Sensor Lattices: Structures for Comparing Information Feedback

Steven M. LaValle
steven.lavalle@oulu.fi
Center for Ubiquitous Computing
Faculty of Information Technology and Electrical Engineering
University of Oulu, Finland

***Abstract*— This paper addresses the sensing uncertainty associated with the many-to-one mapping from a physical state space onto a sensor observation space. By studying preimages of this mapping for each sensor, a notion of *sensor dominance* is introduced, which enables interchangeability of sensors and a clearer understanding of their tradeoffs. The notion of a *sensor lattice* is also introduced, in which all possible sensor models are arranged into a hierarchy that indicates their power and gives insights into the construction of filters over time and space. This provides a systematic way to compare and characterize information feedback in robotic systems, in terms of their level of ambiguity with regard to state estimation.**

I. Introduction

We rely heavily on sensing to achieve robotic tasks such as navigation, patrolling, searching, pursuit-evasion, and coverage. Every sensor can be viewed as a mapping from an enormously complicated physical world onto a set of observations. We do not attempt to measure *everything* about the world; therefore, we expect sensing uncertainty to remain, even when sensors are performing perfectly without noise or disturbances. Starting from the task, one usually designs a filter that aggregates sensor observations and provides feedback during execution. What sensors are sufficient for the task? What sensors are necessary? When are sensors interchangeable? If they are, then what are the tradeoffs in terms of cost, power consumption, memory, reliability, robustness, and computation time?

This paper takes a step toward understanding these issues by the following chain of reasoning:

1) We introduce the notion of a *virtual sensor* model to precisely define the information that the sensor is supposed to measure. This is expressed as a many-to-one mapping from a state space of world states to sensor observations. Most importantly, it does not depend on the particular *physical sensor* implementation.
2) The preimages of the many-to-one mapping yield a partition of the state space that captures the "resolution" at which the world can be sensed. By comparing these partitions, we introduce a simple notion of *sensor dominance*, which indicates whether one sensor is able to simulate another by using its own observations to predict the observations of the other. This implies that either could be used for the same task.
3) The set of *all* virtual sensor models are arranged into a lattice of partitions (see Figure 1), called a *sensor lattice*, which places the most powerful sensors at the top and the most useless sensors at the bottom. All other sensors appear somewhere in between. The lattice gives insights into combining sensors and constructing filters over space and/or time.

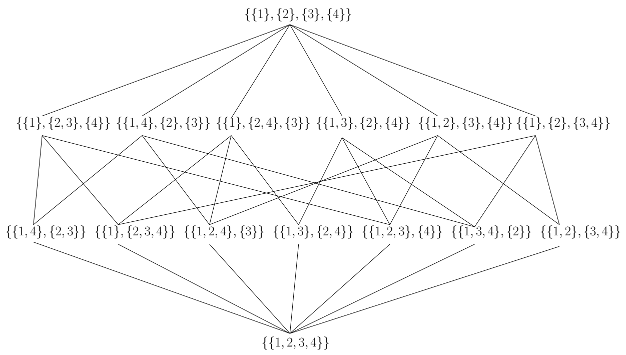


Fig. 1. The partition lattice for a (nearly trivial) four-element set. The sensor lattice considers partitions over the entire state space and places best and worst sensors are at the top and bottom, respectively.

These three steps correspond to Sections II, III, and IV, respectively. Section V generalizes the models and concepts, and then Section VI and extends the sensor dominance and lattice ideas to filters.

The ideas in this paper are inspired by several related works. Sensors were viewed as partitions of state space in [4], [6]. The notion of simulating one sensor using another appears in the *information invariants* framework of Donald [3]. The comparison of entire robotic systems through the use of simulation appears in [13]; one implication of the current paper is that if one sensor dominates another in the sense defined in Section III, then the overall robotic system that uses it will be at least as powerful in the sense defined in [13]. The sensing uncertainty considered in this paper is closely related to information structures from dynamic game theory [1], [12]. The framework presented in this paper can be combined with recent research on minimalist filtering and planning, to establish relationships between various systems and their relative computational hardness [14].

II. Physical vs. Virtual Sensors

Rather than compare physical sensors themselves, we compare mathematical models of sensors based on the information that they are designed to measure. In this way, any analysis that follows is invariant with respect to the particular physical implementation.

The mathematical model starts with a precise description of the physical world. The set of all possibilities for this is called the *state space*, X. The description includes only information that is relevant to whatever tasks are to be accomplished and whatever phenomena are to be measured. A simplified state space is used here for the purposes of clearly explaining the concepts; much more general spaces are presented in [10].

A. Physical state space

Suppose that a single robot is placed into a bounded, closed planar *environment* $E \subseteq \mathbb{R}^2$ that may contain polygonal holes. To avoid complications of obstacle boundaries in configuration space, assume the robot is a point. The robot's configuration includes both position $(q_x, q_y) \in E$ and orientation $q_\theta \in S^1$ in which $S^1 = [0, 2\pi]$ with 0 and 2π identified (made equivalent).

If the environment is fixed and known, then the state space is defined as $X = E \times S^1$. However, for many problems, a precise map of the environment is not given (hence, the popular SLAM problem). In this case, only the *map family* $\mathcal{E}$ is given. For a concrete example, suppose in this section that $\mathcal{E}$ is the set of all connected, bounded polygonal regions (possibly with holes) in $\mathbb{R}^2$. The state space X then becomes the set of all pairs (q, E) in which $q = (q_x, q_y, q_\theta)$, $(q_x, q_y) \in E$ and $E \in \mathcal{E}$.

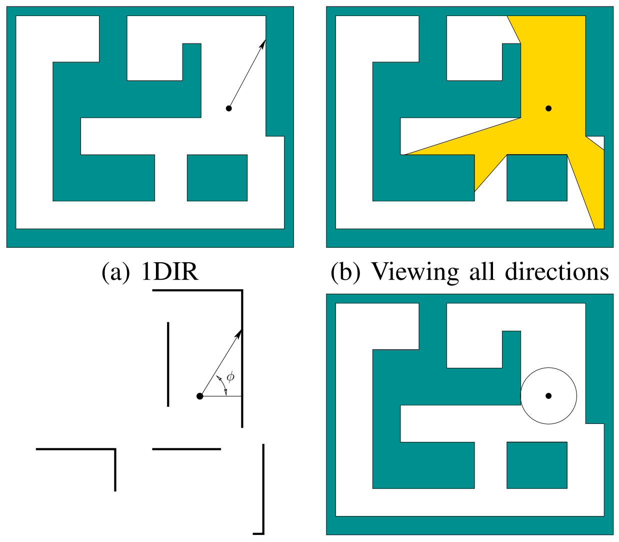
B. The sensor mapping

Now that the physical state space has been defined, the mathematical model of a sensor is a mapping from physical states to possible sensor readings. Let X be any state space. Let Y denote the *observation space*, which is the set of all possible sensor observations. A *virtual sensor* is defined by a function

$$h : X \to Y, \tag{1}$$

called the *sensor mapping*. When $x \in X$, the sensor instantaneously observes $y = h(x) \in Y$.

What does an observation y tell us about the external, physical state? To understand this, we should think about



(a) 1DIR (b) Viewing all directions (c) OMNI (d) NWAL

Fig. 2. (a) The 1DIR virtual sensor provides a single distance measurements, from the robot to the wall, in the direction that the robot is facing. (b) Consider instead sensing in all directions. (c) The resulting observation of OMNI is a distance function parametrized by viewing angle. (d) The NWAL virtual sensor observes the distance to the nearest wall.

all states $x \in X$ that could have produced the observation. For a given sensor mapping h this is defined as

$$h^{-1}(y) = \{x \in X \mid y = h(x)\}, \tag{2}$$

and is called the *preimage* of y. If h were invertible, then h^{-1} would represent the inverse; however, because our sensor models are usually many-to-one mappings, $h^{-1}(y)$ is a subset of X, which yields all x that map to y.

C. Depth sensors

Consider the collection of subsets of X obtained by forming $h^{-1}(y)$ for every $y \in Y$. These sets are disjoint because a state x cannot produce multiple observations. Since h is a function on all of X, the collection of subsets forms a partition of X. For a given sensor mapping h, the corresponding partition is denoted as $\Pi(h)$.

The connection between h and $\Pi(h)$ is fundamental to sensing. The sets in $\Pi(h)$ can be viewed as equivalence classes. For any $x, x' \in X$, equivalence implies that $h(x) = h(x')$. These states are indistinguishable when using the sensor. In an intuitive way, $\Pi(h)$ gives the sensor's sensitivity to states, or the "resolution" at which the state can be observed. The equivalence classes are the most basic source of uncertainty associated with a sensor.

Consider measuring depth information from the perspective of the robot. Some sensor models are given in Figure 3. In every case, a sensor mapping $h : X \to Y$ can be formally defined (see [10]). Each state $x =$

1DIR:	Distance to the wall in the direction that the robot is facing.
OMNI:	Distances to walls in all directions.
NWAL:	Distance to the nearest wall.
PROX(ϵ):	Detection of whether the wall is within distance ϵ.
CON:	Detection of robot contact with the wall.
CONE:	Distances to walls in a range of directions from $-\pi/4$ to $\pi/4$.
MAXD(ϵ_2):	Distances to walls in the range of directions, but reports no value when the distance is larger than ϵ_2.
MNMX(ϵ_1, ϵ_2):	Distances to walls in the range of directions, but reports no value when the distance is less then ϵ_1 or larger than ϵ_2.
OCC(ϵ_1, ϵ_2):	Detection of whether there is a wall within the range of directions and between distances ϵ_1 and ϵ_2.
GAPS:	The angular locations and lengths of the depth discontinuities.
AGAP:	The angular locations of the gaps without depths.
NGAP:	The number of gaps.
STAR:	Detection of the center of a star-shaped region.

Fig. 3. Various kinds of virtual sensors that measure depth and will be used for comparison purposes.

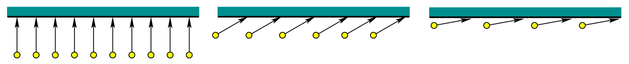


Fig. 4. The preimage for 1DIR is a two-dimensional subset of $E \times S^1$, assuming E is given. Shown here are several robot configurations within the same preimage.

$(q_x, q_y, q_\theta, E) \in X$ provides sufficient information to construct any of these virtual sensors. The 1DIR virtual sensor, shown in Figure 2(a) provides a single distance measurement in the forward direction and could be implemented physically using a sonar or a single laser/camera combination. What do the preimages look like? Suppose E is given, so that X becomes $E \times S^1$, the C-space of the robot. Almost every preimage $h^{-1}(y)$ is a two-dimensional subset of X; see Figure 4. An interesting task is to characterize the preimages for all $x \in X$ (closely related to motion planning for a ladder robot [8], [11]).

Whereas 1DIR provides the depth in one direction, OMNI provides depth measurements in all directions; see Figures 4(b) and (c). The observation can be considered as a function $y : S^1 \rightarrow [0, \infty)$ and could be implemented by SICK-like laser scanners that provide dense, accurate depth measurements.[1] Using OMNI, the preimages are much smaller. If E is given, we might able to uniquely localize the robot from a single observation. If the environment contains symmetries, then the preimage

[1]In practice, of course, depth is obtained only at a finite number of directions; however, a useful mathematical model is that a continuum of depths is observed.

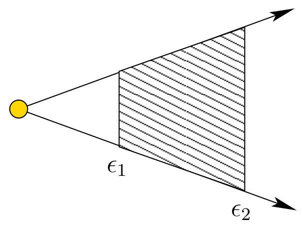


Fig. 5. The sensing range is limited for the MNMX virtual sensor, which could correspond physically camera-based depth systems, such as the KINECT

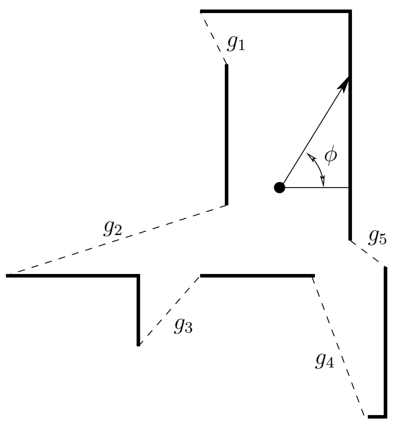


Fig. 6. A gap sensor example: Five discontinuities in depth are observed.

$h^{-1}(y)$ is a set of possible robot configurations (which are the standard hypotheses obtained in localization [5], [9], [15]). By taking the minimum distance over all directions, NWAL is obtained, which observes the distance to the nearest wall; see Figure 2(d). A binary-valued variant, PROX(ϵ), can be construced in which $y = 1$ means that the wall is no more than distance ϵ away and $y = 0$ means it is further than ϵ. Two preimages are obtained. For the special case of $\epsilon = 0$, we obtain CON, for which $y = 1$ if the robot position is on the boundary and $y = 0$ otherwise. In this case, the two preimages are clear: The boundary of E (for $y = 1$) and the interior of E (for $y = 0$). The sensor could be implemented using a contact switch.

For camera-based distance measurements, limiting the angular range may be appropriate, to obtain CONE, for which $y : [\phi_{min}, \phi_{max}] \rightarrow [0, \infty)$. MAXD($\epsilon_2$) is a variant of CONE that is depth limited: If the wall is at least ϵ_2 units away in some direction, it does not return the precise distance (instead returning a special symbol for that direction). MNMX(ϵ_1, ϵ_2) is a variant of MAXD(ϵ_2) that similarly cannot measure depths that are within distance ϵ_1 of the robot; see Figure 5. OCC(ϵ_1, ϵ_2) is an *occupancy detector* for which $y = 1$ means that a wall lies within the detection zone and $y = 0$ means it does not. In this case, there are only two preimages.

Gap sensors form another family of virtual sensor models. Note in Figure 2(c) that several discontinuities may appear in the depth function provided by OMNI. The virtual sensor GAPS indicates the angles in S^1 at which these occur (if there are any) and the length of each gap (the amount that the distance jumps). AGAP is similar to GAPS, but provides only the angles at which gaps occur. NGAP is weaker than AGAP and indicates only the number of gaps. Finally, STAR is a binary-valued sensor that yields $y = 1$ if there are any gaps and $y = 0$ otherwise. Note that if $y = 1$, then it is immediately known that E is star-shaped.

By considering preimages of $h : X \to Y$, every virtual sensor above partitions X into sets that produce the same observation. If E is given, then each preimage $h^{-1}(y)$ is a set of robot positions and orientations. If E is unknown, then the preimages are much larger. For example, if $y = 1$ for the CON sensor, then it is known only that "the robot is touching the wall in some environment." For OMNI, $h^{-1}(y)$ indicates all ways that the environment could exist beyond the field of view of the sensor.

The depth sensors presented in this section are just one family among many possibilities. They were chosen to provide clear illustrations in coming sections. Other families of sensors that fit into this framework include detection sensors [10], relational sensors [7], and field sensors [10].

III. Simulation Between Virtual Sensors

After seeing the virtual sensors in Figure 3, it is clear that they could be physically implemented in various ways. Which is most advantageous or appropriate in a particular robotic system? A key step in answering this question is to understand when it is possible for one virtual sensor to simulate another and to know the cost involved. Sensor simulation was considered earlier in [3]; here it will be defined differently.

Suppose the state space X is predetermined and fixed. Let $h_1 : X \to Y_1$ and $h_2 : X \to Y_2$ be any two virtual sensor models. A partition Π_1 is called a *refinement* of Π_2 if every set in Π_1 is a subset of some set in Π_2. Let $\Pi(h)$ denote the partition of X induced by the preimages of h. We say that h_1 *dominates* h_2 if and only if $\Pi(h_1)$ is a refinement of $\Pi(h_2)$. This is denoted as $h_1 \succeq h_2$.

For some state $x \in X$, imagine receiving $y_1 = h_1(x)$ and $y_2 = h_2(x)$. If $h_1 \succeq h_2$, then $h_1^{-1}(y_1) \subseteq h_2^{-1}(y_2) \subseteq X$. This clearly means that h_1 provides at least as much information about x as h_2 does. Furthermore, using y_1, we could infer what observation y_2 would be produced by h_2. Why? Since $\Pi(h_1)$ is a refinement of $\Pi(h_2)$, then every $x \in h_1^{-1}(y_1)$ must produce the same observation $y_2 = h_2(x)$. This implies that there exists a function $g : Y_1 \to Y_2$ such that $h_2(x) = g(h_1(x))$, written as $h_2 = g \circ h_1$. Hence, y_1 contains enough information so that h_1 could simulate h_2. The simulation is accomplished by g. Here is a diagram of the functions:

$$\begin{tikzcd} X \arrow[rr, "h_2"] \arrow[dr, "h_1"'] & & Y_2 \\ & Y_1 \arrow[ur, "g"'] & \end{tikzcd} . \quad (3)$$

Note that if $\Pi(h_1) = \Pi(h_2)$, then h_1 and h_2 can simulate each other.

One immediate conclusion is that if a robot system can accomplish a task using h_2 and $h_1 \succeq h_2$ with g efficiently computable, then h_1 could be swapped for h_2 while still accomplishing the task. The contrapositive is also interesting: If the task cannot be solved using h_1, then it certainly cannot be solved using h_2 because the preimages are larger.

From the collection of sensors in Figure 3, it is tempting to always prefer OMNI because it dominates all of the other sensors. However, this is not a good idea in most cases. A physical implementation of OMNI might cost thousands of dollars and it might draw significant energy from the system. Furthermore, the computation of g might be time consuming and possibly not robust due to observation errors. Therefore, a direct implementation of a simpler virtual sensor is preferred. The method: 1) Start by considering the task; 2) choose the weakest virtual sensor that is sufficient for solving it; 3) obtain a physical sensor that robustly implements the virtual sensor. Neglecting the computational cost of g is similar to the construction of an oracle machine in complexity theory to obtain relativizations [2]. For some simulations, note that g may be intractable to compute or not even computable.

IV. The Lattice of All Sensors

For a fixed state space X, the dominance relation $\succeq$ induces a partial ordering on *all* possible virtual sensor models of the form $h : X \to Y$. Figure 7 shows how the depth sensors of Figure 3 are related.

What happens as we include more and more sensors, and continue to extend the tree? There may be multiple paths from the root to lower nodes, leading to a directed acyclic graph. Continuing further, note that Y does not need to be fixed, meaning we could take any set Y and define any mapping $h : X \to Y$. Consider defining an equivalence relation $\sim$ on the enormous collection of all possible virtual sensors for a fixed state space X. We say that $h_1 \sim h_2$ if and only if $\Pi(h_1) = \Pi(h_2)$. If we no longer pay attention to the particular h and Y, but only consider the induced partition of X, then we imagine that a virtual sensor *is* a partition of X. Continuing in this way, the set of all possible virtual sensors is the set of all partitions of X.

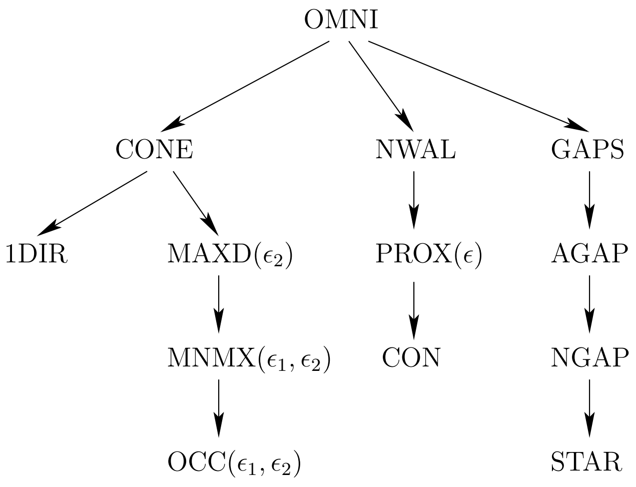


Fig. 7. The virtual sensor models in Figure 3 are compared based on refinements of the partitions they induce over X. Models higher in the tree induce finer partitions. A lower sensor model can be "simulated" by any model along the path from the root of the tree to itself.

The relationship between sensors in terms of dominance then leads to the well-known idea of a *partition lattice*, depicted in Figure 1 for the set $X = \{1, 2, 3, 4\}$. Recall that a lattice is a set together with a partial order relation $\succeq$ for which every pair of elements has a *least upper bound (lub)* and a *greatest lower bound (glb)*. Starting with any set, the set of all partitions forms a lattice. The relation $\succeq$ is defined using refinements of partitions: $\pi_1 \succeq \pi_2$ if and only if π_1 is a refinement of π_2.

Now observe that for any state space X, all possible sensors fit nicely into the partition lattice of X. Furthermore, $\succeq$ indicates precisely when one sensor dominates another. The tree depicted in Figure 7 is embedded in this lattice. The top of the lattice corresponds to *bijective* sensors, for which $h : X \to Y$ is a bijection and $\Pi(h)$ is the collection of all singleton sets. It is the finest partition possible because the sensor provides enough information to reconstruct x, assuming $h^{-1}(y)$ is computable. The bottom of the lattice corresponds to *dummy* sensors, for which $\Pi(h) = \{X\}$. An example dummy sensor is $h : X \to \{0\}$, which outputs $y = 0$, regardless of the state. All "interesting" sensors lie between the dummy and bijective sensors in the lattice.

The glb and lub have interesting interpretations in the sensor lattice. Suppose that for two partitions, $\Pi(h_1)$ and $\Pi(h_2)$, neither is a refinement of the other. Let $\Pi(h_{glb})$ and $\Pi(h_{lub})$ be the glb and lub, respectively, of h_1 and h_2. The glb $\Pi(h_{glb})$ is the partition obtained by "overlaying" the partitions $\Pi(h_1)$ and $\Pi(h_2)$. Take any state $x \in X$. Let y_1, y_2, and y_{glb}, be the observations obtained by applying h_1, h_2, h_{glb}, respectively. An element of $\Pi(h_{glb})$ is obtained by intersecting preimages, $h_1^{-1}(y_1) \cap h_2^{-1}(y_2)$. There is a straightforward way to construct some representative h_{glb} from h_1 and h_2. Let $Y_{glb} = Y_1 \times Y_2$ and $h_{glb} : X \to Y_{glb}$ be defined as $y_{glb} = (y_1, y_2) = (h_1(x), h_2(x))$. This means that both h_1 and h_2 are combined to produce a single sensor. The partition $\Pi(h_{glb})$ is just the common refinement. This process actually corresponds to spatial filtering in Section VI-A.

The lub, $\Pi(h_{lub})$, is the opposite of $\Pi(h_{glb})$ in some sense. The partition $\Pi(h_{lub})$ is as coarse as it needs to be so that every element contains the complete preimages of h_1 and h_2. Again starting from any $x \in X$, $\Pi(h_{lub})$ is the finest partition for which $h^{-1}(y_1) \cup h^{-1}(y_2) \subseteq h^{-1}(y_{lub})$.

One way to "visualize" these relationships is to imagine the case in which $X = Y = \mathbb{R}^3$ and restrict the set of all sensor mappings to be only linear ones, $y = Cx$. If C has rank 2, then the preimages $h^{-1}(y)$ are lines in $\mathbb{R}^3$. Consider two linear sensors, with matrices C_1 and C_2 having rank 2. The glb produces preimages that are the intersection of two lines. The lines must always intersect because both preimages are observations of same state $x \in \mathbb{R}^3$. If the combined 3 by 6 matrix, $[C_1\ C_2]$, has rank three, then all preimages will be points, and the glb is a bijective sensor. The preimages for the lub in this case is the set of all planes in $\mathbb{R}^3$. Each plane is obtained by taking the union of the preimages, which forms a pair of intersecting lines.

A well studied algebraic structure exists over the partition lattice with respect to the glb and lub operations. The algebra satisfies axiomatic identities for commutativity, associativity, absorption, and idempotency. These immediately apply to the space of all virtual sensors to yield an algebra of sensor models. This may provide further insight into the selection of sensors and the design of filters. It will be seen in Section VI that filter design is based heavily on the glb.

V. Generalizations and Extensions

The state spaces and virtual sensors described in Sections II to IV were chosen facilitate explanations of sensor dominance and the sensor lattice. The concepts, however, extend well beyond these examples. This section briefly indicates some of these extensions. More details, and additional concepts such as sensor noise and other sensor families, appear in [10].

A. More bodies in the environment

Consider placing various kinds of *bodies* into an environment E, which may or may not contain robots. A body B *occupies* a subset of E and can be transformed using its own configuration parameters. For example, a body could be a point that is transformed by (q_x, q_y) parameters or a rectangle that is transformed by (q_x, q_y, q_θ)

parameters. We can write $B(q_x, q_y, q_\theta) \subset E$ to indicate the set of points occupied by B when at configuration (q_x, q_y, q_θ). In applications, bodies may have many alternative interpretations, such as robots, people, landmarks, obstacles, objects, pebbles, targets, evaders, treasures, and transmitters.

Each body has predefined properties that become critical for sensing and actuation tasks: 1) What are its *motion capabilities*? It could be static, predictably moving, or unpredictably moving. 2) Can it be *distinguished* from other bodies? Each could be assigned a possibly nonunique label, allowing the bodies to be anywhere along the range from completely indistinguishable to partially distinguishable (teams) to uniquely identifiable. 3) How does it *interact* with other bodies? It could obstruct sensors, obstruct motions of others, or even be manipulated by others.

A rich variety of filtering and planning tasks can be formulated in this way, such as target tracking, monitoring, counting bodies, pursuit-evasion, and mapping. The state space X is then expanded to account for their configurations, and the sensor mappings are once again expressed as $h : X \to Y$. Sensor dominance and the sensor lattice follow accordingly.

B. Sensors over state-time space

Of course the world is not static. Let T refer to an interval of time, such as $T = [0, \infty)$. Using any physical state space X, we define *state-time space* $Z = X \times T$. Sensors that provide information about both state and time are defined as $h : Z \to Y$. Preimages in Z are defined as

$$h^{-1}(y) = \{(x, t) \in Z \mid y = h(x, t)\}, \qquad (4)$$

which means that h can now be considered as a partition of state-time space, Z. As an example, a sensor observation might imply that a person entered the room within the past hour. Sensor dominance from Section III and the sensor lattice from Section IV extend directly by replacing X with Z.

C. Sensors over state-trajectory space

Many sensors depend on a previous history of states. Examples are odometry and delayed measurements. This results in a sensor definition that subsumes both $h : X \to Y$ and $h : Z \to Y$. Let a *state trajectory* up to time t be denoted as $\tilde{x} : [0, t] \to X$. The set of all possible trajectories for any possible $t \in T$ is called the *trajectory space* and is denoted by $\tilde{X}$. For a *history-based sensor*, the sensor mapping is $h : \tilde{X} \to Y$. In this case, a given state trajectory $\tilde{x}$ produces an observation $y = h(\tilde{x})$. The preimage,

$$h^{-1}(y) = \{\tilde{x} \in \tilde{X} \mid y = h(\tilde{x})\}, \qquad (5)$$

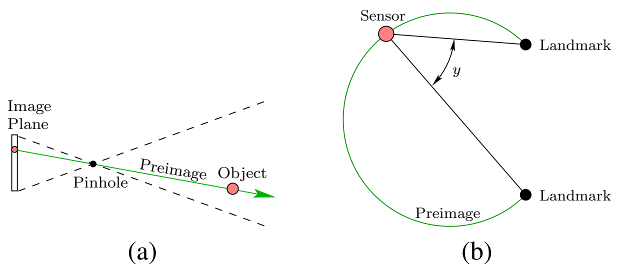


Fig. 8. (a) The preimage is a ray for a pinhole camera model. Using two calibrated cameras, the rays are intersected to locate the object. (b) Ancient triangulation is based on holding the viewing angle fixed. This leads to circular preimages.

yields the set of possible trajectories in $\tilde{X}$ that produce the same y. The preimages induce a partition of $\tilde{X}$, and all history-based sensors can be arranged into a sensor lattice over $\tilde{X}$.

VI. From Sensors to Filters

Filtering is concerned with combining information from multiple sensor observations. If these occur at the same instant or the state does not vary over time, then filtering can be viewed as triangulation, which is covered in Section VI-A. The more general, time-varying case is addressed in Section VI-B.

A. Time-independent filtering

Consider any n sensor mappings $h_i : X \to Y_i$ for i from 1 to n. If each produces an observation $y_i \in Y_i$ at some common instant, then what are the possible states? The following problems are equivalent: 1) The observations were obtained at the same instant and we want the possible states at that same instant; 2) the state does not vary over time, but the observations might occur at various points in time. Both of these are referred to as a *time-independent* or *spatial* filtering problem.

A filter can be considered as a generalization of the ancient principle of measurement triangulation. The *triangulation* of the observations is denoted by $\triangle(y_1, \ldots, y_n)$ and is the intersection of preimages (2) of each sensor observation, to obtain:

$$\triangle(y_1, \ldots, y_n) = h_1^{-1}(y_1) \cap h_2^{-1}(y_2) \cap \cdots \cap h_n^{-1}(y_n), \quad (6)$$

which is a subset of X.

Many familiar examples of spatial filtering can be modeled with (6). Figure 8(a) shows a small object appearing in an image under the pinhole camera model. Let $X = \mathbb{R}^2$. The preimage $h^{-1}(y)$ is a ray that extends outward from the pinhole and through the object in $\mathbb{R}^3$. By placing two calibrated cameras into a scene and observing the same object, (6) intersects the rays to uniquely determine the object location. This is classical stereo vision. Figure 8(b) shows classical triangulation, which is a technique used for thousands of years by ancient Greeks, Egyptians, and Chinese. The sensor mapping provides the angle between a pair of landmarks, as observed from the sensor location. The preimage is a circular arc. For three landmarks, we can observe two angles, one per pair of landmarks. The sensor location in $\mathbb{R}^2$ is uniquely determined by (6), which is the intersection of circular arcs. For another example, suppose that radio towers emit signals and the distance to each tower can be measured, possibly by using *time of arrival* (TOA). The preimage for one distance observation is a circle (or sphere in $\mathbb{R}^3$), and (6) yields the classical *trilateration* localization technique. Suppose that instead of TOA, only the *time difference of arrival* (TDOA) is available. In that case, the preimages are hyperbolic curves (or surfaces in $\mathbb{R}^3$), and (6) produces the method of *hyperbolic positioning*, which was used in the Decca Navigator System in World War II to locate ships.

Note that these filtering methods are idealized. If measurement noise or disturbances occur, then it is well known in practice that many more measurements are needed. Rather than finding a common intersection, an error cost can be formulated and the best state estimate is often found by least squares minimization of the sum-of-squares error. However, the filter is originally designed based on the ideal intersection of preimages.

Now consider the implications of Sections III and IV. Using triangulation (6), imagine replacing one of the sensors, say h_i, with another, say h'_i, that dominates it. Since $h^{-1}(y'_i) \subset h^{-1}(y_i)$, (6) produces a filter that is at least as good as the original. The set of possible sets using h'_i must be a subset of the possible states obtained from using h_i. Regarding the sensor lattice, interpret (6) as the construction of a "super" sensor $h_\triangle : X \to Y$ that obtains a smaller preimage due to the intersection of original preimages. A partition $\Pi(h_\triangle)$ of X is induced and hence this must live in the sensor lattice over X. The methods described above all yield unique states from the observations (assuming general position of landmarks), and therefore $h_\triangle$ is equivalent to the bijective sensor in these cases. Furthermore, for any pair of sensors, the resulting super sensor $h_\triangle$ corresponds to the greatest lower bound (glb) in the sensor lattice. It is therefore interesting to consider spatial filter design as a trip through the sensor lattice, attempting to reach super sensors that are higher than original sensors.

B. Temporal filtering

In the time-varying case, we once again consider the time interval T, the state-time space $Z = X \times T$, and the trajectory space $\tilde{X}$. It will be helpful to distinguish between complete and partial trajectories. Suppose $T = [0, t_f]$, in which t_f is the final time. A *complete* trajectory

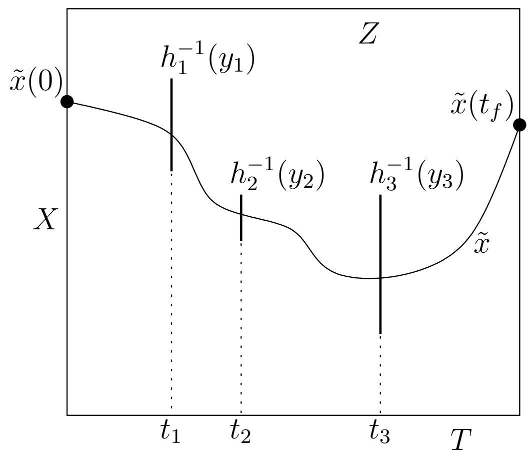


Fig. 9. Each complete trajectory that is consistent with the observations must pass through the preimage of each observation, y_1, y_2, and y_3.

is of the form $\tilde{x} : T \to X$ and a *partial* trajectory is of the form $\tilde{x} : [0, t] \to X$ for any $t \in [0, t_f)$. (We could alternatively allow unbounded trajectories: Define $T = [0, \infty)$ and allow any $t \in T$ to define a partial trajectory.) Let $\tilde{X}_c$ denote the set of complete trajectories.

Suppose that each sensor is of the form $h_i : Z \to Y$ and $y_i = h_i(x_i, t_i) = (y'_i, t_i)$, in which $y'_i = h'_i(x)$ is a standard sensor mapping from Section II. Here x_i is the state at time $t_i \in T$, which can be written as $x_i = \tilde{x}(t_i)$ for the trajectory $\tilde{x} : T \to X$ in $\tilde{X}$. Essentially, h_i is a sensor that indicates the precise time of each observation.

Suppose that n observations, y_1, …, y_n are obtained, each of which is generated by $y_i = h_i(\tilde{x}(t_i), t_i)$. What is the set of possible complete trajectories? Figure 9 illustrates the calculations. Each y_i introduces a thin vertical window $h^{-1}(y_i) \subset Z$ at time t_i through which a trajectory must travel to account for the observation. The set of possible complete trajectories are those that travel through all n windows.

For a single observation $y_i = h_i(x_i, t_i)$, the set of possible complete trajectories is

$$\tilde{h}_i^{-1}(y_i) = \{\tilde{x} \in \tilde{X}_c \mid \tilde{x}(t_i) = h_i(x_i, t_i)\}, \qquad (7)$$

which is similar to (2). For the set of n observations, we calculate the subset of $\tilde{X}_c$ of possible complete trajectories by

$$\tilde{\triangle}(y_1, \ldots, y_n) = \tilde{h}_1^{-1}(y_1) \cap \tilde{h}_2^{-1}(y_2) \cap \cdots \cap \tilde{h}_n^{-1}(y_n), \quad (8)$$

which can be considered as a form of triangulation, similar to (6).

Since (7) and (8) appear to be preimages and triangulation, they can be once again viewed in a sensor lattice. However, the lattice is over $\tilde{X}_c$, the set of all complete trajectories. Each observation partitions $\tilde{X}_c$ into class of trajectories that could account for it. The triangulation (8) applied to a pair of observations calculates a refinement of partitions, which is a lub in the lattice. Thus, this form of temporal filtering can be considered as traveling in a sensor lattice. In most settings, a unique trajectory will not be discovered due to time gaps in which no observations are made. Hence, the top of the lattice is not reached.

Another filtering problem is to determine the possible states at some time $t \in T$, assuming $t > t_i$ for all i from 1 to n. This is expressed as

$$\{x \in X \mid \exists \tilde{x} \in \tilde{\triangle}(y_1, \ldots, y_n) \text{ with } \tilde{x}(t) = x\}. \quad (9)$$

In many forms of temporal filtering, actions u, selected from some set U, are additionally given [16]. These are applied to a state transition equation to further constrain the possible trajectories. Using our formulation, this requires restricting $\tilde{X}_c$ so that only trajectories that satisfy the observed actions are included. Once this is handled, the preimage and triangulation concepts remain unchanged.

What was neglected above is the more complicated case in which perfect time stamps t_i are not available with each y_i. In this case, the vertical bars at precise times in Figure 9 become "blobs" or more general regions in Z because the times are unknown. For an observation $y \in Y$, any preimage in Z and corresponding partition of Z is possible. The temporal filter then uses the observations to construct a refinement of partitions over Z. This might not, however, induce a refinement of partitions over $\tilde{X}_c$ because one trajectory could be explained by multiple observation sets y_1, …, y_n (this was not the case when perfect time stamps were given).

VII. Conclusions

We introduced the notion of a sensor lattice, which provides a new way understand the relative power of sensors and filters that are constructed from them. Within a filter or robotic system that uses the filter, a sensor can clearly be replaced with another if sensor domination is achieved as defined in Section III and the resulting simulation is efficiently computable. A clear interpretation of the uncertainty due to the many-to-one mapping of sensors is provided by the place in the sensor lattice. Filters attempt to travel up in the lattice by improving information.

Many important and interesting open questions remain. The topic of sensor noise is somewhat orthogonal to this paper because we are concerned with what is possible or impossible, and we want to understand how sensors are related from a design perspective (when the system is conceived). Understanding implementation tradoffs and extending sensor lattices to account for noise and disturbances remains for future work. In another direction, the computational decidability and complexity of the various constructions in the paper have not been determined in particular contexts. How much does it cost to do simulate one sensor with another? What are the other tradeoffs? Computing the preimages explicitly (recall Figure 4) for various sensors may be useful in the design of novel filters, which require preimage intersections (recall (6)). Substantial work remains in identifying useful families of virtual sensors. What virtual sensors lead to efficient algorithms for filtering, which can then be used in feedback control laws?

Acknowledgments: This work was supported in part by NSF grants 0904501 (IIS Robotics) and 1035345 (Cyberphysical Systems), DARPA SToMP grant HR0011-05-1-0008, MURI/ONR grant N00014-09-1-1052.

References

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

[2] T. P. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[3] B. R. Donald. On information invariants in robotics. *Artificial Intelligence Journal*, 72:217–304, 1995.

[4] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 190–197, 1991.

[5] G. Dudek, K. Romanik, and S. Whitesides. Global localization: Localizing a robot with minimal travel. *SIAM Journal on Computing*, 27(2):583–604, April 1998.

[6] M. A. Erdmann. Understanding action and sensing by designing action-based sensors. *International Journal of Robotics Research*, 14(5):483–509, 1995.

[7] L. Guibas. Sensing, tracking, and reasoning with relations. *IEEE Signal Processing Magazine*, 19(2):73–85, 2002.

[8] Y. Ke and J. O'Rourke. Lower bounds on moving a ladder in two and three dimensions. *Discrete and Computational Geometry*, 3:197–217, 1988.

[9] S. Koenig, A. Mudgal, and C. Tovey. An approximation algorithm for the robot localization problem. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[10] S. M. LaValle. *Sensing and Filtering: A Fresh Perspective Based on Preimages and Information Spaces*, volume 1:4 of *Foundations and Trends in Robotics Series*. Now Publishers, Delft, The Netherlands, 2012.

[11] D. Leven and M. Sharir. An efficient and simple motion planning algorithm for a ladder moving in a 2-dimensional space amidst polygonal barriers. *Journal of Algorithms*, 8:192–215, 1987.

[12] J. R. Marden and J. S. Shamma. Game theory and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:105–134, 2018.

[13] J. M. O'Kane and S. M. LaValle. On comparing the power of robots. *International Journal of Robotics Research*, 27(1):5–23, 2008.

[14] J. M. O'Kane and D. A. Shell. Concise planning and filtering: hardness and algorithms. *IEEE Transactions on Automation Science and Engineering*, 14(4):1666–1681, 2017.

[15] M. Rao, G. Dudek, and S. Whitesides. Randomized algorithms for minimum distance localization. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2004.

[16] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.