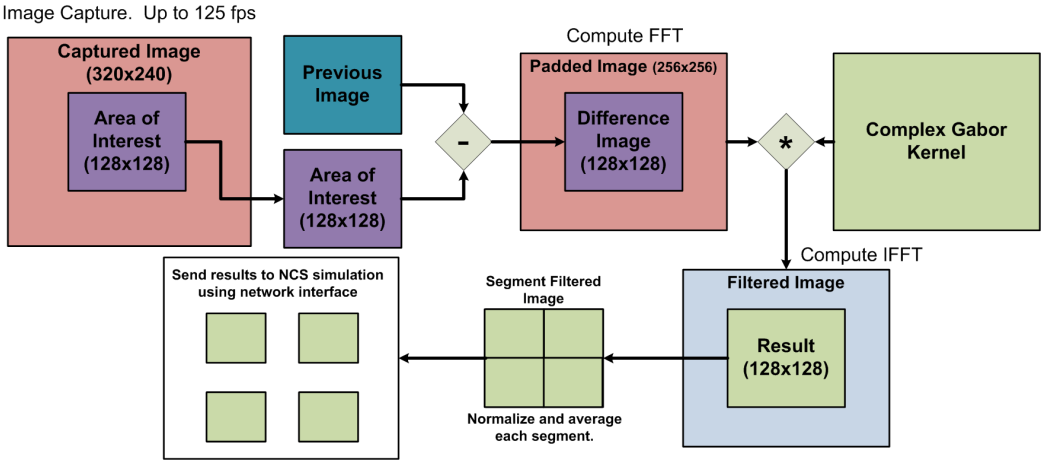


Example

Trust the Intent

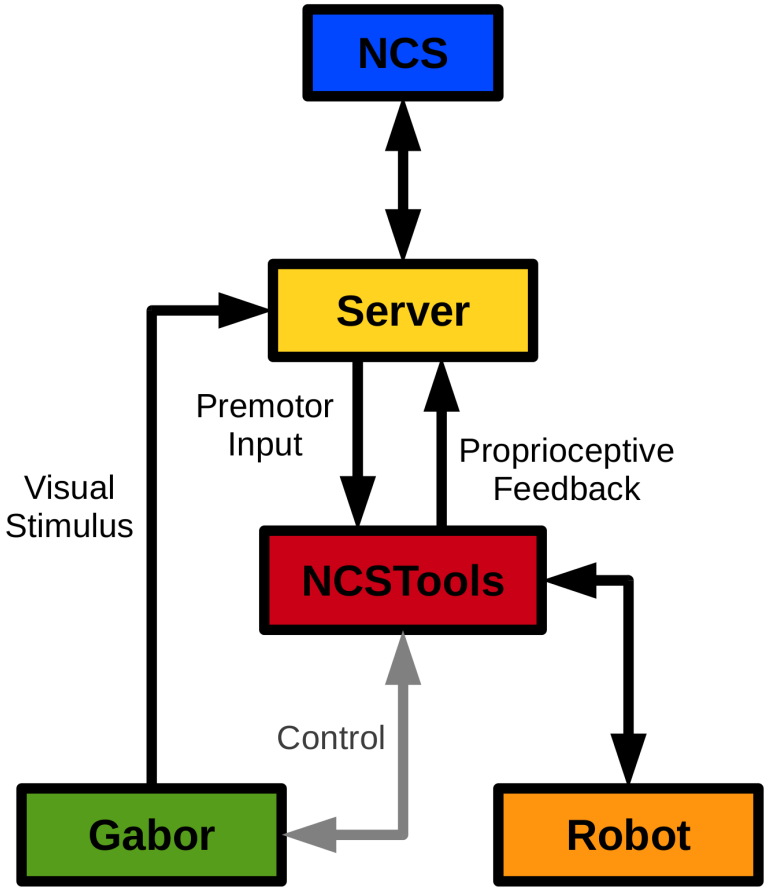


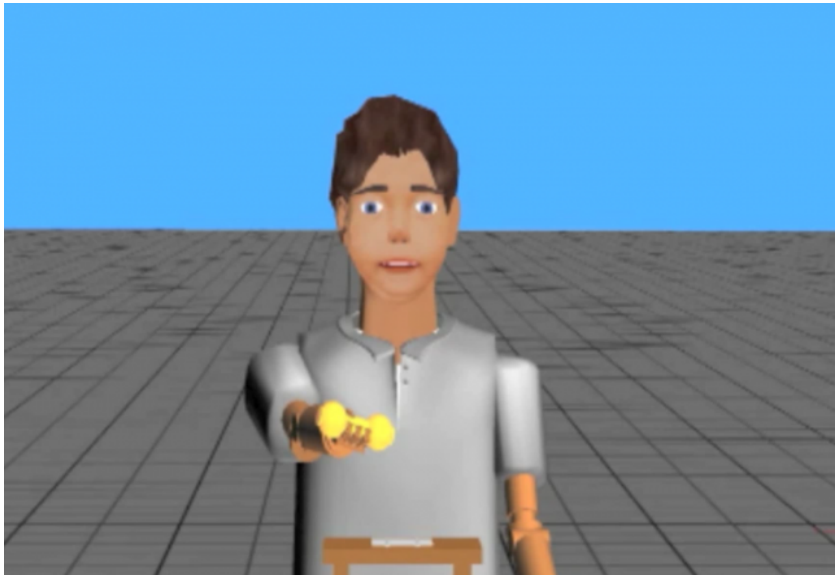
User is Captured by Gabor Filter Program:



Example

Trust the Intent

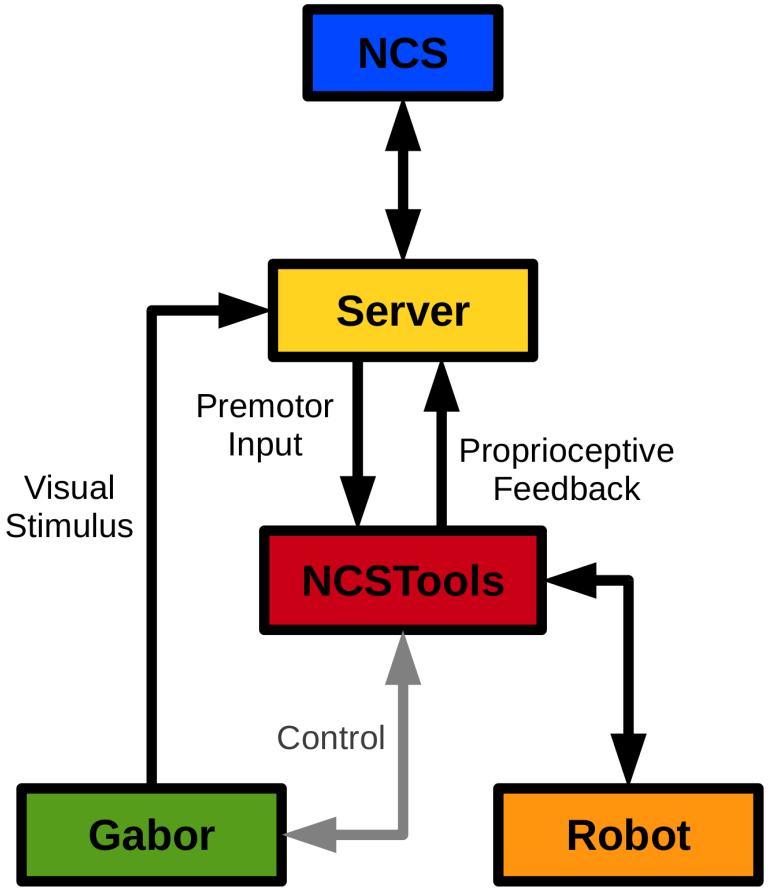


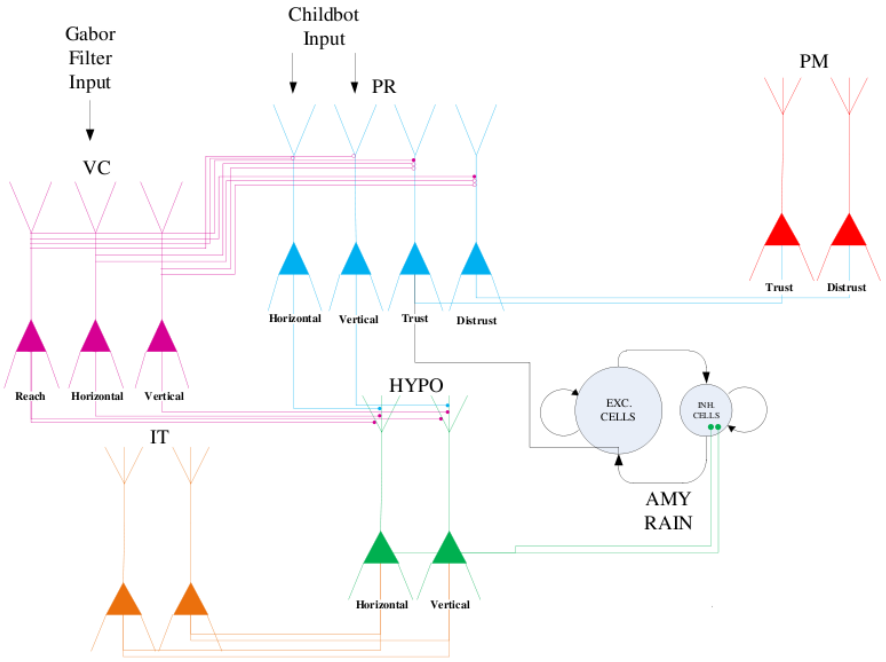


Robotic avatar sends proprioceptive information to NCSTools and receives movements back.

Example

Trust the Intent

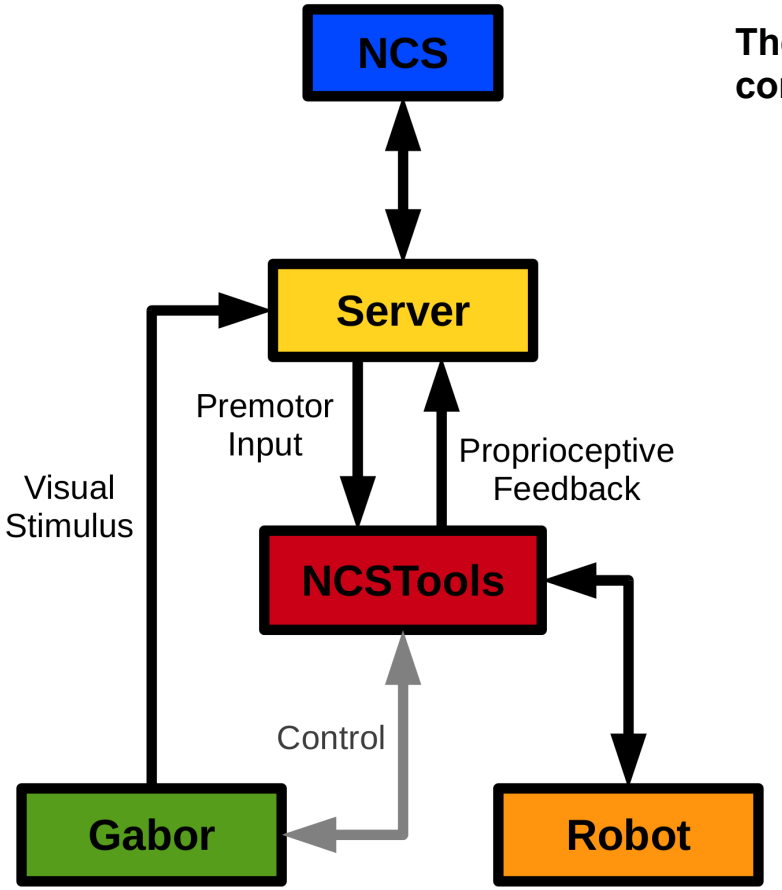




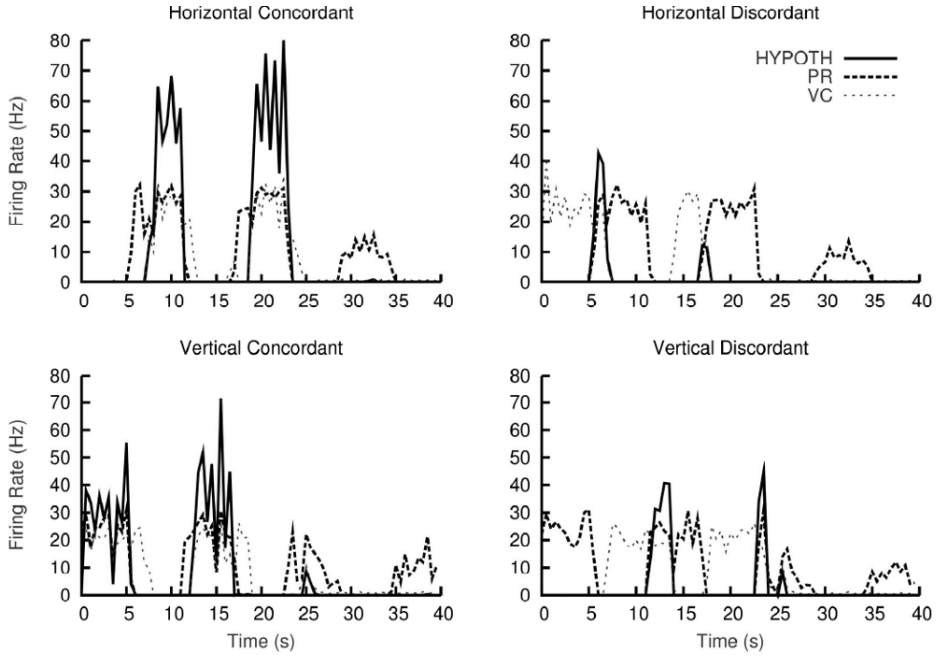
NCSTools sends stimulus information to NCS and received back neural activity.

Example

Trust the Intent



The information is processed based on the user configuration.



Example

Trust the Intent