A Compositional Sheaf-Theoretic Framework for Event-Based Systems

Gioele Zardini*, David I. Spivak†, Andrea Censi*, and Emilio Frazzoli*

*Institute for Dynamic Systems and Control (IDSC)
Department of Mechanical and Process Engineering, ETH Zürich

†Department of Mathematics
Massachusetts Institute of Technology

7th July, 2020

Engineering complex systems causes pain

	actuation	perception	hardware
	sensing	planning	localization
robot design =	control	learning	mapping
	energetics	interactions	coordination
	computation	software	calibration

Engineering complex systems causes pain

$$\text{robot design} = \begin{matrix} \text{actuation} & \text{perception} & \text{hardware} \\ \text{sensing} & \text{planning} & \text{localization} \\ \text{control} & \text{learning} & \text{mapping} \\ \text{energetics} & \text{interactions} & \text{coordination} \\ \text{computation} & \text{software} & \text{calibration} \end{matrix}$$

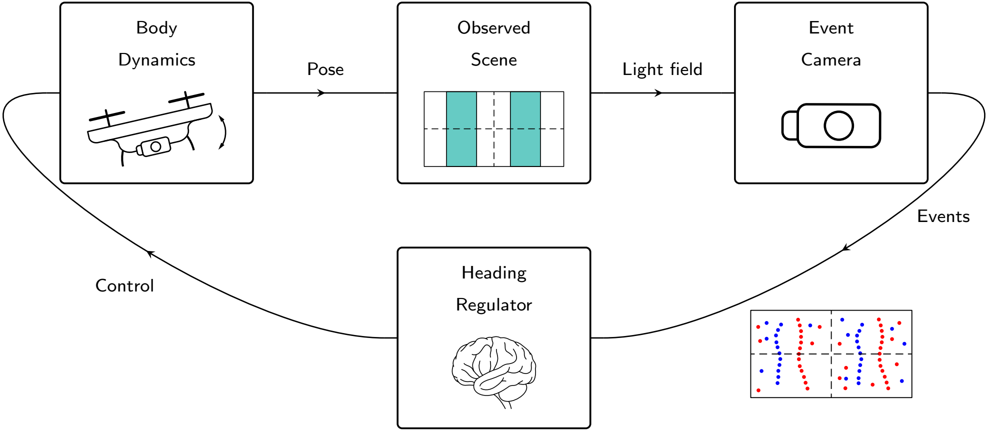
Engineering malaise:

- Too many components, too different (continuous, discrete, event-based).
- Each component is modeled in a specific way: Engineers like to partition.
- Even if modeled differently, the components have to be co-designed at some point: Pain.

An example: Event-Based Sensors

`https://www.youtube.com/watch?v=kPCZESVfHoQ`

How can we put together continuous time, event-based, discrete time?



Towards a solution: A common framework

$$\text{robot design} = \begin{matrix} \text{actuation} & \text{perception} & \text{hardware} \\ \text{sensing} & \text{planning} & \text{localization} \\ \text{control} & \text{learning} & \text{mapping} \\ \text{energetics} & \text{interactions} & \text{coordination} \\ \text{computation} & \text{software} & \text{calibration} \end{matrix}$$

- Components are different, but there is an abstraction level at which they are the same.
- These are *behavior types*, which can be *composed* in different ways.

Towards a solution: A common framework

$$\text{robot design} = \begin{matrix} \text{actuation} & \text{perception} & \text{hardware} \\ \text{sensing} & \text{planning} & \text{localization} \\ \text{control} & \text{learning} & \text{mapping} \\ \text{energetics} & \text{interactions} & \text{coordination} \\ \text{computation} & \text{software} & \text{calibration} \end{matrix}$$

- Components are different, but there is an abstraction level at which they are the same.
- These are *behavior types*, which can be *composed* in different ways.

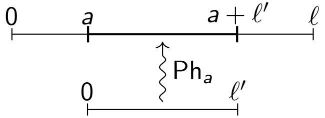
Some references:

- Schultz, Spivak, Vasilakopoulou, *Dynamical systems and sheaves*, '20.
- Schultz, Spivak, *Temporal Type Theory: A topos-theoretic approach to systems and behavior*, '19.

Describing system behaviors: The category of continuous intervals

Given $a \in \mathbb{R}_{\geq 0}$, the translation-by-$a$ function is

$$\begin{aligned} \mathrm{Ph}_a \colon \mathbb{R}_{\geq 0} &\to \mathbb{R}_{\geq 0} \\ \ell &\mapsto a + \ell \end{aligned}$$



Definition (Category of continuous intervals Int)

- Objects: *Durations* $\mathrm{Ob}(\mathrm{Int}) := \{\ell \in \mathbb{R}_{\geq 0}\}$.
- Morphisms: Given durations ℓ', ℓ, one has $\mathrm{Int}(\ell', \ell) := \{\mathrm{Ph}_a \mid a \in \mathbb{R}_{\geq 0} \text{ and } a + \ell' \leq \ell\}$.
- Identity morphism: $\mathrm{id}_\ell := \mathrm{Ph}_0 \in \mathrm{Int}(\ell, \ell)$.
- Composition of morphisms: Given $\mathrm{Ph}_a \colon \ell \to \ell'$, $\mathrm{Ph}_b \colon \ell' \to \ell''$, one has

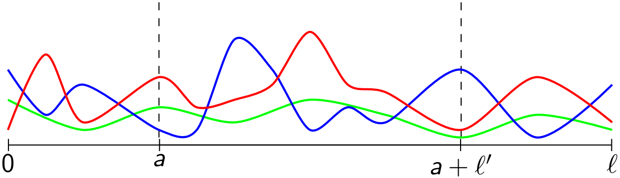
$$\mathrm{Ph}_a \fatsemi \mathrm{Ph}_b = \mathrm{Ph}_{a+b} \in \mathrm{Int}(\ell, \ell'').$$

Describing system behaviors: Int-presehaves and sections

Definition (Int-presheaves)

An Int-*presheaf* A is a functor $A\colon \mathrm{Int}^{\mathrm{op}} \to \mathrm{Set}$.

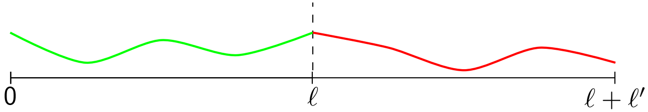
- $x \in A(\ell)$ is a *length-ℓ section (behavior)*.
- Given $x \in A(\ell)$, $\mathrm{Ph}_a\colon \ell' \to \ell$, the *restriction* is $x|_{[a,a+\ell']} = A(\mathrm{Ph}_a)(x) \in A(\ell')$.



Describing system behaviors: Compatible sections and Int-sheaves

Definition (Compatible sections)

Given a presheaf A, the sections $a \in A(\ell)$ and $a' \in A(\ell')$ are *compatible* if $a|_{[\ell,\ell]} = a'|_{[0,0]}$.



Describing system behaviors: Compatible sections and Int-sheaves

Definition (Compatible sections)

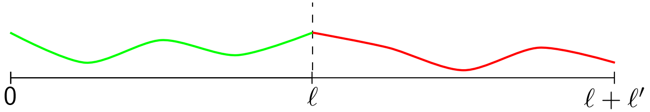
Given a presheaf A, the sections $a \in A(\ell)$ and $a' \in A(\ell')$ are *compatible* if $a|_{[\ell,\ell]} = a'|_{[0,0]}$.

Definition (Int-sheaf)

An Int-presheaf $A\colon \mathsf{Int}^{\mathrm{op}} \to \mathsf{Set}$ is an Int-*sheaf* if, for all ℓ, ℓ' and compatible sections $a \in A(\ell)$, $a' \in A(\ell')$, there exists a unique $\bar{a} \in A(\ell + \ell')$ such that

$$\bar{a}|_{[0,\ell]} = a \quad \text{and} \quad \bar{a}|_{[\ell,\ell+\ell']} = a'.$$

We denote by $\widetilde{\mathsf{Int}}$ the category of Int-sheaves.



Describing different components: Machines

Definition (Machine)

Let $A, B \in \widetilde{\mathrm{Int}}$. An (A, B) *machine* is a span in $\widetilde{\mathrm{Int}}$ of the form:

$$\begin{tikzcd} & C \arrow[dl, "f^{\mathrm{in}}"'] \arrow[dr, "f^{\mathrm{out}}"] & \\ A & & B \end{tikzcd}$$

Describing different components: Machines

Definition (Machine)

Let $A, B \in \widetilde{\mathrm{Int}}$. An (A, B) *machine* is a span in $\widetilde{\mathrm{Int}}$ of the form:

$$\begin{tikzcd} & C \arrow[dl, "f^{\mathrm{in}}"'] \arrow[dr, "f^{\mathrm{out}}"] & \\ A & & B \end{tikzcd}$$

Example (Continuous Dynamical System)

Consider A, B euclidean spaces. An (A, B)-continuous dynamical system consists of

- State space: S.
- Dynamics: $\dot{s} = f^{\mathrm{dyn}}(s, a)$, for $a \in A$, $s \in S$, and smooth f^{dyn}.
- Readout: $b = f^{\mathrm{rdt}}(s)$, $b \in B$ and smooth f^{rdt}.

Describing different components: Machines

Definition (Machine)

Let $A, B \in \widetilde{\mathrm{Int}}$. An (A, B) *machine* is a span in $\widetilde{\mathrm{Int}}$ of the form:

$$\begin{tikzcd} & C \arrow[dl, "f^{\mathrm{in}}"'] \arrow[dr, "f^{\mathrm{out}}"] & \\ A & & B \end{tikzcd}$$

Example (Continuous Dynamical System)

Consider A, B euclidean spaces. An (A, B)-continuous dynamical system consists of

- State space: S.
- Dynamics: $\dot{s} = f^{\mathrm{dyn}}(s, a)$, for $a \in A$, $s \in S$, and smooth f^{dyn}.
- Readout: $b = f^{\mathrm{rdt}}(s)$, $b \in B$ and smooth f^{rdt}.

$$\begin{tikzcd} & D \arrow[dl, "f^{\mathrm{in}}"'] \arrow[dr, "f^{\mathrm{out}}"] & \\ C^1(A) & & C^1(B) \end{tikzcd}$$

$$D(\ell) = \{(a, s, b) \in C^1(A) \times S \times C^1(B) \mid \dot{s} = f^{\mathrm{dyn}}(a, s),\ b = f^{\mathrm{rdt}}(s)\}$$

Composing components via machines composition

Definition (Composition of machines)

Given two machines $M_1 = (D_1, f_1^{\text{in}}, f_1^{\text{out}})$ and $M_2 = (D_2, f_2^{\text{in}}, f_2^{\text{out}})$ of types (A, B) and (B, C) respectively, their *composite* is the machine $M = (D_1 \times_B D_2, f^{\text{in}}, f^{\text{out}})$ of type (A, C), namely the span given by pullback:

$$
\begin{tikzcd}
 & & D_1 \times_B D_2 \arrow[dl] \arrow[dr] \arrow[dd, phantom, "\vee" very near start] & & \\
 & D_1 \arrow[dl, "f_1^{\text{in}}"'] \arrow[dr, "f_1^{\text{out}}"] & & D_2 \arrow[dl, "f_2^{\text{in}}"'] \arrow[dr, "f_2^{\text{out}}"] & \\
A & & B & & C
\end{tikzcd}
$$

A closer look to a particular behavior type: Event streams

Definition (Event stream)

Given a set A and $\ell \geq 0$. A length-ℓ *event stream* of type A is an element of

$$\mathrm{Ev}_A(\ell) := \{(S, a) \mid S \subseteq \tilde{\ell}, S \text{ finite}, a\colon S \to A\}.$$

Example (Swiss traffic light)

$(S, a) \in \mathrm{Ev}_A(60)$, $S = \{20, 25, 45, 50\}$

$$a\colon S \to A = \{\mathrm{redToOrange}, \mathrm{orangeToGreen}, \mathrm{greenToOrange}, \mathrm{orangeToRed}\}$$

$$s \mapsto \begin{cases} \mathrm{redToOrange}, & \text{if } s = 20, \\ \mathrm{orangeToGreen}, & \text{if } s = 25, \\ \mathrm{greenToOrange}, & \text{if } s = 45, \\ \mathrm{orangeToRed}, & \text{if } s = 50. \end{cases}$$

Facts about event streams

Proposition (Ev is functorial)

Ev is functorial: Given a map $f\colon A \to B$, there is an induced morphism $Ev_f\colon Ev_A \to Ev_B$ in $\widetilde{\mathsf{Int}}$, preserving identities and composition.

Proposition (Ev_A is a sheaf)

For any set A, Ev_A is an Int*-sheaf.*

Proposition (Ev is a strong monoidal functor)

$Ev\colon (\mathsf{Set}, \odot, \varnothing) \to (\widetilde{\mathsf{Int}}, \times, 1)$ *is a strong monoidal functor, i.e.*

$$1 \cong Ev_{\varnothing} \text{ and } Ev_A \times Ev_B \cong Ev_{A \odot B},$$

where $A \odot B := A + B + A \times B$.

We are ready to define event-based systems: A particular type of machines

Definition (Event-based system)

Let A, B be sets. An *event-based system* $P = (C, f^{\text{in}}, f^{\text{out}})$ of type (A, B) is a machine

$$\begin{tikzcd} & C \arrow[dl, "f^{\text{in}}"'] \arrow[dr, "f^{\text{out}}"] & \\ \mathrm{Ev}_A & & \mathrm{Ev}_B \end{tikzcd}$$

between two event streams. Graphically:

$$\mathrm{Ev}_A - \boxed{\;P\;} - \mathrm{Ev}_B$$

Discrete dynamical systems as an example of event-based systems

Example (Discrete Dynamical Systems (DDS))

Let A, B be sets. An (A, B)-DDS consists of:

- State space S.
- Update function: $f^{\text{upd}} \colon A \times S \to S$.
- Readout function $f^{\text{rdt}} \colon S \to B$.

Discrete dynamical systems as an example of event-based systems

Example (Discrete Dynamical Systems (DDS))

Let A, B be sets. An (A, B)-DDS consists of:

- State space S.
- Update function: $f^{\text{upd}} \colon A \times S \to S$.
- Readout function $f^{\text{rdt}} \colon S \to B$.

This is an (A, B)-event-based system with

$$D(\ell) := \{T \subseteq \tilde{\ell}, (a, s) \colon T \to A \times S \mid T \text{ finite and } s_{i+1} = f^{\text{upd}}(a_i, s_i) \text{ for all } 1 \leq i \leq n-1\}.$$

Given $(T, a, s) \in D(\ell)$:

$$f^{\text{in}}(T, a, s) = (T, a),$$
$$f^{\text{out}}(T, a, s) = (T, (s \mathbin{⨾} f^{\text{rdt}})).$$

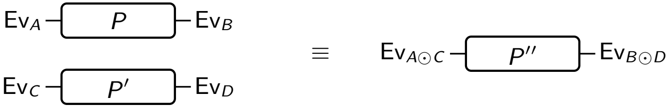
Tensor product of event-based systems

Definition (Tensor product of event-based systems)

Given two event-based systems $P(E, f^{\text{in}}, f^{\text{out}})$, $P'(E', f^{\text{in}\prime}, f^{\text{out}\prime})$, of types (A, B) and (A', B'), their tensor product is of type $(A \odot A', B \odot B')$ and is given by:

$$\begin{tikzcd} & E \times E' \arrow[dl, "f^{\text{in}} \times f^{\text{in}\prime}"'] \arrow[dr, "f^{\text{out}} \times f^{\text{out}\prime}"] & \\ \mathrm{Ev}_{A \odot A'} & & \mathrm{Ev}_{B \odot B'} \end{tikzcd}$$

Graphically:



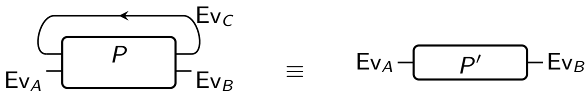
Trace of event-based systems

Definition (Trace of event-based systems)

Given an event based system $P(D, f^{\text{in}}, f^{\text{out}})$ of type $(A \times C, B \times C)$, its trace is an event-based system $P(E, f^{\text{in}}_{\text{tr}}, f^{\text{out}}_{\text{tr}})$ of type (A, B), with

$$
\begin{aligned}
E(\ell) &= \{d \in D(\ell) \mid \pi_2(f^{\text{in}}(d)) = \pi_2(f^{\text{out}}(d))\} \\
f^{\text{in}}_{\text{tr}}(d) &= \pi_1(f^{\text{in}}(d)) \in \text{Ev}_A \\
f^{\text{out}}_{\text{tr}}(d) &= \pi_1(f^{\text{out}}(d)) \in \text{Ev}_B.
\end{aligned}
$$

Graphically:



Yes, but what if I have to model continuous signals? Continuous streams

Definition (Continuous stream)

Given a topological space A, we can define the sheaf of *continuous streams* of type A as

$$\mathrm{Cnt}_A(\ell) := \{a \mid a\colon \tilde{\ell} \to A \text{ continuous}\}.$$

Lipschitz continuous streams are given by the sheaf:

$$\mathrm{LCnt}_A(\ell) = \{a\colon \tilde{\ell} \to A \mid a \text{ Lipschitz continuous}\} \subseteq \mathrm{Cnt}_A(\ell).$$

How to go from continuous to discrete? A sampler

Definition (Sampler)

Let A be a topological space and $d \in \mathbb{R}_{\geq 0}$ the *sampling time*. A *period-d A-sampler* is a span

$$\begin{tikzcd}
& \mathsf{Clock}_d \times \mathsf{Cnt}_A \arrow[dl, "f^{\mathrm{cnt}}"'] \arrow[dr, "f^{\mathrm{evt}}"] & \\
\mathsf{Cnt}_A & & \mathsf{Ev}_A
\end{tikzcd}$$

where:

$$\begin{aligned}
f^{\mathrm{cnt}} \colon \mathsf{Clock}_d \times \mathsf{Cnt}_A &\to \mathsf{Cnt}_A \\
(\phi, a) &\mapsto a, \\
f^{\mathrm{evt}} \colon \mathsf{Clock}_d \times \mathsf{Cnt}_A &\to \mathsf{Ev}_A \\
(\phi, a) &\mapsto \phi \mathbin{⨾} a.
\end{aligned}$$

How to go from continuous to discrete? A sampler

Definition (Sampler)

Let A be a topological space and $d \in \mathbb{R}_{\geq 0}$ the *sampling time*. A *period-d A-sampler* is a span

$$\begin{tikzcd} & \mathsf{Clock}_d \times \mathsf{Cnt}_A \arrow[dl, "f^{\mathrm{cnt}}"'] \arrow[dr, "f^{\mathrm{evt}}"] & \\ \mathsf{Cnt}_A & & \mathsf{Ev}_A \end{tikzcd}$$

where:

$$\begin{aligned} f^{\mathrm{cnt}} \colon \mathsf{Clock}_d \times \mathsf{Cnt}_A &\to \mathsf{Cnt}_A \\ (\phi, a) &\mapsto a, \\ f^{\mathrm{evt}} \colon \mathsf{Clock}_d \times \mathsf{Cnt}_A &\to \mathsf{Ev}_A \\ (\phi, a) &\mapsto \phi \fatsemi a. \end{aligned}$$

... but does this capture all engineering applications?

Solution: A level crossing sampler

Definition (L-level-crossing sampler)

Let (A, dist) be a metric space and consider a Lipschitz input stream $\mathrm{LCnt}_A(\ell)$. Consider the *level* $L \in \mathbb{R}$. A *L-level-crossing sampler* of type A is a span with $P(\ell) := \{(c, a_0) \mid c \in \mathrm{LCnt}_A(\ell), a_0 \in A\}$:

$$\begin{tikzcd} \mathrm{LCnt}_A & P \arrow[l, "f^{\mathrm{cnt}}"'] \arrow[r, "f^{\mathrm{evt}}"] & \mathrm{Ev}_A \end{tikzcd}$$

- $f^{\mathrm{cnt}}(c, a_0) = c$.
- If $\mathrm{dist}(c(t), a_0) < L$ for all $t \in \tilde{\ell}$: $f^{\mathrm{evt}}(c) = (\varnothing, !)$.
- If there exists $t \in \tilde{\ell}$ with $\mathrm{dist}(c(t_1), a_0) \geq L$: $t_1 = \inf\{t \in \tilde{\ell} \mid \mathrm{dist}(c(t_1), a_0) \geq L\}$, $a_1 = c(t_1)$.
- Recursively, define $t_{i+1} \in \tilde{\ell}$ to be the least time such that $\mathrm{dist}(c(t_{i+1}), a_i) \geq L$ (if there is one).
- $f^{\mathrm{evt}}: P(\ell) \to \mathrm{Ev}_A, \quad (c, a_0) \mapsto \{t_1, \dots, t_n, c(t_1), \dots, c(t_n)\}$

How to go from discrete to continuous? A reconstructor

Definition (Reconstructor)

Let A be a set. A *reconstructor* of type A is a span

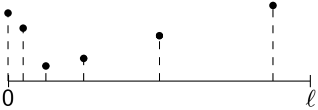
$$\begin{tikzcd} & C \arrow[dl, "f^{\mathrm{evt}}"'] \arrow[dr, "f^{\mathrm{cnt}}"] & \\ \mathrm{Ev}_A & & \mathrm{Cnt}_A \end{tikzcd}$$

with

$$C(\ell) = \{(S, a_0, a) \mid S \subseteq \tilde{\ell} \text{ finite}, a_0 \in A, a\colon S \to A\},$$

where $f^{\mathrm{evt}}(S, a_0, a) \coloneqq (S, a)$, and where $a' \coloneqq f^{\mathrm{cnt}}(a_0, s_1, \dots, s_n, a_1, \dots, a_n)\colon \tilde{\ell} \to A$ is given by

$$a'(t) \coloneqq \begin{cases} a_0, & 0 \leq t < s_1 \\ a(s_i), & s_i \leq t < s_{i+1}, \quad i \in \{2, \dots, n-1\} \\ a(s_n), & s_n \leq t \leq \ell. \end{cases}$$



How to go from discrete to continuous? A reconstructor

Definition (Reconstructor)

Let A be a set. A *reconstructor* of type A is a span

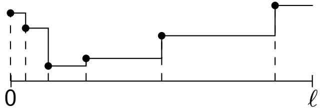
$$\begin{tikzcd} & C \arrow[dl, "f^{\mathrm{evt}}"'] \arrow[dr, "f^{\mathrm{cnt}}"] & \\ \mathrm{Ev}_A & & \mathrm{Cnt}_A \end{tikzcd}$$

with

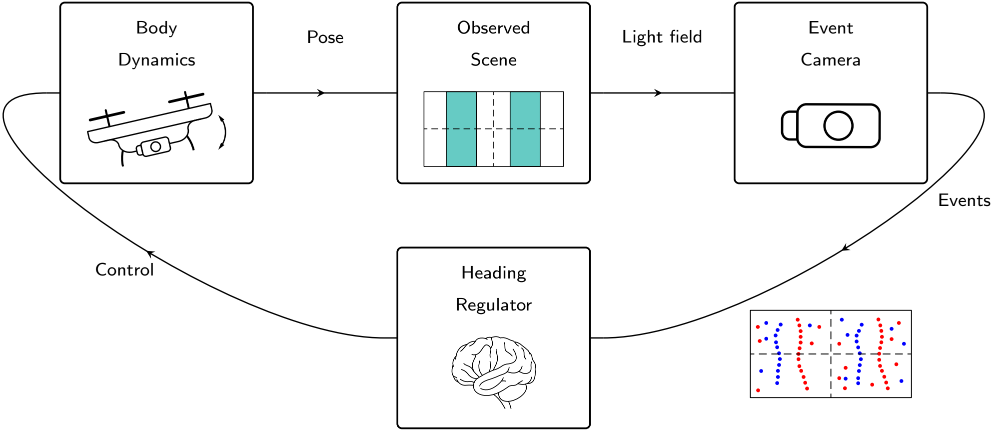
$$C(\ell) = \{(S, a_0, a) \mid S \subseteq \tilde{\ell} \text{ finite}, a_0 \in A, a \colon S \to A\},$$

where $f^{\mathrm{evt}}(S, a_0, a) \coloneqq (S, a)$, and where $a' \coloneqq f^{\mathrm{cnt}}(a_0, s_1, \dots, s_n, a_1, \dots, a_n) \colon \tilde{\ell} \to A$ is given by

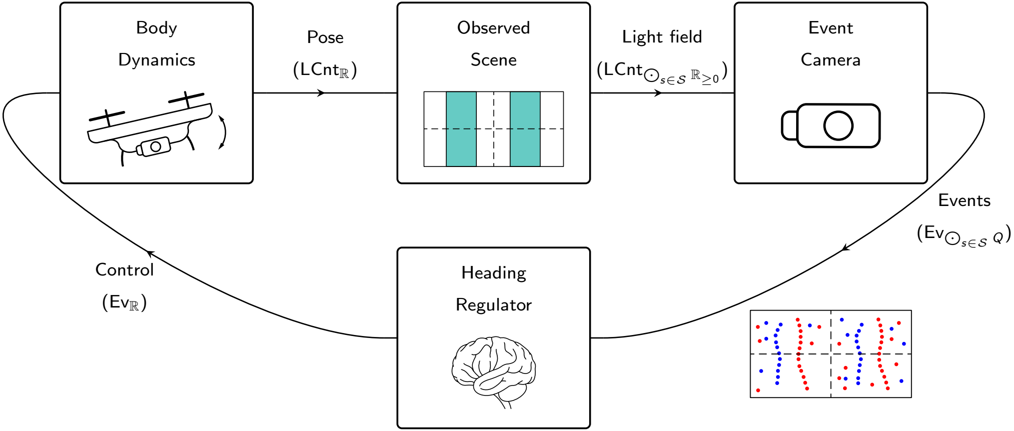
$$a'(t) \coloneqq \begin{cases} a_0, & 0 \leq t < s_1 \\ a(s_i), & s_i \leq t < s_{i+1}, \quad i \in \{2, \dots, n-1\} \\ a(s_n), & s_n \leq t \leq \ell. \end{cases}$$



Neuromorphic heading regulation problem within a unified framework



Neuromorphic heading regulation problem within a unified framework



Outlook

In engineering, we need to understand many different components in the same system.

- We developed a framework which allows putting together different components (events, clocks, continuous-time).
- This framework is *descriptive*.

Outlook

In engineering, we need to understand many different components in the same system.

- We developed a framework which allows putting together different components (events, clocks, continuous-time).
- This framework is *descriptive*.

We now want to work on the *synthesis* of such components:

- The theory of *co-design* allows design across field boundaries.