An abstract theory of sensor eventification

Yulin Zhang
Amazon Robotics*
North Reading, MA, USA
Email: zhangyl@amazon.com
* This work was done prior to joining Amazon.

Dylan A. Shell
Dept. of Computer Science & Engineering
Texas A&M University
College Station, TX, USA.
Email: dshell@tamu.edu

***Abstract*—Unlike traditional cameras, event cameras measure changes in light intensity and report differences. This paper examines the conditions necessary for other traditional sensors to admit eventified versions that provide adequate information despite outputting only changes. The requirements depend upon the regularity of the signal space, which we show may depend on several factors including structure arising from the interplay of the robot and its environment, the input–output computation needed to achieve its task, as well as the specific mode of access (synchronous, asynchronous, polled, triggered). Further, there are additional properties of stability (or non-oscillatory behavior) that can be desirable for a system to possess and that we show are also closely related to the preceding notions. This paper contributes theory and algorithms (plus a hardness result) that addresses these considerations while developing several elementary robot examples along the way.**

I. Introduction

Advances in sensing technologies have the potential to disrupt the field of robotics. Twenty-five years ago, the shift from sonar to LiDAR sensors triggered a significant change as robots became, rather abruptly, much more capable. Since information enters a robot through its sensors, a change to its sensor suite often has ramifications downstream — sometimes quite far downstream. Accordingly, it is useful to have tools to help us understand the impact of sensor modifications.

The last decade has seen steadily-growing interest in *event cameras*, a novel type of camera that operates on a separate principle from traditional devices [17], [9]. These cameras afford various new opportunities, and a growing body of work has begun exploring these possibilities [24], [41], [29], [33]. At a high level, event cameras report changes in intensity rather than an absolute measurement. These devices perform per-pixel differencing instead of operating with entire frames — the very concept of a 'frame', while crucial for devices employing a shutter (whether rolling or global), is absent from event cameras. Because this different principle of operation eases some hardware design constraints, current technology allows for consumer-grade devices that, when compared to frame-based cameras, can be much more efficient in transmitting image data and operate at considerably higher temporal resolution with greater dynamic range [9]. Not only are these performance traits attractive for reducing motion blur, but event cameras report information that is important for robots in key applications: their output naturally focuses on changes to the scene, picking up dynamic elements within the perceived environment (e.g., [20] and [42]).

Whether event cameras will form an impetus for new sets of innovative applications, or will drive some radical departure from existing methods, or even initiate a thorough re-examination of the field's underpinnings—all remains to be seen. This paper is not about event cameras *per se*. Instead, it asks the question:

> For any sensor, say, of type $X \in \{$compasses, IMUs, LiDARs, $\ldots\}$, is there a useful "event X" version?

The transformation from the raw sensor into an event version, a process which we dub *eventification*,[1] involves disentangling, conceptually, several different facets. We introduce theory by which one can formulate the preceding question in a meaningful way. The present paper is an abstract treatment of the essential properties that make event cameras interesting, expressed with reasonable rigor, and in adequate detail to lay open some connections that were not immediately apparent.

The long-term goal of this theory is to try to change the way our field interacts with the areas of sensor design and with signal processing researchers. Today, most roboticists are consumers: we see what is out there, we buy something from a catalogue, we bolt it to a robot, and then integrate it with software. Robot use-cases (i.e., task performance) should play a greater role in informing what sensors ought to exist, what should be designed, and how manufacturers might target roboticists.

A. Related work

Our work was inspired by the recent paper of Zardini, Spivak, Censi, and Frazzoli [38], wherein the authors provide a compositional architecture with which they express a model of a UAV system. That robot system has, as one specific sensor, an event camera. Their model leads one to ponder whether the 'eventfulness' of the camera might be obtained by some abstract transformation of a traditional camera — if so, what would such a transformation look like? Hence the present paper, which retains some of the spirit of their work. That same spirit is also apparent in the important, early paper of Tabuada, Pappas, and Lima [35] which provides an expressive mathematical framework through which aspects of robotic systems' behavior can be represented and examined.[2] Their work employs equivalence based on bisimulation; the notion of output simulation we employ is similar (but known to be

[1]A term inspired by [25].
[2]A recent ICRA workshop [43] attests to expanding interest in such topics.

distinct, cf. [28]). The concept of stutter bisimulation —where sequences may have repeated subsequences— was introduced and studied in the early model checking literature [1], [5], though we are not aware of applications to robotics. The present paper can be understood as generalizing output simulation so that, among other things, it may also treat a form of stutter.

The question of whether some sensors provide a system with a sufficiency of information has roots in the classic notion of observability [13], [11]. More recently, and more directly in the robotics community, the subject has been related to concepts such as perceptual limits [7], information spaces [14] (originally of von Neumann and Morgenstern), and lattices of sensors [15], [40]. Erdmann's work [8] reverses the question, asking not what information some given sensor provides, but what a (virtual) sensor ought to provide. His action-based sensors become, then, a computational abstraction for understanding the discriminating power needed to choose productive actions. The idea of the discriminating power and a (virtual) sensor wrapping some computation permeates this paper's treatment as well. Whether some transformation undermines the ability to extract sufficient information, especially as a model for non-idealized sensors, appears in [31] — a paper which we shall refer to again, later. An important class of transformations are ones that seek to compress or reduce information. These fit under the umbrella of minimalism, an idea with a long history in robotics [3], [21], but with adherents of a more recent generation having a greater focus on algorithmic [26] and optimization-based tools [27], [39].

Neuromorphic engineering, the field that pioneered event cameras, is concerned with a class of devices much broader than just cameras [18]. In recent years, along with advancements in spiking-neuron and neural computing [4], [22], [23], event-driven tactile sensors [36], synthetic cochlea [19], chemical concentration and gas detection sensors [34], [37] have been developed. We feel the robotics community could be better at informing sensor designers about what devices would be germane for robot use.

B. Paper Organization with a Preview of Contributions

The next section deals with preliminaries and begins by introducing, with some basic notation, definitions that have mainly been established elsewhere. Section III introduces the core notion of substitutable behavior (Definition 5) on which this work is based; it takes a new and general form, subsuming and unifying two previous concepts, while affording much greater expressive power. In Section IV this power is put to use. We give a basic structure, which we call an observation variator (Definition 9), that is capable of reporting differences in the signal space, leading to the formation of a derivative. We pose a form of optimization question, asking how to find a smallest variator, and then establish that minimization is NP-hard (Theorem 18). As we then show, modes of data acquisition affect the sensor's power, so Section V turns to this in depth, moving beyond synchronous data flow. The key result (Theorem 27) is that polling and event-triggered acquisition modes are equivalent to one another. Section VI considers the fully asynchronous data acquisition mode; doing so requires the variator to have additional structure (Definition 28, a monoidal variator). The problem, when expressed directly, appears complicated; we construct a conceptually simpler version, and show that they are actually equivalent (Theorem 35). The penultimate section motivates and examines some simple notions of stable behavior, which ensure the sensor will not chatter. But fortunately chatter-free behavior can be obtained, essentially for free, in problems of interest (Theorem 39). Section VIII offers a brief summary of the paper.

Overall, the work explicates the concept of eventification, and then identifies and explores some further connections. With an eye toward an axiomatic theory of sensing, some care has been exercised to be economical: additional structure is introduced just when actually demanded; for instance, only in Section VI do any algebraic properties make an appearance.

II. Background: filtering problems

To be analogous to event cameras, event sensors must couple raw sensor devices (i.e., physical components and electronics for energy transduction) with some computation (e.g., signal differencing). Thus, our treatment will consider them to be units that are abstract *sensori-computational devices* (borrowing this term of Donald [6]). These units implement a kind of abstract sensor (here, a term inspired by Erdmann [8]). We will use procrustean filters, a basic framework for treating (potentially stateful) stream processing units, to model such sensori-computational devices:

Definition 1 ([31])**.** A *sensori-computational device* is a 6-tuple (V, V_0, Y, τ, C, c) in which V is a non-empty finite set of states, V_0 is the non-empty set of initial states, Y is the set of observations, $\tau : V \times V \to \wp(Y)$ is the transition function, C is the set of outputs, and $c : V \to \wp(C) \setminus \{\varnothing\}$ is the output function. (We write $\wp(A)$ to denote the powerset of set A.)

A sensori-computational device translates between streams of discrete symbols. These objects are transition systems for processing streams, with finite memory (represented as the set of states) used to track changes in sequences as they're being processed incrementally. Acting as transducers, they receive a stream of observations as input, revealed one symbol at a time, and generate one output per input symbol. In our setting, the observations will come from a raw sensor or after some simple post-processing; outputs, represented abstractly as colors, encode either actions (for a policy) or state estimates (for a filter).

The sets of states, initial states, and observations for F are denoted $V(F)$, $V_0(F)$ and $Y(F)$, respectively. All the sensori-computational devices throughout this paper (i.e., units modeled, in the terminology of [31], via some filter F) will just be presented as a graph, with states as its vertices and transitions as directed edges bearing sets of observations. For simplicity, for all such devices we shall assume that $Y(F)$ is finite. The values of the output function will be visualized as a set of colors at each vertex, hence the naming of C and $c(\cdot)$.

Given a particular sensori-computational device $F = (V, V_0, Y, \tau, C, c)$, an observation sequence (or a string) $s = y_1 y_2 \dots y_n \in Y^*$, and states $v, w \in V$, we say that w is *reached by* s (or s *reaches* w) when traced from v, if there exists a sequence of states $w_0, w_1, \dots, w_n$ in F, such that $w_0 = v$, $w_n = w$, and $\forall i \in \{1, 2, \dots, n\}, y_i \in \tau(w_{i-1}, w_i)$. (Note that Y^* denotes the Kleene star of Y.) We let the set of all states reached by s from a state v in F be denoted by $\mathscr{V}_F(v, s)$ and denote all states reached by s from any initial state of the filter with $\mathscr{V}_F(s)$, i.e., $\mathscr{V}_F(s) = \bigcup_{v_0 \in V_0} \mathscr{V}_F(v_0, s)$. If $\mathscr{V}_F(v, s) = \varnothing$, then we say that string s *crashes* in F starting from v.

We also denote the set of all strings reaching w from some initial state in F by $\mathscr{S}^F_w = \{s \in Y^* | w \in \mathscr{V}_F(s)\}$. The set of all strings that do not crash in F is called the *interaction language* (or, briefly, just *language*) of F, and is written as $\mathscr{L}(F) = \{s \in Y^* | \mathscr{V}_F(s) \neq \varnothing\}$. We also use $\mathscr{C}(F, s)$ to denote the set of outputs for all states reached in F by s, i.e., $\mathscr{C}(F, s) = \bigcup_{v \in \mathscr{V}_F(s)} c(v)$. When s crashes, the vacuous union gives $\mathscr{C}(F, s) = \varnothing$. Definition 1 ensures that any $\mathscr{L}(F)$ contains at least ε, the empty string; we have $\mathscr{C}(F, \varepsilon) = \cup_{v_0 \in V_0} c(v_0)$.

We focus on sensori-computational devices with deterministic behavior:

Definition 2 (deterministic)**.** An sensori-computational device $F = (V, \{v_0\}, Y, \tau, C, c)$ is *deterministic* or *state-determined*, if for every $v_1, v_2, v_3 \in V$ with $v_2 \neq v_3$, $\tau(v_1, v_2) \cap \tau(v_1, v_3) = \varnothing$. Otherwise, we say that it is *non-deterministic*.

Algorithm 2 in [30] can turn any non-deterministic sensori-computational device into one with an identical language but which is deterministic. Hence, without loss of generality, in what follows all sensori-computational devices will be assumed to be deterministic.

Overwhelmingly we shall give simple examples, but one easily gains expressive power by constructing complex sensori-computational devices by composing more elementary ones:

Definition 3 (direct product)**.** Given $F = (V, V_0, Y, \tau, C, c)$ and $F' = (V', V'_0, Y', \tau', C', c')$, then their *direct product* is the 6-tuple $F \times F' = (V \times V', V_0 \times V_0', Y \times Y', \tau_{F \times F'}, C \times C', c_{F \times F'})$ with

$$\begin{aligned} \tau_{F\times F'}: \quad & (V \times V') \times (V \times V') \to \wp(Y \times Y'), \\ & ((v_i, v'_j), (v_k, v'_m)) \mapsto \tau(v_i, v_k) \times \tau'(v'_j, v'_m); \end{aligned}$$

$$\begin{aligned} c_{F\times F'}: \quad & (V \times V') \to \wp(C \times C') \setminus \{\varnothing\}, \\ & (v_i, v'_j) \mapsto c(v_i) \times c'(v'_j). \end{aligned}$$

(Note: To save notational bloat, we shall only present pairwise products, trusting the reader will be comfortable with the obvious extension to any finite collection.)

Remark 1. If $s_1 s_2 \dots s_n \in \mathscr{L}(F \times F')$ then each $s_i = (y_i, y'_i)$ has $y_i \in Y(F)$ and $y'_i \in Y(F')$, and further $y_1 y_2 \dots y_n \in \mathscr{L}(F)$, and $y'_1 y'_2 \dots y'_n \in \mathscr{L}(F')$. The converse, however, needn't hold: e.g., for some $y_1 y_2 \dots y_n \in \mathscr{L}(F)$ there may exist no $s_1 s_2 \dots s_n \in \mathscr{L}(F \times F')$ with $s_i = (y_i, y'_i)$.

The standard way to compare sensori-computational devices is in terms of input-output substituability, that is, whether one can serve as an functional replacement for another. The following expresses this idea.

Definition 4 (output simulation [26])**.** Let F and F' be two sensori-computational devices, then F' *output simulates* F if $\mathscr{L}(F) \subseteq \mathscr{L}(F')$ and $\forall s \in \mathscr{L}(F) : \mathscr{C}(F', s) \subseteq \mathscr{C}(F, s)$.

If F' output simulates F, the intuition is that then any stream of observations that F can process can also be processed effectively by F'; the outputs that F' yield will be consistent with those F could produce. In terms of functionality, F' may serve as an alternative for F.

When considering F' and F, often F would be treated as providing a specification (with $\mathscr{L}(F)$ circumscribing aspects of the world that may arise, and $\mathscr{C}(F, \cdot)$ characterizing suitable outputs); an output simulating F' realizes behavior that is acceptable under this specification. This is because such a sensori-computational device F' is able to handle all strings from F and yields some suitable outputs for each string. Note that the output may be the result of some sort of estimation (like a combinatorial filter [14]), or the output may be a representation of an action to be executed, and so encode a policy (e.g., [26]).

III. Generalized output simulation

For this paper, the point of departure is a more general notion of output simulation. We consider a case where one may specify some relation that modifies the strings of one sensori-computational device, so that the second device must process strings through (or in the image of) the relation.

Definition 5 (output simulation modulo a relation)**.** Given two sensori-computational devices F and F', and binary relation $\mathrm{R} \subseteq A \times B$ we say that F' *output simulates* F *modulo* R, denoted by $F' \sim F \pmod{\mathrm{R}}$, if $\forall s \in \mathscr{L}(F)$:

1) $\exists t \in \mathscr{L}(F')$ such that $s \mathrm{R} t$;
2) $\forall t \in B$ such that $s \mathrm{R} t$, $t \in \mathscr{L}(F')$ and $\mathscr{C}(F, s) \supseteq \mathscr{C}(F', t)$.
   (Notice that, as $t \in \mathscr{L}(F')$, $\mathscr{C}(F', t) \neq \varnothing$.)

Some sensori-computational device F is *output simulatable modulo relation* R if there exists some F' which output simulates F modulo R. More concisely, in such cases we may say that F is R*-simulatable*. When F' output simulates F modulo R, intuitively, the streams of observations F can process can also be effectively processed by F' after they've been pushed through binary relation R. Because R may be 1-to-1, 1-to-many, many-to-1, or many-to-many, this generalization gives the ability to treat several phenomena of interest.

Remark 2. As most relations we will use are binary relations, we'll suppress the 'binary' qualifier in that case. Also, when some relation is a (partial or total) function and it is clearer to