Virtual Reality for Robots

Markku Suomalainen[1] Alexandra Q. Nilles[2] Steven M. LaValle[1]

***Abstract*— This paper applies the principles of Virtual Reality (VR) to robots, rather than living organisms. A simulator, of either physical states or information states, renders outputs to custom displays that fool the robot's sensors. This enables the design of targeted experiences for the robot that are more realistic than pure simulation, yet more feasible and controllable than real-world experiences. Potential applications include testing procedures that are better than simulation, reverse engineering of unknown robots, the study of spoofing attacks and anti-spoofing techniques, and sample generation for machine learning. A general mathematical framework is presented, along with a simple experiment, detailed examples, and discussion of the implications.**

I. Introduction

Imagine a food delivery robot, such as the Kiwibot in Fig. 1a. Assume it gets input data from three sensors: GPS, a front-facing camera, and wheel encoders. When the robot moves in cities and among people, it can easily lose GPS signal, get kicked or stuck, or even get kidnapped. These scenarios raise a question: What is a feasible but realistic method for replicating these scenarios during development and testing? Issues such as system integration or systematic sensor errors cannot be completely replicated in simulation; thus, we would like to prioritize interaction with the actual, physical robot. How can we use simulation to achieve principled, systematic interaction with the robot? How would such an interaction affect testing, reverse engineering, and verification of robotic systems?

Virtual Reality (VR) uses simulation and displays to trick humans and other organisms into believing they are having a perceptual experience that is different from reality. This experience is usually interactive and carefully crafted. Humans might, for example, believe they are exploring exotic worlds in a game. Other organisms, such as monkeys, rats, or *Drosophila*, interact with artificial worlds so that scientists can study their creation of neural place and grid cells [10] or other neurological phenomena. This paper explores a natural question: *What would happen if the living organisms are replaced by a robot?* This immediately raises additional questions. What new approaches would this enable? How would this form of VR be generally defined and achieved, well beyond what might be imagined from Fig. 1b? How would VR be successfully achieved in this context? We call this class of problems *Virtual Reality for Robots (VRR)*.

This work was supported by Business Finland project HUMORcc 6926/31/2018, Academy of Finland project PERCEPT, 322637 and US National Science Foundation grants 035345, 1328018.

[1]M. Suomalainen and S. M. LaValle are with Center of Ubiquitous Computing, Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland. firstname.lastname@oulu.fi

[2]A. Nilles is with the Department of Computer Science, University of Illinois at Urbana-Champaign , Illinois, USA. nilles2@illinois.edu

(a) (b)

Fig. 1: (a) A Kiwibot food delivery robot. (b) A robot in VR: An Optofidelity Buddy 3 wearing an Oculus Go headset.

We envision at least four distinct applications for VRR: 1) The designers of a robot system or algorithm want to evaluate its robustness in environments that are difficult to reach or construct, or infeasible to simulate. In this case, VRR provides a hybrid approach of real and virtual worlds that is more realistic than simulation but more accessible than a normal deployment. To easily verify sensor errors or attacks, VRR allows us to create scenarios that are extraordinary or implausible, such as the *kidnapped robot problem* [5] in which the robot is unexpectedly transported into another environment. 2) To reverse engineer the design of an unfamiliar robot, VRR can put it through carefully designed, contrived scenarios. The robot receives an adaptive series of tests to determine its inner decision making strategies. 3) As security and delivery robots increase in popularity, their protection against sensor spoofing attacks must be considered. VRR could be used to intentionally create sensor spoofing attacks on the actual sensors, and to study the robustness of the robot system. 4) In a machine learning context, precious new data can be generated by measuring how the actual robot responds to numerous hybrid scenarios, addressing the gap between simulation and reality (*sim-to-real*; see e.g. [21]).

To the best of our knowledge, this the first attempt to generally introduce and mathematically define VRR. The most similar related work is [9], in which real drone dynamics (infeasible to simulate) are combined with a simulated visual display. *Spoofing* literature is also related, which considers adversarial attacks against sensor systems. Many attacks are against biometric security systems, such as face [7], fingerprint [33] or speech [35] recognition systems. There also exists mathemetical analysis on when spoofing is feasible [36], [37]. These works, however, aim at fooling a classifier, whereas VRR is meant for continuous fooling of entire sensing and information processing subsystems.

Other recent works show that MEMS sensors such as Inertial Measurement Units (IMU)s can be distracted [27] and even controlled [29] with external amplitude-modulated noise. LiDAR, a key component in many autonomous cars, has been shown to be susceptible to spoofing attacks [26]. GPS is not immune either, and GPS-based capture of au-

tonomous vehicles is a major concern [11]. Anti-spoofing methods for drones have also been proposed, by observing whether the combination of sensor inputs obeys the laws of physics [4]. These examples help to enable VRR (we can provide controllable input to real sensors) and further motivate it. To study and counter spoofing attacks, the concept of VRR must be well defined and understood.

Section II introduces background on virtual reality and a provides a concrete robotics scenario. Section III provides a mathematical framework that captures the essence of VR for living organisms, but characterizes it in the general context of robots. This framework builds upon the usual state spaces (with configurations and environments), sensor mappings, and state transition mappings; we then introduce VR-specific notions such as a virtual world generator, renderers, and displays, which are used to fool the robot's sensors. VRR displays do not necessarily resemble a display or video screen in the usual sense; instead, each is custom designed to spoof a particular sensor, techniques for which will be explained in Section IV. Section V discusses virtual world and rendering challenges. Section VI presents a simple mobile experiment, and Section VII concludes by assessing the differences between VR and VRR, and speculating on the implications of this work.

II. Experiencing Virtual Reality

In this section we give background on Virtual Reality (VR) for humans and other living organisms. Then, we will show the connections to VRR, and define *full VR* and *partial VR*.

A. How living organisms experience virtual reality

VR for living organisms can be defined as "inducing targeted behavior in an organism by using artificial sensory stimulation, while the organism has little or no awareness of the interference" [15]. Creating "targeted behavior" requires tracking the organism and rendering the virtual world accordingly, as shown in Fig. 2. "Awareness of the interference" refers to the phenomenon of *presence* [24], and is an important criterion of a successful human VR experience. Interestingly, according to "poison theory" [18] the failure of human sensor fusion to accept a credible situation is proposed as one of main reasons causing cybersickness; in such a situation being poisoned is a possible cause, and thus vomiting is a reasonable reaction. Similarly, if a robot expects to be spoofed, such a non-viable set of sensor inputs could warn the system of this possibility, analogously as proposed in [2].

Next, consider "artificial sensory stimulation." Fig. 2 demonstrates how a VR experience is generated. It shows that VR is not limited to visual stimuli but includes *all* possible sensing modalities. Besides vision, common sensing modalities include audio and tactile feedback, proprioception (through e.g., treadmills), olfaction [23], and even Electrical Muscle Stimulation (EMS) [19]. Also, fooling vision does not necessarily require an Head-Mounted Display (HMD). For example, the "visual odometry" of honeybees can be influenced by changing the pattern of a tunnel they flew through [28]. Also, for humans, *CAVE* systems (i.e., surrounded by screens) are considered VR [3].

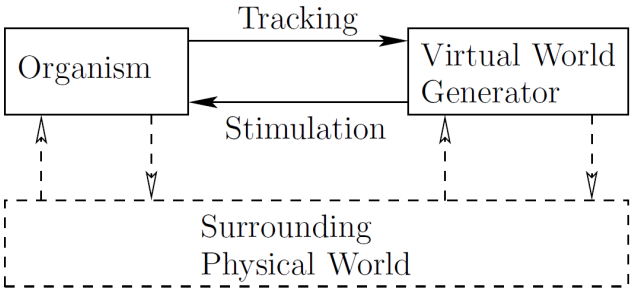


Fig. 2: The organism interacts directly with a virtual world generator, which tracks the organism and has renderers that calculate display outputs based on the simulation state. Rendering on displays causes stimulation of the organism.

If a VR system does not target all sensing modalities, we will call it *partial VR*; let *full VR* mean that all sensors are targeted. Biological sensor systems such as proprioception, the vestibular system, and temperature/pressure sensing in the skin are quite complex and incomprehensible. Thus, achieving *full VR* on humans is practically impossible. Some simplifications are possible; for example tilting forward a human wearing a HMD can simulate accelerating due to gravity. However, for robots, full VR is more feasible. Various levels of partial VRR can thus be considered.

B. How robots might experience virtual reality

We consider VRR by direct adaptation of how VR is currently defined for humans. We assume that the robot's sensors are not bypassed to inject information directly into the processor. This would be similar to a Brain-Computer Interface (BCI) for humans, and for robots it would be essentially simulation and thus hamper the sensor testing advantage of VRR. In contrast, we assume that the only way to affect the robot is to interact with its sensors through a *display*, to which *stimuli* are *rendered* from a *virtual world*, as shown in Fig. 2. For the stimuli to be rendered correctly, the VRR system must also *track* and possibly *predict* the actions of the robot.

Rendering is not limited to a visual display, but encompasses all the sensors being fooled, to be discussed in Section IV. This provides the main strength of VRR: We can create a mixed world of *real* and *virtual* sensor inputs. For example, simulation of dynamics involving air and fluid flow, or slipping on granular materials, is still computationally intractable and thus prone to artifacts and assumptions in simulation. Moreover, in simulators it is often assumed that sensors are perfect, or if noise is considered, then the injected noise is almost always assumed to be Gaussian. However, it is usually the case that the Gaussian assumption is inadequate. There may be unexpected systematic errors not captured in simulation, such as temperature dependence of sensors, or interference between electromagnetic or ultrasonic sensors. By retaining real sensors on the robot, VRR would enable targeted experiments with results that can be more reasonably expected to transfer to deployment.

C. A motivating example

Recall the food delivery robot from Fig. 1a. First, imagine the robot is equipped with only a single sensor, such as a camera. For such a robot we can only perform full VR

because with one display all of the robot's sensors are fooled, and its actions will be based only on the stimuli coming from the display, and its internal programming. Next, assume that the robot is equipped with more sensors, such as a forward-facing camera, range finders pointed in four orthogonal directions, GPS, and wheel encoders. The robot navigates to GPS waypoints, using wheel encoder data to augment the GPS state estimation. It also avoids static and dynamic obstacles using data from its camera and range finders. Whereas performing full VR on such a multi-sensor robot is possible, it is often too complicated and its main applications would be spoofing and reverse engineering. For verification and learning, full VR bypasses one of VRR's main strengths, which is the combination of real and virtual sensor inputs. Thus, we will consider only *partial VR* on the following example. The type of partial VR needed will depend on the use case of VRR.

Imagine a few possible scenarios for each of the four purposes described in Section I: 1) *robustness:* The robot needs to be able to drive over potholed and graveled streets while avoiding static and dynamic obstacles. 2) *reverse engineering:* the robot has been accused of colliding with a small child and the robot creators want to analyze all the possible conditions that could cause such an event; 3) *spoofing:* someone is feeding false GPS data into a security robot to derail it from the usual circle; 4) *learning:* the robot designers are training a control policy that safely navigates through crowds of humans, but they wish to avoid needing to spend many hours walking around the robot to collect training data.

In the first scenario, we want to test the entire robot system in environments that are difficult to simulate, with a mixture of real and virtual inputs. For example, the robot could drive across a real street while the camera and range finders are stimulated to create obstacles that require driving across potholes. In the second scenario, we would need to use the actual robot and create many scenarios to discover the cause of the collision-avoidance bug. In the third scenario, we would need to test that anomaly-detection code and fallback strategies work well in many different situations. In the final scenario, training is expensive to do with real humans, but the need for safety means we would like to make sure the trained policy works on the actual robot before full deployment.

III. General Mathematical Framework

This section takes the examples from Section II as a starting point and both formalizes and generalizes them into a precise mathematical framework that captures VRR and extends naturally from typical robotics models.

Consider a generic robot that is equipped with sensors, actuators, and computation. Let X denote its standard *state space*, which could be its configuration space or more general phase space to include time derivatives. Let U denote its *action space*, which corresponds to the set of commands that can be given to the robot. The robot has one or more sensors, which are each modeled by a sensor mapping; the most common form of this mapping is $h : X \to Y$, but in general, X could be replaced by a space that includes time, state histories, or unpredictable disturbances [17]. Each $y = h(x)$ is called a *sensor observation*.

A *state transition equation* $x' = f(x, u)$ determines the effect of the action $u \in U$ when applied at state $x \in X$, resulting in a new state $x' \in X$. The system feedback loop could occur over discrete time at fixed intervals, be event driven, or any other common possibility.

The robot selects an action u according to a *plan*, which has the form $\pi : \mathcal{I} \to U$, in which $\mathcal{I}$ is an *information space*, defined for robots in [14], but derived from [1], [31]. Thus, the action $u = \pi(\eta)$ is chosen based on an *information state* or *I-state* η, which is derived (a mapping) from initial conditions, the sensor observation history, the action history, and possibly the state transition equation and sensor mappings. A common example is that in a Bayesian setting, η corresponds to a posterior pdf that takes into account all models and existing information. As another example, η could simply be the most recent sensor observation, y, resulting in pure sensor feedback. Note that there is no direct access to the state x at any time (unless a powerful enough sensor can measure it, which is unlikely in practice).

To provide VRR, the robot should be "tricked" into executing its plan as designed, even though it may not traverse the state space in the way that it "believes". The safest way to implement this is to ensure that every sensor provides observations that the robot would expect based on execution of its plan. Because the sensors are an integral part of VRR, we assume we cannot bypass them and directly manipulate the robot's memory and I-states.

Let D denote a *display output space*, in which a particular *display output* is denoted by $d \in D$. A display is associated with a sensor, implying a relationship in which the display output d causes a targeted observation y. Thus, there is a function $\sigma : D \to Y$, called the *spoof mapping*. More generally, the spoof mapping may depend on state, to yield $\sigma : D \times X \to Y$.[1]

How does the display know what to output? For human-based VR, a *Virtual World Generator (VWG)* (see Fig. 2) maintains a virtual world, which is then rendered to displays according to the tracked configuration of the human. The same idea is needed for VRR. Let S denote the *virtual state space*. Note that S and X could be the same or vastly different. Each display uses the state $s \in S$ to determine the output through a *rendering mapping*, $r : S \to D$. Thus, $d = r(s)$ is the rendered output to the display when the VWG is in state s.

More generally, the display output might depend on both the physical state x and the virtual state s. In this case, the rendering mapping is $r : S \times X \to D$ and $d = r(s, x)$. This analogously happens in human-based VR, in which we must know where the user is looking (equivalent to x) and what in the virtual world needs to be rendered (equivalent to s). The implementation of r might then require a tracking system to estimate x (analogous to VR head tracking [16], [32]), but in this paper we will assume that the VRR system includes sufficiently accurate tracking.

There are now two options for creating the Virtual World Generator, depending on our knowledge of the robot:

[1]Of course, the display is embedded in the physical world. We use this notation for the spoof mapping for clarity, implicitly defining X as "the rest of the world." Ideally, the display would not alter configuration space obstacles; in reality this will depend on the design of the display.

1) *Black-box robot:* If we have no knowledge of the internal algorithms of the robot, then the VWG should maintain a complete and perfect virtual state space, mimicking the behavior of the real world with sufficient fidelity that the appropriate display outputs can always be determined.
2) *White-box robot:* If we know the internal algorithms of the robot, the VWG can directly induce the transitions of I-states inside of the robot with an incomplete or imperfect virtual state space; the VWG need not maintain a high-fidelity artificial world.

The first choice is appropriate when we do not have direct access to the I-states. This is the common situation in human-based VR, in which it is impossible to measure or understand the brain's I-state (all relevant neural activity). The second choice is available for VRR but not human-based VR because we might have access to the robot's design. The implications of this are quite powerful. For example, if we know that the I-state completely ignores one sensor, then there is no need to design a display for it. Similarly, if the I-state severely quantizes sensor observations, then the display resolution could be significantly lowered. The VWG may more resemble a simple state machine that emits the correct displays rather than a physics simulator.

A. Black-box robots

Consider the case of a robot where we know its sensors and their mappings, and can track the robot's actions, but know nothing of its algorithms or internal state. The VWG ideally would construct and maintain a complete physically plausible world, of sufficient fidelity that no sensor would be able to detect the presence of the artificial display. Precise information about the robot's supposed configuration in a fictitious environment is maintained by the tracking system.

The sensor mapping implies a *sensor preimage* for each sensor reading $y \in Y$, defined as

$$h^{-1}(y) = \{x \in X \mid y = h(x)\}. \quad (1)$$

Since sensors usually are many-to-one mappings, $h^{-1}(y)$ could be a large subset of X. For example, a proximity sensor that returns TRUE if the sensor is within five centimeters of a wall, and FALSE otherwise, $h(x) =$ TRUE would induce a preimage that correspond to all states that put the sensor within five centimeters of a wall.

In general, the collection of sensor preimages for all possible sensor readings forms a partition of X. For a given sensor mapping h, let the partition be denoted $\Pi(h)$. The sets in $\Pi(h)$ should be thought of as equivalence classes, because for any two x_i, x_j in the same sensor preimage, $h(x_i) = h(x_j)$ and the states are indistinguishable with that single sensor.

This has implications for the necessary resolution of the virtual state space. Whereas X may be continuous, S can be formed by discretizing X as the common refinement of all partitions $\Pi(h)$ for all sensors. Within each element of S, all corresponding states x will then be guaranteed to produce the same sensor readings. Two challenges remain: First, some sensors (such as cameras) have such fine-grained mappings that this approach would be prohibitively complex. Second, a computational burden is added of needing to compute whether the robot will transition between states in S, based on its state transition model in X. However, especially for robots with simple sensing modalities (such as swarm robots or micro-scale applications), this approach can dramatically simplify the corresponding VRR system.

B. White-box robots

For the case of a white-box robot, we focus on how to use the VRR system to induce specific targeted behaviors by reasoning in I-states. The I-states of the robot may correspond to a set of possible physical states, a belief distribution over the physical state space, or something more abstract. For example, a simple vacuuming robot may have I-states that correspond to cleaning in free space, cleaning along a wall, having completed cleaning, and has reactive transition rules between these states and their associated behaviors.

For simplicity, assume the robot has a finite I-state $\mathcal{I}$, action space $\mathcal{U}$, and sensor mapping h for every sensor. Also assume that the plan $\pi : \mathcal{I} \to U$ is known. Assume an event-based transition model, in which the robot performs an action and gets a sensor observation at each transition. Each sensing and action history can be thought of as an input string that drives the robot into a specific I-state. The traversal of the information space can be mathematically treated as a deterministic finite automaton, with state space $\mathcal{I}$, input symbol space Y, and a state transition map that yields next I-states based on the current I-state, action according to π, and observation y.

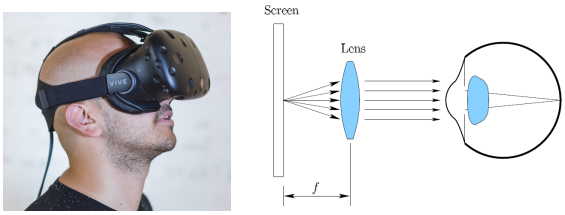
If we know the initial I-state, then inducing specific behaviors involves searching for a sequence of sensor values that drive the robot to the target I-state. The case in which we do not know the initial I-state is more interesting because it allows for analysis of behavior over all possible initial conditions, and is a first step toward reverse engineering an unknown robot.

We now develop a first, simple, theoretical result for VRR. Assume a robot is given, with known sensors, action space, plan, and deterministic finite information space, and that the robot's actions are hidden or unknown.

Proposition: *A polynomial-time algorithm in $|I|$ and $|Y|$ exists that computes a sequence of targeted sensor observations that will drive the robot to any desired target state, or else decide that no such sequence exists.*

Sketch of proof: In the *synchronizing word* problem in automata theory, we seek a word in the input alphabet of a given deterministic finite automata (DFA) that will send any state of the DFA to one and the same state [30]. In our problem, this is equivalent to finding a sequence of sensor readings that would cause a robot with known structure and plan to be in a specified I-state after the sequence. For n-state DFAs over a k-letter alphabet, algorithms exist to find the existence of and example of a synchronizing word of length polynomial in n and k, in polynomial time [6]. ■

The above is an "open-loop" approach to inducing targeted behaviors in a known robot; we do not observe the actions that the robot takes or use them to estimate the current I-state. The problem of adaptively estimating the I-state is an interesting open problem. If we can control some or all



(a) HMD (b) Optical system

Fig. 3: (a) For human-based VR, the human vision system is spoofed by wearing a screen on the head and blocking external light. (b) A lens is placed between the screen and eye so that the display appears to be further away.

inputs to a robot's sensors, then under what conditions can observing its resulting actions give us enough information to conclusively determine its I-state? Can the approach be extended to the case where we are attempting to infer the structure of the information space, or deciding between a few candidate information space models?

IV. Designing Displays

The mathematical framework of Section III is simple but abstract. This section provides more details on how displays are designed to confuse particular sensors. This results in the spoof mapping $\sigma : D \rightarrow Y$.

A. Displays for a camera

Suppose the robot uses a standard RGB camera, for example, with VGA resolution, global shutter, and standard lens. A sensor observation $y = h(x)$ would correspond to a complete specification of an image captured by the camera. The observation space Y is large, containing $640 \times 480 \times 256 \times 3$ elements. We want a display so that for any y of interest, there exists a display output $d \in D$ for which $y = \sigma(d)$.

A common means to spoof a camera is by placing an ordinary RGB screen in front of it; this approach has been taken to spoof security systems in [12]. For VRR, the approach is analogous to a human wearing an HMD; see Figure 3a. Each display output d would specify the eight-bit RGB values every pixel in a display image. For a standard 1080p display panel, this would imply that the display output space D contains precisely $1080 \times 1920 \times 256 \times 3$ values.

Placing a standard smartphone or tablet display over a camera causes several issues: 1) ambient light, if not blocked, also affects the camera input; 2) A lens is required between the display and the camera to make the camera focus correctly (see Figure 3b), unless an exotic alternative, such as a light field display [13], is used; 3) The camera might capture the rolling scanout of the display images, rather than a complete image, which suggests that vsync must be used to control the display and its frame rate should be significantly larger than that of the camera; 4) The resolution of the display should be significantly higher than that of the camera, in terms of pixels per degree; even though in theory matching resolution should suffice, the display is almost impossible to align perfectly to avoid severe quantization artifacts; 5) numerous other problems might affect performance, such as noise, limited dynamic range of the display, and optical aberrations.

In the case of a black-box robot, it is generally assumed that the robot's internal processing is unknown. However, in the case of a white-box robot, algorithms such as image processing, computer vision, and SLAM, could be known. What if the camera is merely used to detect and track large blobs of solid colors? In this case, a much simpler display might be sufficient, which could have a lower resolution or frame rate. Whereas this may not be a meaningful use case for the presented white box purposes of verification and learning, it is a powerful idea for *grey box* (where we have partial knowledge of the robot's internal algorithms) versions of spoofing and reverse engineering because in these cases we often have an idea how the algorithms behave and we strive for minimalism, due to practical reasons, in display and virtual world design.

B. Displays for contact or proximity sensors

Next, consider simple sensors for which $Y = \{0, 1\}$, such as a mechanical contact or bump sensor. In one mode, $y = 0$, there is no contact. In the other mode, $y = 1$, the bumper is pressed and contact is made. In this case, the display needs only to press the bumper to spoof the robot, which can be accomplished by a mechanical attachment. In this case, $D = \{0, 1\}$, and the spoof mapping takes an obvious form: $y = \sigma(d) = d$. Thus, a "display" in this case merely smacks the contact sensor so that it reports contact! The situation is similar for a typical proximity sensor. If proximity is detected by a simple infrared detector, then an object needs to be placed into its field of view to report detection. The set D and mapping σ remain the same as for the contact sensor. Naturally this interference must occur at desired intervals to create a "virtual world" for the robot. Thus, the VWG must maintain information from which occlusions are rendered as appropriate to make the robot perform the targeted behavior.

C. Other display examples

A display could be designed for any sensor aboard a robot. A force/torque sensor can be fooled by a more complicated smacker, such as a robot arm. A method for building a display for LiDAR was presented in [26], such that objects can appear closer than the display, or are even erased from the LiDAR. Although we did not find existing work for other sorts of distance sensors, it is not difficult to imagine displays such as a fully sound-absorbing surface with a microphone and a loudspeaker for a sonar, or a system of adjustable mirrors for infrared.

A display that would fool wheel encoders is challenging to consider. Without tampering with the electronics, either the wheel or the whole robot must be somehow altered. An interesting example is fooling ants' odometers by shortening or lengthening their legs [34]. If the geometry of the robot allows, the size of the wheels could be altered to mimic systematic errors in wheel encoders. If the robot is suspended in the air, then an external clamp could be used to impede the movement of the wheels by a desired amount by increasing friction, allowing dynamic control of wheel movement, but this also depends on the mechanical design of the robot. However, in many use cases, such as detecting slippage or getting stuck, a more useful VRR design would be to spoof the other sensors and allow the real wheel encoder data.

V. Virtual World and Rendering Challenges

Rendering in the classical computer graphics sense means generating an image from a model [25]. To render into an HMD, an additional element of *tracking* is required because rendering depends on the device's location, thus combining the *virtual* and *real* worlds. Whereas rendering on an HMD and a visual display meant to fool a robot's camera may sound similar at first thought, subtle differences must be taken into account. For an HMD and any screen meant for humans, displays have been optimized to "fool" human eyes, for which there is an accepted notion of *normal vision*. However, because of the wide variety of possible cameras, it can be difficult to design a display that would fool any camera, due to challenges explained in Section IV-A.

Rendering and virtual world models are also required for other sensors, and the additional corresponding displays may also carry unique tracking challenges. Consider human-based VR. Whereas audio rendering requires only head tracking, for haptic rendering all relevant degrees of freedom, for example hands, must also be tracked. For VRR, the challenges regarding the virtual world and tracking are similar but broader. In tracking, all parts of the robot that contain sensors to be spoofed must be tracked in the real world. Moreover, the idea of VWG and rendering to a haptic display or an IMU is a concept that must be properly defined. Whereas for a touch display the idea is simple, such as outputting 0 or 1, the possible delays in rendering (smacking) must also be considered. A display for an IMU would be more complicated because internal mechanical elements would have to be manipulated.

Finally, any knowledge of the robot's internal algorithms can be used to simplify the complexity of the VWG, as explained in Section III. However, in many modern algorithms, the robot's internal state space may be intractably large. State-space coarsening or approximation strategies should be explored to make testing these systems more feasible.

VI. A Simple Proof of Concept

We performed a simple experiment to demonstrate how a vacuum cleaner robot, Neato Botvac D5, can be fooled to think it is in a smaller passable area than it really is. The setup is shown in Fig. 4. The human is holding a piece of cardboard that acts as a haptic display to stimulate senors so that the robot believes it cannot move beyond. In Fig. 5 are two maps created by the robot, where in (a) the haptic display is used, and in (b) the robot is free to use the whole room. Interestingly, we also observed that this haptic display is insufficient to fool the range finder, because most of the room was mapped but simply deemed impassable, thereby demonstrating the concept of partial VR being useful for inducing behaviors.

VII. Conclusion

This paper has introduced the notion of virtual reality for robots (VRR), by drawing parallels between VR applied to humans (and other organisms) and ways in which a robot could be tricked by spoofing its sensors. This has led to a mathematical framework that contains general and formalized notions of displays, rendering, and VWG, which are directly interleaved with standard notions from robotics,

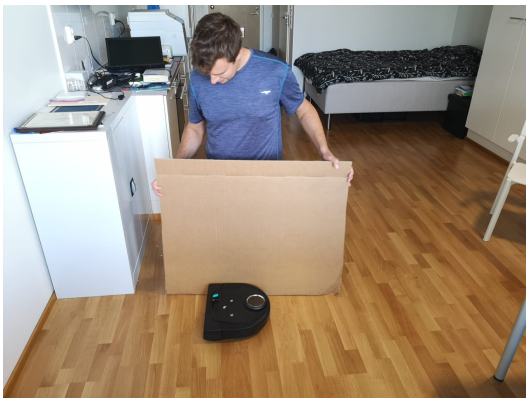
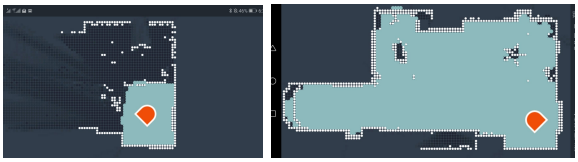


Fig. 4: A human (Markku) with cardboard, acting as a VWG, haptic renderer and display for a Neato vacuum cleaner robot.



(a) (b)

Fig. 5: Two maps created by the Neato robot. (a) is the map where the robot thinks it cannot pass any further, and (b) is the full room.

including state spaces, actions, sensor mappings, state transitions, and information states. Using this framework, we clearly see many open and interesting questions for further research by identifying both the similarities and differences between standard VR and VRR, and also for conducting extensive testing and experimentation. These are worth pursing because of the enormous potential for applications such as reliability testing, reverse engineering, security, and machine learning, as explained in Section I. For example, better anti-spoofing sensor fusion could be developed after spoofing possibilities and connections to human-based VR are properly understood, perhaps by building on a language-theoretic view of planning and filtering [22].

We expect significant future work to emerge in both VR and robotics by leveraging the parallels and distinctions made in this paper. Interesting VRR questions are inspired by human-based VR, and vice versa. For example, most "information" that would correspond to activation of photoreceptors on the retina is discarded or compressed by the ganglion, amacrine, horizontal, and bipolar cells before neural impulses are passed to the brain along the optic nerve [20]; this is analogous to the calculation of information states in a robot. Furthermore, photoreceptor density, sensitivity, and activity rates vary substantially along the retina. Such understanding has motivated techniques such as *foveated rendering* for human-based VR [8], and leads to questions such as how knowledge about sensor limitations and sensor fusion methods can be exploited to facilitate VRR solutions. Likewise, the ability to completely know the inner workings of a robot may offer insights into the improvement of human-based VR, for which sensory systems, perception, and physiological effects are not fully understood (we did not engineer ourselves!).

References

[1] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory, 2nd Ed.* Academic, London, 1995.

[2] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Xinyan. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 801–816. ACM, 2018.

[3] Carolina Cruz-Neira, Daniel J Sandin, and Thomas A DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142. Citeseer, 1993.

[4] Drew Davidson, Hao Wu, Rob Jellinek, Vikas Singh, and Thomas Ristenpart. Controlling UAVs with sensor input spoofing attacks. In *10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*, 2016.

[5] Sean P Engelson and Drew V McDermott. Error correction in mobile robot map learning. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2555–2560. IEEE, 1992.

[6] David Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990.

[7] Nesli Erdogmus and Sébastien Marcel. Spoofing 2D face recognition systems with 3D masks. In *2013 International Conference of the BIOSIG Special Interest Group (BIOSIG)*, pages 1–8. IEEE, 2013.

[8] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM Transactions on Graphics (TOG)*, 31(6):164, 2012.

[9] Winter Guerra, Ezra Tal, Varun Murali, Gilhyun Ryou, and Sertac Karaman. Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality. *arXiv preprint arXiv:1905.11377*, 2019.

[10] Christopher D Harvey, Forrest Collman, Daniel A Dombeck, and David W Tank. Intracellular dynamics of hippocampal place cells during virtual navigation. *Nature*, 461(7266):941, 2009.

[11] Andrew J Kerns, Daniel P Shepard, Jahshan A Bhatti, and Todd E Humphreys. Unmanned aircraft capture and control via gps spoofing. *Journal of Field Robotics*, 31(4):617–636, 2014.

[12] Jukka Komulainen, Abdenour Hadid, and Matti Pietikäinen. Context based face anti-spoofing. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8. IEEE, 2013.

[13] Douglas Lanman and David Luebke. Near-eye light field displays. *ACM Transactions on Graphics (TOG)*, 32(6):220, 2013.

[14] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.

[15] S. M. LaValle. *Virtual Reality*. Cambridge University Press, 2019. In press, but also available at http://vr.cs.uiuc.edu/.

[16] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov. Head tracking for the Oculus Rift. In *IEEE International Conference on Robotics and Automation*, 2014.

[17] Steven M LaValle. Sensing and filtering: A fresh perspective based on preimages and information spaces. *Foundations and Trends® in Robotics*, 1(4):253–372, 2012.

[18] Joseph J LaViola Jr. A discussion of cybersickness in virtual environments. *ACM Sigchi Bulletin*, 32(1):47–56, 2000.

[19] Pedro Lopes, Sijing You, Lung-Pan Cheng, Sebastian Marwecki, and Patrick Baudisch. Providing haptics to walls & heavy objects in virtual reality by means of electrical muscle stimulation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 1471–1482. ACM, 2017.

[20] G. Mather. *Foundations of Sensation and Perception*. Psychology Press, Hove, UK, 2008.

[21] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[22] Fatemeh Zahra Saberifar, Shervin Ghasemlou, Dylan A. Shell, and Jason M. O'Kane. Toward a language-theoretic foundation for planning and filtering. *International Journal of Robotics Research*, 38(2):236–259, March 2019.

[23] Katri Salminen, Jussi Rantala, Poika Isokoski, Marko Lehtonen, Philipp Müller, Markus Karjalainen, Jari Väliaho, Anton Kontunen, Ville Nieminen, Joni Leivo, et al. Olfactory display prototype for presenting and sensing authentic and synthetic odors. In *Proceedings of the 2018 on International Conference on Multimodal Interaction*, pages 73–77. ACM, 2018.

[24] Maria V Sanchez-Vives and Mel Slater. From presence to consciousness through virtual reality. *Nature Reviews Neuroscience*, 6(4):332, 2005.

[25] Will J Schroeder, Bill Lorensen, and Ken Martin. *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.

[26] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 445–467. Springer, 2017.

[27] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 881–896, 2015.

[28] M Srinivasan, Shaowu Zhang, M Lehrer, and T Collett. Honeybee navigation en route to the goal: visual flight control and odometry. *Journal of Experimental Biology*, 199(1):237–244, 1996.

[29] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 3–18. IEEE, 2017.

[30] Mikhail V Volkov. Synchronizing automata and the černỳ conjecture. In *International Conference on Language and Automata Theory and Applications*, pages 11–27. Springer, 2008.

[31] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 1944.

[32] G. Welch and E. Foxlin. Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6):24–28, 2002.

[33] Peter Wild, Petru Radu, Lulu Chen, and James Ferryman. Robust multimodal face and fingerprint fusion in the presence of spoofing attacks. *Pattern Recognition*, 50:17–25, 2016.

[34] Matthias Wittlinger, Rüdiger Wehner, and Harald Wolf. The ant odometer: stepping on stilts and stumps. *Science*, 312(5782):1965–1967, 2006.

[35] Zhizheng Wu, Nicholas Evans, Tomi Kinnunen, Junichi Yamagishi, Federico Alegre, and Haizhou Li. Spoofing and countermeasures for speaker verification: A survey. *speech communication*, 66:130–153, 2015.

[36] Jiangfan Zhang, Rick S Blum, Lance M Kaplan, and Xuanxuan Lu. Functional forms of optimum spoofing attacks for vector parameter estimation in quantized sensor networks. *IEEE Transactions on Signal Processing*, 65(3):705–720, 2016.

[37] Zhongshun Zhang, Lifeng Zhou, and Pratap Tokekar. Strategies to design signals to spoof kalman filter. In *2018 Annual American Control Conference (ACC)*, pages 5837–5842. IEEE, 2018.