Bayesian Information Gain to Design Interaction

Wanyu Liu, Olivier Rioul, Michel Beaudouin-Lafon

▶ **To cite this version:**

Wanyu Liu, Olivier Rioul, Michel Beaudouin-Lafon. Bayesian Information Gain to Design Interaction. Nikola Banovic, Per Ola Kristensson, Antti Oulasvirta, and John H. Williamson. Bayesian Methods for Interaction Design, Cambridge University Press, 2022. hal-03323514

HAL Id: hal-03323514

https://telecom-paris.hal.science/hal-03323514v1

Submitted on 21 Aug 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Bayesian Information Gain to Design Interaction

Wanyu Liu, Olivier Rioul, and Michel Beaudouin-Lafon

Abstract

This chapter discusses a perspective on designing interaction by quantifying information that reduces the computer's uncertainty about the user's goal. We begin with how to quantify *uncertainty* and *information* using Shannon's information-theoretic terms and how to optimize decisions under uncertainty using an expected utility function with Bayesian Experimental Design. We then describe the BIG framework – Bayesian Information Gain – where the computer "runs experiments" on the user by sending feedback that maximizes the expected gain of information by the computer, and uses the users' subsequent input to update its knowledge as interaction progresses. We demonstrate a BIG application to multiscale navigation, discuss some limitations of the BIG framework and conclude with future possibilities.

1 Introduction

Imagine Alice and Bob are playing the 20 questions game[1]. Alice has a number between 1 and 100 in mind and Bob can ask up to 20 questions to which Alice can reply only 'yes' or 'no' to guess that number. To maximize his chances of winning, Bob asks questions that give him maximum information at each step. He starts with "Is it between 1 and 50"? Alice replies "No". Bob continues with "Is it between 51 and 75"? Alice says "Yes". And the game continues until Bob guesses the correct number. Rather than asking less informative questions, such as "Is it between 1 and 10?", which would leave Bob with a range between 11 and 100, he optimizes the questions to reduce his uncertainty about the number in Alice's head.

[1] `https://en.wikipedia.org/wiki/Twenty_Questions`

Twenty questions is a common spoken parlor game, but how is it related to our interaction with computers? We are familiar with the notion that we give inputs (or commands) to the computer which in return executes these commands in a predetermined way. For example, when looking for a particular item on the web, we click on links to navigate the pages and the computer simply displays the pages. What if it could be more active by asking more informative questions to find out which item we are looking for?

In this chapter, we discuss an information-driven approach to design interaction. Information is defined in terms of the computer's knowledge about what the user wants. At the beginning of the interaction, the user (the role of Alice) has certain goals in mind, e.g. looking for a particular item on a website or typing a particular word on the keyboard. The computer (the role of Bob) has some uncertainty about the user's goal. This uncertainty is represented by the computer's prior knowledge, expressed in a Bayesian probabilistic model. When receiving input from the user, the computer updates its knowledge about what the user is looking for. Therefore, the information carried by the user input is the knowledge gained by the computer to discover the user's goal. We call this framework the Bayesian Information Gain (BIG) framework; it is based on Bayesian Experimental Design [5] using the criterion of information gain, also known as mutual information in information theory [2].

One can simply use BIG to measure the information sent by the user to the computer. However, by manipulating the feedback to maximize or leverage the expected information gain from the user's subsequent input, the computer can increase the information gain from the user, improving interaction efficiency.

2 Bayesian Information Gain Framework

The key concepts of the BIG framework are *uncertainty* and *information* on one hand, and *experimental design* on the other. We first go through these two concepts, described in information-theoretic terms (§ 2.1) and in Bayesian probability-theoretic terms (§ 2.2). This will help clarify the notion of "making optimal decisions under uncertainty". Finally we put it all together in the BIG framework (§ 2.3).

2.1 Uncertainty and Information

2.1.1 Entropy as a Measure of Uncertainty

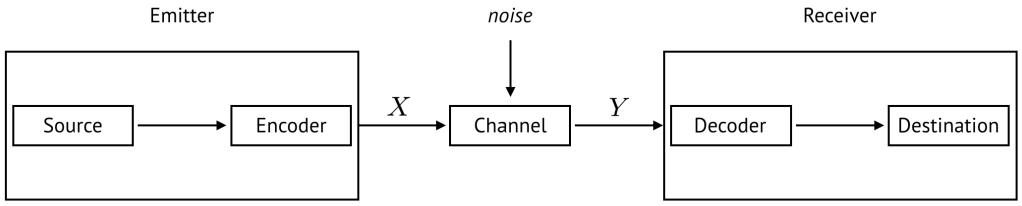

Figure 1: Shannon's communication scheme.

Information theory was originally proposed by Claude Shannon [9] using a communication paradigm (Fig. 1): A *source* produces messages, which are adapted by a *encoder* before being sent over a *channel*, and then decoded by a *decoder* to the final *destination*. The pair of source and encoder is called the *emitter* and the pair of decoder and destination is the *receiver*.

The emitter inputs X to the channel and the channel outputs Y to the receiver. Since there might be *noise* in the channel, output Y does not always equal input X. The semantic aspect of communication is not relevant to the engineering process of transmitting a source message through a channel [9]. Here the significant aspect of communication is only related to the probability of each possible outcome. Therefore, X and Y refer to random variables [2] with respective probability distributions $p(x)$ and $p(y)$, and the channel is completely described by the probability distribution of Y conditional on X, denoted by $p(y|x)$.

The receiver has a certain uncertainty on the encoded source message X, a random variable that can take several values. This uncertainty is captured by the *entropy* $H(X)$, a function of the distribution $p(x)$ defined as

$$H(X) = -\sum_x p(x) \log_2 p(x). \tag{1}$$

[2] To simplify, we only consider discrete random variables taking a finite number of possible values.

Because the logarithm is taken to base 2, the entropy is measured in bits (binary units).

The higher the entropy, the more uncertain the outcome, the harder the prediction. If N denotes the number of possible values of X, the entropy is bounded by $0 \leq H(X) \leq \log_2 N$:

- Minimum entropy is zero when X is deterministic:

$$H(X) = 0 \qquad \text{if} \qquad p(x) = 0 \text{ or } 1.$$

- Entropy is maximal when X is uniformly distributed, with N equiprobable values:

$$H(X) = \log_2 N \qquad \text{if} \qquad p(x) = \tfrac{1}{N}.$$

Taking the Alice and Bob example, if Bob (the receiver) has no idea which number Alice (the emitter) has in mind, he assumes a uniform distribution over the numbers from 1 to 100 and uncertainty is highest: $H(X) = \log_2 100 = 6.6$ bits. On the other extreme, if Bob somehow knows that Alice has the number 56 in mind, uncertainty is 0. All other cases will fall between 0 and 6.6.

2.1.2 Information as a Measure of Uncertainty Reduction

After asking the first question $Y =$"Is it between 1 and 50?", Bob reduces his uncertainty by receiving Alice's answer "No". Now the answer is reduced to 51 to 100, which is half of the original set. In other words, knowing the specific answer $Y = y$ ('No') has reduced the uncertainty about X: The remaining uncertainty is naturally captured by the entropy of X conditional on Alice's answer $Y = y$:

$$H(X|Y = y) = -\sum_x p(x|y) \log_2 p(x|y). \tag{2}$$

Knowing Alice's answer, the distribution over the remaining numbers is still uniform; the new uncertainty can be calculated as $H(X|Y=y) = \log_2 50 = 5.6$ bits. So in this round, he gained exactly $H(X) - H(X|Y=y) = 6.6 - 5.6 = 1$ bit of information. Had Bob asked the question $Y' =$"Is it between 1 and 10?" and had Alice reply "No", he would have gained only $H(X) -$

$H(X|Y'=y) = 6.6 - \log_2 90 = 0.1$ bits of information. That is why we consider this question less informative. Note that 1 bit of information gain is here the maximum possible since Bob's question can only be answered by the binary alternative 'yes' or 'no'.

Knowing the answer $Y = y$ from Alice decreased Bob's uncertainty (increased Bob's knowledge) about X. This specific case provides a certain amount of information, given by $H(X) - H(X|Y = y)$. The expected remaining uncertainty (averaged over all possible values y) is known as the *conditional entropy*:

$$H(X|Y) = \sum_y p(y) H(X|Y = y) = -\sum_y \sum_x p(x, y) \log_2 p(x|y) \qquad (3)$$

where $p(x, y)$ denotes the joint probability distribution of X and Y. The expected average information gain is then given by $H(X) - H(X|Y)$, which is the *mutual information* between the two random variables X and Y:

$$I(X;Y) = H(X) - H(X|Y) = \sum_y \sum_x p(x, y) \log \frac{p(x, y)}{p(x)p(y)}. \qquad (4)$$

In the latter expression we have used the formula $p(x, y) = p(x|y)p(y)$ relating joint, conditional, and marginal probability distributions. It can be shown that $I(X;Y)$ is always nonnegative [9], hence *on average*, knowledge of Y always reduces uncertainty about X.

It is also commonly considered that information is transmitted over a noisy channel, therefore some information might get lost: Alice may not hear the question correctly and thus gives the wrong answer, or Bob might not hear the answer correctly. Mutual information $I(X;Y)$ captures the *actually* transmitted information over the channel (Fig. 1). Mutual information is also bounded by two quantities: $0 \leq I(X;Y) \leq H(X)$:

- If no message gets transmitted correctly from the source to the receiver, (because the channel is too noisy) mutual information drops to its minimum 0;

- If all messages get transmitted perfectly from the source to the receiver, then the remaining uncertainty is $H(X|Y) = 0$ and mutual information is maximized, equal to the source entropy $H(X)$.

2.2 Bayesian Experimental Design

So how does Bob choose which question to ask? In other words, if he wants to guess the right number as efficiently as possible, how does he optimize the questions? This can be a rather complex problem depending on how Bob perceives Alice's behavior and on his previous knowledge of how she is inclined to choose a specific number. Perhaps Bob knows that Alice's lucky number is 13, and she would rather pick her lucky number than any other number like 56. Or perhaps Alice is mathematically inclined and is known to prefer prime numbers to composite ones, and so on.

This can be given a probability-theoretic framework as an instance of a *parameter estimation problem*: Bob wants to estimate parameter θ, an unknown value that Alice is thinking about. To estimate θ, Bob has at his disposal some set of observations on Alice, or some measured data $Y = y$. Bob assumes some statistical model of Alice's behavior as a probability distribution $p_\theta(y)$.

In a *Bayesian* setting, Bob assumes a prior distribution $p(\theta)$ of the unknown value that may help his estimation. Without any prior knowledge, he can simply assume that θ is uniformly distributed as in the 20 questions example described above. But if Bob has some prior knowledge about Alice, $p(\theta)$ may be more informative. Since θ is now the outcome of a random variable Θ, the statistical model can then be seen as a conditional distribution $p_\theta(y) = p(y|\theta)$ of Y given Θ. Using this, Bob can then update (hopefully improve) his knowledge by applying Bayes' rule to compute the *posterior* distribution of θ given his observations:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}.$$

The theory of Bayesian experimental design [1] was originally proposed by Lindley [5], inspired by Shannon's work [9]. In this framework, Bob designs an experiment $X = x$ to challenge Alice and receives an observation Y from Alice that depends on X and, of course, also on the parameter θ that she is thinking about. Alice's behavior model is now given by the conditional distribution $p(y|x, \theta)$ and Bob tries to optimize the experiment outcome $X = x$ using some utility function $U(x)$ before updating the posterior distribution using Bayes' rule:

$$p(\theta|x, y) = \frac{p(y|x, \theta)p(\theta)}{p(y|x)} \tag{5}$$

where

$$p(y|x) = \sum_{\theta} p(y|x,\theta)p(\theta)$$

is an average conditional distribution which can be seen as a "communication channel" between Bob and Alice. Here we have used that $p(\theta|x) = p(\theta)$ because the parameter Θ unknown to Bob is *a priori* independent from the experimental design X.

The utility $U(x)$ is computed from the prior and posterior distributions and is averaged over all possible Alice's outcomes y (*expected* utility). It can be defined as the information gained about the random variable [5], or the financial or other cost of performing the experiment. Maximizing $U(x)$ provides the *optimal decision under uncertainty* (with respect to the given utility function). In other words, when designing an experiment, the goal is to maximize the expected utility of the experiment's outcome.

Interaction between Alice and Bob can further be modelled as a *sequence of experiments*: Each question Bob asks is an "experiment" on Alice and the criterion for choice of the experiment then becomes to maximize the expected utility between the current prior and posterior distributions. At every step, the old posterior becomes the new prior and is further updated from the new experiment. Ideally the process continues until no uncertainty remains, which means that θ is perfectly known. This corresponds to a deterministic $\Theta = \theta$ given all previous experiments.

2.3 Putting it Together – Bayesian Information Gain

BIG is a general framework to design interaction (Fig. 2). We consider the user Alice and the computer Bob. Here the goal of the computer is to ask "clever" questions guided by a utility function to find out what the user wants. The computer does so by "running experiments" on the user through the feedback $X = x$ that it provides, and using the user's subsequent input $Y = y$ as the outcome of the experiment to update its knowledge about the user's goal $\Theta = \theta$.

BIG uses the following notations that are common for Bayesian Experimental Design [5]:

1. The random variable Θ represents the possible intended targets in the user's mind.

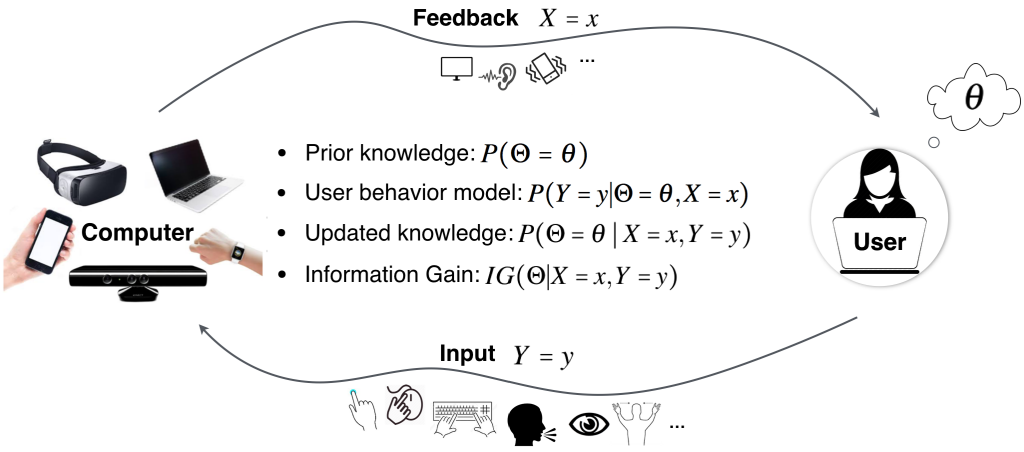

Figure 2: The BIG framework. There are three key random variables: The potential targets Θ, system feedback X and user input Y. The computer also has some prior knowledge about the user's intended target $p(\theta) = P(\Theta = \theta)$ and a user behavior function expressing what the user would do $p(y|x, \theta) = P(Y = y|\Theta = \theta, X = x)$. After sending the feedback $X = x$ and receiving the user input $Y = y$, the computer updates its knowledge about the user's goal and calculates the information gain from the user input. In order to play a more active role, the computer can try to maximize the expected information gain or leverage it for better interaction by manipulating the feedback.

2. Its probability distribution $p(\theta)$ (given for all values of θ) is the prior distribution of targets, which expresses the computer's prior knowledge about Θ. This distribution can be uniform if no data about the user's interests is available, or can be based on external data sources or interaction history.

3. X represents any possible feedback provided by the computer and $X = x$ is a particular feedback sent to the user.

4. Y represents any particular command y issued by the user.

5. $p(y|x, \theta)$ is the probability of the user giving an input command $Y = y$ when she wants $\Theta = \theta$ and sees $X = x$. This can be modeled from the interaction history, or by user calibration, and can be user-independent or user-dependent.

6. $p(\theta|x, y)$ is the computer's updated knowledge about the user's goal after showing the user $X = x$ and receiving the input $Y = y$ from the user. This is the posterior distribution calculated through Bayes' theorem as in Equation (5).

7. $I(\Theta; Y|X = x)$ is the mutual information between what the user wants and what she provides as input when seeing $X = x$. As explained above, it is the difference between the entropy and the conditional entropy:

$$I(\Theta; Y|X = x) = H(\Theta) - H(\Theta|X = x, Y). \tag{6}$$

   Here for a given $X = x$, knowing Y decreases uncertainty about Θ, by a quantity which is precisely the mutual information $I(\Theta; Y|X = x)$.

   We use $U(x) = I(\Theta; Y|X = x)$ as the expected utility function in the Bayesian experimental design: It can be interpreted as the *expected* information gain, and as such is always positive. To calculate this, we use Bayes' theorem for entropy [2] to convert Equation (6) to:

$$I(\Theta; Y|X = x) = H(Y|X = x) - H(Y|\Theta, X = x). \tag{7}$$

   where the first term is given by $-\sum_y p(y|x) \log_2 p(y|x)$ as in Equation (2), and the second term is $-\sum_{y,\theta} p(\theta)p(y|x, \theta) \log_2 p(y|x, \theta)$ as in Equation (3).

8. $IG(\Theta|X = x, Y = y)$ is the difference between the computer's previous knowledge $H(\Theta)$ and current knowledge $H(\Theta|X = x, Y = y)$ about the user's goal, representing the *actual information gain* carried by the user input:

$$IG(\Theta|X = x, Y = y) = H(\Theta) - H(\Theta|X = x, Y = y). \tag{8}$$

   Information gain might be negative if the user, e.g. makes an error, but is positive on average since

$$\sum_y p(y) IG(\Theta|X = x, Y = y) = I(\Theta; Y|X = x) \geq 0.$$

Table 1 summarizes the notations in Bayesian Experimental Design and Bayesian Information Gain respectively.

	BED	*BIG*
θ	parameter to be determined	intended target in the user's mind
y	observation	user input
x	experimental design	system feedback
$p(y\|\theta, x)$	model for making observation y, given θ and x	model for user providing input y, given θ and x
$p(\theta)$	prior	system's prior knowledge about the user's goal
$p(\theta\|y, x)$	posterior	updated knowledge
$I(\Theta; Y\|X = x)$	utility of the design x	utility of the feedback x
$IG(\Theta\|X = x, Y = y)$	utility of the experiment outcome after observation y with design x	utility of the outcome after user input y with system feedback x

Table 1: Notations in Bayesian Experimental Design (BED) and in Bayesian Information Gain (BIG) respectively.

One can always calculate the actual information gain, or the information carried by the user input informing the computer what she wants with Equation (8) – "Running a normal experiment". By manipulating the feedback with Equation (6), e.g. finding the $X = x$ that maximizes or leverages the expected information gain, the system "redesigns the experiment", or "runs a better experiment" on the user in order to gain more information about the user's goal, i.e. the intended target. The computer then plays a more active role and therefore increases interaction efficiency.

3 Application

BIG is a general approach that can be applied to a wide range of interaction tasks. We describe BIGnav, an application of BIG to multiscale navigation [6].

Multiscale interfaces are a powerful way to represent large datasets such as maps, documents and high-resolution images. The canonical navigation commands in this type of interfaces are pan and zoom (as seen in many applications such as Google Maps [3]): Panning lets users change the position of the view while zooming lets them modify the magnification of the viewport [3,4].

3.1 BIGnav Implementation

First we define the three key random variables in the BIG framework Θ, X and Y for the multiscale navigation scenario:

- Θ represents any point of interest in the multiscale space. For each target θ, the probability that it is the actual intended target is $p(\theta)$. These probabilities constitute the a priori knowledge that the system has about the user's interest, and is updated as the user navigates.
- X represents any possible view provided by the system. $X = x$ is a particular view shown to the user. Note that the number of possible views is potentially very large.
- Y represents any particular command y issued by the user. The possible input commands are: move in one of the 8 cardinal directions, zoom in or click on the target when it is big enough to be clickable. Note that zooming out is not implemented in BIGnav: if the target is out of view, the user should indicate in which direction it is rather than zooming out.

(1) Interpreting user input: Given the view x shown to the user and the user's intended target θ, $p(y|x,\theta)$ is the probability that the user provides an input command $Y = y$ given θ and $X = x$. This probability distribution is the system's interpretation of the user's intention when giving this command. For example, if city A is to the left of the user, what is the probability of the user giving the left command when knowing that city A is located to her left, provided she can only go left or right? $p($*go left* $|$ *city A is located to the left of the current view, city A is the intended target*$) = 1$ if the user is completely confident about what she is doing. But maybe the user is not accurate all the time. Say she is correct only 95% of time, then we need to

[3] `https://www.google.com/maps`

consider that she makes errors: $p(\textit{go left} \mid \textit{city A is located to the left of the current view, city A is the intended target}) = 0.95$ and $p(\textit{go right} \mid \textit{city A is located to the left of the current view, city A is the intended target}) = 0.05$. Probability $p(y|x, \theta)$ is a priori knowledge that must be given to the system.

(2) Updating system's knowledge: Given the view x shown to the user and the user reaction y to that view, the system can update its estimate $p(\theta|x, y)$ of the user's interest with Equation (5). If the system has no prior knowledge about the user's intended target, e.g. at the beginning, each θ has the same probability of being the target and $p(\theta)$ is uniform. As the user issues commands, the system gains knowledge about the likelihood that each point of interest be the target, reflected by the changes to the probability distribution. This is done, for each point of interest, by taking its previous probability, multiplying by the above user input function $p(y|x, \theta)$, and normalizing it so that the sum of the new probabilities over all the points of interest equals one.

(3) Navigating to a new view: With the new probability distribution after receiving user input, BIGnav then goes over each view $X = x$, calculates its expected information gain with Equation (7) and picks the view for which it is maximal. To maximize Equation (7), BIGnav looks for a trade-off between two entropies. To maximize the first term, the view should be such that all user commands given that view are equally probable (for the system). To minimize the second term, the view should provide the user with meaningful information about the points of interest. Maximizing a difference does not necessarily mean to maximize the first term and minimize the second, so the maximum information gain is a trade-off between these two goals. For example, showing only ocean will increase the first term but will also increase the second term. After locating the view with maximal information gain, BIGnav navigates there and waits for the user's next input.

An interactive 1D version of this implementation can be found in a Jupyter notebook [4] and a video is available on YouTube [5].

3.2 Comparison with Standard Navigation

In order to compare BIGnav with standard pan and zoom navigation (STDnav), we implemented a 2D version [6]. In this more realistic setting, we face

[4] `https://github.com/wanyuliu/5thComputationalInteraction/blob/master/BIGMap.ipynb`

[5] `https://www.youtube.com/watch?v=N2P-LFh1oLk`

a computational challenge because the system feedback X can have a huge number of possible views. With BIGnav, we need to calculate the information gain corresponding to every single view $X = x$, which would incur an enormous computational cost if views could be centered at any pixel and have any size. We therefore discretize the set of views by using tiles and discrete zoom factors. This is similar to some pan-and-zoom applications where users can pan in four directions by fixed amounts, and zoom in and out by fixed amounts. We therefore reduce the set of commands to make computation tractable in our prototype. We slice the view into nine regions representing eight panning directions and a central zooming region (Fig. 3). The eight panning regions have a 45° angle, and the zooming region is half the size of the view. Furthermore, we model user behavior with a calibration session. The results (Table 2) show that 90% of panning commands are correct and 4% are in one of the adjacent directions (Fig. 3). For zooming commands, 95% of the commands are correct while for clicking on the target, 100% of the commands are correct.

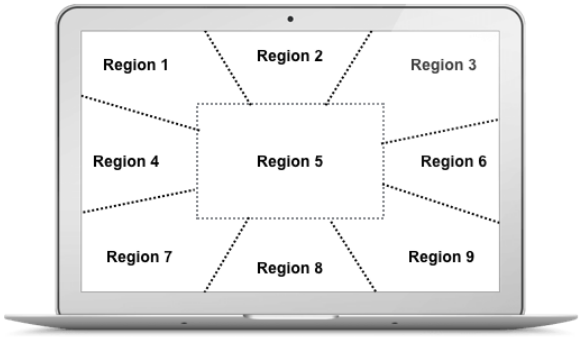

Figure 3: Nine regions representing user input, delimited by dotted lines. Panning regions also include the space outside the current view.

Based on this setup, we ran a controlled experiment and found that BIGnav was up to 40% faster than STDnav. Fig. 4 shows the number of navigation steps required (x-axis) to reach the target in STDnav and BIGnav, as well as how uncertainty and information gain evolve over time. We can see that with STDnav, sometimes a command does not make a difference in uncertainty, i.e. the information gain is null. This is typically the case when the system is certain of what the user is going to do. For example,

Command	*Main Region*	*Adjacent Regions*	*Other Regions*
Pan	0.90	0.04	0.0033
Zoom	0.95	0.00625	0.00625
Click	1	0	0

Table 2: Calibration results used as prior knowledge about the user behavior $p(y|x, \theta)$.

when completely zoomed out, users must zoom in. Similarly, if a view contains 99% of the probability distribution, users will almost certainly zoom in. Therefore such feedback is not an "intelligent" question. On the contrary, BIGnav optimizes the feedback at each step to gain a maximum amount of information and reduce the computer's uncertainty.

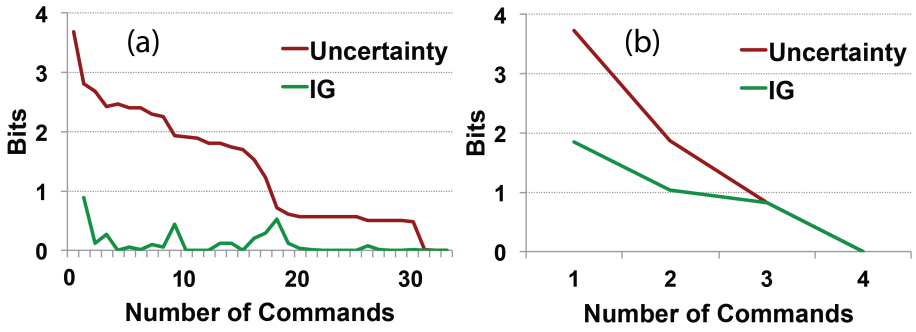

Figure 4: Uncertainty and information gain (IG) for each successive command in (a) STDnav and (b) BIGnav.

However, despite being more efficient, BIGnav incurs a higher cognitive load (Fig. 5). Instead of executing users' panning and zooming commands, BIGnav returns the feedback that maximizes the expected information gain from the user's subsequent input. This new feedback might be far away from the current view, therefore the user has to interpret what the system has just done and reorient themselves before inputting the next command, whereas with STDnav the user can anticipate the system response. A subsequent version of BIGnav uses animation during the transitions between views to help users orient themselves and anticipate their next action.

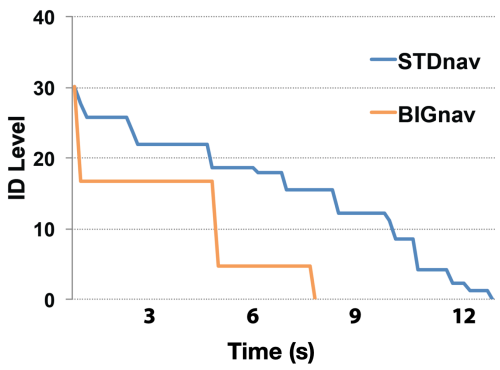

Figure 5: Time plot of the decrease in index of difficulty (ID) in the STDnav and BIGnav conditions, for two trials with the same other conditions.

3.3 Other Applications

BIG is a general framework that can be applied to a wide range of interaction tasks. Once the potential targets Θ and their probability distribution $p(\theta)$, system feedback X, user input Y and user behavior $p(y|x,\theta)$ are modeled, one can compute the actual information gain, or the information carried by the user input informing the computer of what she wants. We have applied this approach to two other areas: file retrieval [7] and collective music making [8].

BIGFile [7] is an interaction technique for fast hierarchical file retrieval that leverages the expected information gain. Unlike BIGnav, BIGFile features a split adaptive interface that combines a traditional file navigation interface with an adaptive part that displays the shortcuts selected by BIG. Users can use any shortcut in the adaptive area or simply navigate the hierarchy as usual. BIGFile uses an approximate but efficient algorithm to select shortcuts, and was shown to be up to 40% faster than standard file navigation.

Entrain [8] is an intelligent agent for encouraging social interaction in collective music making. Here BIG is used to adapt a probabilistic model of individual user behavior, which is calculated based on user activity and rhythmic periodicity, and modulates music creation at both individual and collective levels via visual and auditory feedback.

We encourage readers to think about their own interaction scenario in a "BIG" way: How much uncertainty and information are there in the interaction loop? Is the current interaction information efficient? How to have the

computer play a more active role by demanding more information?

4 Discussion

In this section, we discuss some limitations of the BIG approach and propose potential solutions as well as future possibilities.

4.1 Approximation of Prior Distribution

When no prior information is available, a common practice is to assume a uniform distribution. Then the computer continuously updates its knowledge when receiving user input via Bayes' rule. The approximation of prior distribution can be derived from external data. For example, for a map navigation, it could be based on a large dataset representing the general population, such as the popularity of tourist destinations. It could also be based the user's own interaction history, such as "my favorite top destinations". In BIGnav [6], we showed that even with a uniform prior distribution, BIGnav is more efficient than the standard pan and zoom navigation. The more precise the approximation of the prior distribution, the better BIG works.

4.2 Dynamic User Behavior Model

The user behavior model is the computer's belief about which command the user will issue given what she wants and what she sees. In Bayesian Experimental Design's terms, it is the probability of observing an outcome given the parameter of nature and the experiment. Similar to the approximation of the prior distribution, the better we model it, the better BIG works. We collected user behavior from a calibration session in BIGnav [6] and from the literature in BIGFile [7]. In both cases this model stays constant for all participants throughout the session. However, we know that users' behavior changes over time. Novices' behavior is different from experts'; people have individual differences; one's behavior might change given a particular environment. Therefore, a better solution is to have a dynamic user behavior model, tailored to a specific user whose behavior changes due to time or her interaction with the environment. This can be done by logging user's interaction over time and constructing the user behavior model from real data.

4.3 Computational Cost

A crucial aspect of interaction is to provide immediate system feedback to users' actions. The BIG method is computationally costly since the calculation of mutual information requires the sum of all possibilities: The overall computational cost is $N_\Theta \times N_X \times N_Y$ (N is the number of possible values). In BIGnav, we discretized system feedback and user input to achieve real-time feedback by the system. In BIGFile we introduced an efficient but approximate algorithm to reduce the computing cost [7].

Other approaches are worth exploring. For example, instead of searching the entire information space (exhaustive search), we could regularize the search to compute the locally maximal expected information gain. We could also use a timer to provide a first, coarse response in, e.g. 0.1s, and refine the search if the user does not provide input. The actual optimization depends on the exact context of use.

4.4 Other Utility Functions

While information alone can be used as the expected utility function for an "informative experiment" to reduce uncertainty as efficiently as possible, other measures can be considered. For example, we can add a cost to the utility function for the estimated user's cognitive load, so that an optimal feedback needs to be both informative and comfortable. In BIGnav, this cost could be a function of the estimated time for the user to get reoriented after the new feedback.

Other utility functions might include, for example, successful completion of a task, such as success rate, or the time to correct errors if some errors are more severe than others, or dwell time over areas of interest in an eye-tracking experiment. The optimal system feedback, i.e. the experiments the computer runs on the users, depends on the chosen utility criterion.

4.5 Future Challenges

We started this chapter with an example of Alice having a number between 1 and 100 in her head and Bob trying to guess that number by asking informative questions. Similarly, we assumed that there is a finite set of potential targets, that the user has a single intended target in her head, and that this target does not change until it is found. BIG is a goal-oriented framework:

It does not support pure exploration, when the user has no intended target. How to incorporate these situations, i.e. a changing target, no target or more than one targets is an interesting challenge for future work.

5 Conclusion

In this chapter we introduced Bayesian Information Gain, a general framework that can be applied to many interaction tasks. In order to compute *information*, one needs to model Θ, the set of potential targets; $p(\theta)$, the computer's prior knowledge about the user's goal; X, the possible system feedback; Y, the possible user input, and $p(y|x, \theta)$, which models the user behavior. BIG can then calculate how much information there is in the user input to reduce the computer's uncertainty, and optimize interaction efficiency by having the computer extract more information at each step by manipulating the system feedback.

BIG is an example of a probabilistic interface. Similar to other probabilistic interface architectures that treat user input as an uncertain process, BIG uses a user behavior model to represent the ambiguity of user input for the computer. In conventional settings, there is no information-theoretic uncertainty for the user regarding the computer's behavior. However, when presenting feedback to the user, the computer is uncertain about what the user will do. From the user's perspective, BIG presents non-deterministic system feedback that challenges the user by leveraging a Bayesian model in order to maximize information gain. It opens the door to a wide range of "BIG" applications and a new era of probabilistic interaction.

References

[1] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[2] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[3] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of the SIGCHI Confer-*

ence on Human Factors in Computing Systems, CHI '95, pages 234–241, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[4] Yves Guiard and Michel Beaudouin-Lafon. Target acquisition in multiscale electronic worlds. *International Journal of Human-Computer Studies*, 61(6):875–905, 2004.

[5] Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

[6] Wanyu Liu, Rafael Lucas D'Oliveira, Michel Beaudouin-Lafon, and Olivier Rioul. Bignav: Bayesian information gain for guiding multiscale navigation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 5869–5880, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Wanyu Liu, Olivier Rioul, Joanna McGrenere, Wendy E. Mackay, and Michel Beaudouin-Lafon. Bigfile: Bayesian information gain for fast file retrieval. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA, 2018. Association for Computing Machinery.

[8] Hugo Scurto, Wanyu Liu, Benjamin Matuszewski, Frédéric Bevilacqua, Jean-Louis Frechin, Uros Petrevski, and Norbert Schnell. Entrain: Encouraging social interaction in collective music making. In *ACM SIGGRAPH 2019 Studio*, SIGGRAPH '19, New York, NY, USA, 2019. Association for Computing Machinery.

[9] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.