The Clutterpalette: An Interactive Tool for Detailing Indoor Scenes

Lap-Fai Yu, *Member, IEEE*, Sai-Kit Yeung, *Member, IEEE*, and Demetri Terzopoulos, *Fellow, IEEE*

Abstract—We introduce the Clutterpalette, an interactive tool for detailing indoor scenes with small-scale items. When the user points to a location in the scene, the Clutterpalette suggests detail items for that location. In order to present appropriate suggestions, the Clutterpalette is trained on a dataset of images of real-world scenes, annotated with support relations. Our experiments demonstrate that the adaptive suggestions presented by the Clutterpalette increase modeling speed and enhance the realism of indoor scenes.

Index Terms—Interactive 3D modeling, scene modeling, scene understanding, indoor scenes, modeling tools, suggestive user interfaces

1 Introduction

Visual realism is one of the defining goals of computer graphics. While realism is often approached in terms of rendering fidelity, it is also a modeling problem [1]. Realism calls for creating synthetic environments that display a convincing level of detail, on par with typical real-world scenes.

Consider the kitchens in Fig. 2. The contrast between the real-world scenes and the synthetic ones is stark. Without the odds and ends that populate real-world scenes, the synthetic environments appear eerily barren, devoid of traces of life. The lack of small-scale objects is characteristic of synthetic environments and commonly undermines their realism. As observed by Xu et al. [2], "computer graphics scenes are often unrealistically simple or overly tidy."

The difficulty of populating a scene with detail items is due in part to the large number and variety of such objects that can appear in a typical scene. A realistic indoor environment can easily contain over a hundred detail items—books, stationery, and computing equipment in an office; dinnerware, cookware, and food items in a kitchen; clothes, bedding, and decor in a bedroom. Searching for each individual item becomes tedious at this scale. The mere identification of items that could fit well at a particular place in the scene becomes a chore when it must be repeated hundreds of times.

To facilitate the enhancement of synthetic indoor scenes with detail items, we propose the *Clutterpalette*, an interactive tool that helps modelers enrich their scenes. When the modeler points to a location in the scene, the Clutterpalette suggests appropriate items to detail that location. The user thus retains control over the content of the scene, but the laborious search for appropriate clutter objects is automated. The Clutterpalette identifies appropriate clutter items, which are presented to the user. A scene can thus be rapidly populated by repeatedly picking a suggested object from the Clutterpalette until a sufficient level of detail is reached. Fig. 1 shows a living-room scene before and after detailing using the Clutterpalette.

To suggest appropriate objects, the Clutterpalette must be trained using data, since manually codifying the dependencies between hundreds of types of clutter objects, furniture objects, and scenes would be impractical. The need for training data presents us with a circular dependency: Since creating realistically detailed scenes using current modeling tools is tedious and time-consuming, there are few such scenes in the public domain. We overcome this difficulty by training the Clutterpalette on real-world imagery. Specifically, the Clutterpalette is trained on a dataset of images of real-world indoor scenes, annotated with object types and support relations. This yields a scalable training pipeline that transfers the semantics of real-world scenes to 3D modeling.

We evaluate the utility of the Clutterpalette in a set of experiments in which participants model different types of indoor scenes and independent evaluators assess relative scene realism. The experimental results demonstrate that the suggestions presented by the Clutterpalette speed up modeling time and enhance the realism of indoor scenes.

2 Background

2.1 Set Dressing

The task of populating a scene with clutter objects is referred to as *set dressing* in film making [3], which is carried out by set dressers under the direction of a leadman, a set decorator, a property master, and a production designer. Set dressing is an important step that sets the tone and ensures the authenticity of a film. Similarly, in the production of computer-animated films and video games, professional set dressers are hired to model and dress sets and props for the virtual worlds in which filming or gameplay takes place. The Clutterpalette facilitates and simplifies the digital production process, by providing convenient and realistic suggestions of clutter objects to "dress" the virtual scene. Fig. 3 illustrates the task of set dressing a virtual scene.

- *L.-F. Yu is with the University of Massachusetts Boston.*
- *S.-K. Yeung is with the Singapore University of Technology and Design.*
- *D. Terzopoulos is with the University of California, Los Angeles.*

Manuscript received 12 July 2014; revised 3 Nov. 2014; accepted 14 Nov. 2014. Date of publication 29 Mar. 2015; date of current version 2 Jan. 2016. Recommended for acceptance by M. Agrawala.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2015.2417575

Fig. 1. A living-room before (top) and after (bottom) detailing using the Clutterpalette.

Fig. 3. Left: An input scene. Right: Snapshot of a portion of our clutter object database. Set dressing involves the non-trivial task of selecting proper clutter objects to detail a scene.

2.2 Related Work

The difficulty of modeling realistic indoor scenes has been recognized in computer graphics for a long time. Early work focused on assisted placement and arrangement of objects. Bukowski and Séquin [4] developed a system for assisted placement of furniture and other objects in indoor scenes. The system helped align objects by associating them to each other—keeping bookshelves against walls, for example, or tables on floors. The associations between different types of objects were specified manually. Xu et al. [2] introduced a system that automatically placed a large number of objects in an indoor scene. The system is guided by a set of manually-specified constraints that describe relationships between different types of objects. Following this line of work, Merrell et al. [5] presented a system that assists furniture arrangement in indoor scenes. The system relies on manually-encoded interior design guidelines. Relationships between furniture types are again specified manually. The reliance on manual constraint specification limits the scalability of these approaches. For example, in order to handle a new scene type, such as a classroom or a hotel lobby, all relevant relationships between all types of objects would need to be specified in detail. In contrast, our Clutterpalette pipeline is designed to deal with the numerous small and diverse detail items that populate indoor scenes. As we demonstrate, it is highly scalable.

Yu et al. [6] introduced an optimization-based approach for furniture arrangement that is trained on data. For each type of scene, five example scenes were manually constructed to illustrate relationships between furniture types. This limits the scalability of the approach in dealing with new scene types and with a large number of clutter objects. Fisher and Hanrahan [7] described a context-aware search engine for 3D models that is trained on data. Given an incomplete indoor scene and a bounding box specified by the user, the system retrieves objects that fit the scene and the bounding box, based on pairwise relationships and individual object descriptors extracted from the data. The training data is assumed to consist of complete 3D scenes and the system relies on fine-grained geometric properties of and spatial relationships between objects in the dataset. In our setting, this assumption is untenable due to the dearth of realistically cluttered scenes in publicly available 3D

Fig. 2. Real-world (left) and synthetic (right) kitchens. The images of real-world scenes were obtained by searching Flickr Creative Commons with the keyword “kitchen”; the synthetic scenes were obtained by searching the Trimble 3D Warehouse with the same keyword. The synthetic scenes appear comparatively barren.

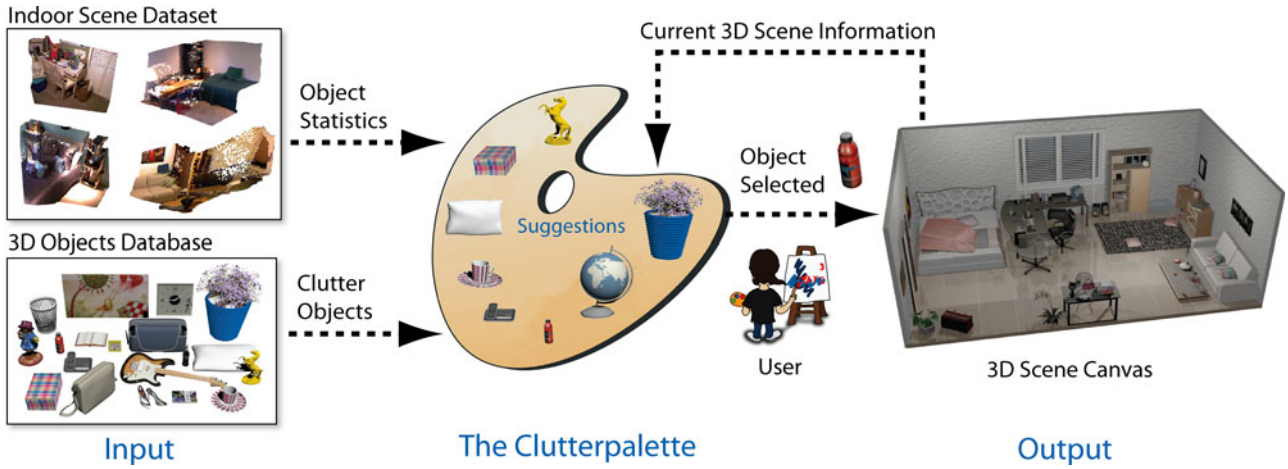


Fig. 4. The Clutterpalette framework.

virtual scene repositories. For this reason, our Clutterpalette is trained on annotated images of real-world scenes.

Fisher et al. [8] describe an approach to comparing synthetic 3D scenes based on geometric properties of and spatial relationships between objects in the scenes, which also relies on the availability of a dataset of synthetic scenes that demonstrate proper relationships between objects. By contrast, our approach decouples the dataset of clutter objects from the dataset of images that demonstrate relationships between objects. This allows our two datasets to scale independently—additional clutter instances broaden the range of objects available to the modeler and additional training images broaden the system's knowledge about object relationships. Clutter instances and real-world training images can be added independently.

Yeh et al. [9] describe an optimization approach that procedurally arranges a large number of objects in a scene, which relies on manually specified relationships between objects. Fisher et al. [10] present an approach that synthesizes object arrangements that are similar to given input arrangements. Xu et al. [11] present a sketch-based modeling system for indoor scenes. These approaches are not trained on real-world data. Alternatively, Majerowicz et al. [12] learn object arrangement from annotated images of real-world shelves to automatically generate style-preserving object arrangements for virtual shelves. Their focus is on synthesizing object arrangements on shelves while ours is on devising an interactive, suggestive modeling interface for detailing general indoor scenes.

Unlike existing approaches, we train a general-purpose model using an indoor scene dataset to learn clutter object distributions as observed in real-world settings. The indoor scene dataset contains real-world scene images precategorized into different scene types. Each image is also annotated with the labels of the supporter and clutter objects as well as the support relations. This allows our model to learn and reason about object distributions both in the global (scene) and local (supporter) contexts, for use in detailing complete virtual scenes. Our Clutterpalette tool learns to suggest different objects for the same supporter in different types of scenes. For example, while detailing the same shelf, a pair of shoes is suggested in the context of a bedroom, while a plate is suggested in the context of a kitchen. Our model also learns different co-occurrence probabilities according to the scene type. For example, while the model finds it likely to put multiple laptops in an office, it finds it much less likely to have more than one or two laptops in a bedroom.

Our general-purpose model is capable of detailing any supporter—e.g., floor, bed, table, shelves, wall, door—on which clutter objects lie or to which clutter objects are attached in the real world. We base our model on first principles and train it with real-world probabilities. The final formulas are simple and closed-form, yet they are well-grounded, making them practical for building a novel interactive modeling tool where real-time performance is necessary. Compared to automatic synthesis approaches, our interactive approach enables the artist to retain control over the modeling process while detailing a scene to any desired extent, greatly reducing the tedium of object selection from a large database of clutter objects.

Our interface builds on the idea of suggestive interfaces proposed by Igarashi and Hughes [13]. In such interfaces, the authoring tool reasons about operations that the user is likely to undertake and presents suggestions that anticipate the user's choices. The user can pick one of the suggestions or disregard them all. In this way, content creation is made faster and easier. Suggestive interfaces have been proposed for use in various applications, including modeling individual 3D objects [14], [15], assigning textures to surfaces in 3D scenes [16], assigning materials to parts of 3D objects [17], designing physically valid furniture [18], and designing building layouts [19]. In our earlier work [20], we developed a data-driven suggestive interface for dressing virtual characters. In the present paper, we develop a suggestive interface for enhancing the level of detail of indoor scenes.

3 Overview

Fig. 4 shows an overview of the Clutterpalette framework. The training input consists of an indoor scene dataset that has been manually annotated with object types and support relations. The Clutterpalette is trained in advance using the object distribution statistics learned from this dataset. Subsequently, during use, the Cluttterpalette is linked to a 3D object database, from which highly probable clutter objects

Fig. 5. Interaction with the Clutterpalette. Left: The user points to the counter to add a clutter object. This prompts the Clutterpalette to suggest objects appropriate for the location. Right: the scene after one of the suggested objects is selected by the user and added to the scene.

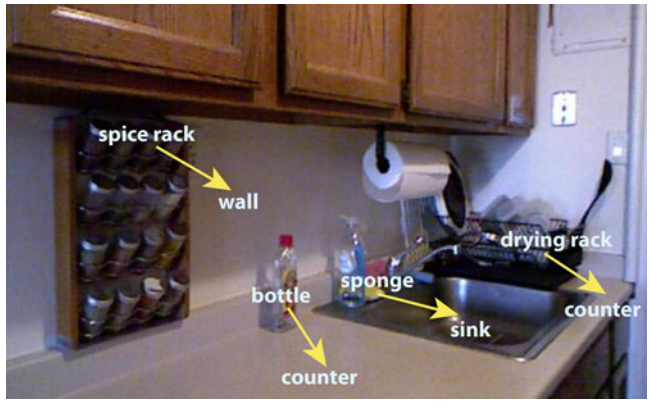


Fig. 6. An image from the NYU Depth Dataset [21] with some of the annotated support relations indicated.

are selected and suggested to the user. As the user interactively details different parts of the scene, the Clutterpalette adaptively provides suggestions, taking into account the objects that have already been added by the user. The interactive modeling process continues until the scene has been detailed to the user's satisfaction. The following sections explain the interaction and the training processes.

4 Interaction

We assume that the user starts with a room in which appropriate furniture has been arranged, say with any of the existing approaches to furniture layout [2], [4], [5], [6]. The user then enhances the scene using the Clutterpalette. When the user points to a location in the scene, the Clutterpalette suggests the types of objects that are most likely to appear there. The likelihood computation is based on the type of scene (kitchen, bedroom, office, etc.), the type of supporting object (floor, wall, desk, bed, etc.), and the clutter items that are already present on the supporter. This computation is based on statistics extracted from a dataset of real-world images of indoor scenes, which we will describe in the next section.

The most likely clutter types for a given location are presented in ranked order within a menu. For each clutter type, a randomly chosen object of this type is shown in situ, in the context of the scene, so that the user can conveniently preview how the scene would appear if the object were added. If the chosen object collides with existing clutter objects, another object of the same type is sampled. Fig. 5 depicts the interface. The user can quickly place the presented object in the scene by double-clicking on it. To see additional instances of the presented type (e.g., other bottles, other jars, other cups, etc.), the user can click on the corresponding object once, which brings up a secondary panel that shows additional instances of the clicked object type (in our experiments described in Section 7, the Clutterpalette presents about five object instances for each object type). Again, the user can immediately place one of the object instances in the scene by double-clicking on it. This hierarchical selection scheme affords greater flexibility in choosing a particular object instance of the suggested type; for example, out of personal preference for object color or style.

Throughout the modeling process, the user can also freely reposition and delete objects, after which the new scene setting will automatically be taken into account and appropriate new suggestions will be presented by the Clutterpalette. The Clutterpalette interface is further demonstrated in the accompanying video.

5 Training Data

The suggestive functionality of the Clutterpalette relies on computing the likelihood of a given clutter type appearing at the selected location. Section 6 describes the likelihood computation more fully. The computation relies on conditional probabilities that relate types and quantities of clutter objects to each other and to the scene type as well as to the type of supporting object on which the clutter resides. These probabilities are estimated by extracting empirical statistics from real-world observations of lived-in, cluttered, indoor scenes. Each observation needs to provide only the following information: The type of the scene (e.g., kitchen), a list of observed supporting objects (e.g., floor, wall, counter), and a list of clutter objects on each supporting object (e.g., bottle on counter, another bottle on counter, cup on counter, clock on wall, garbage bin on floor).

The NYU Depth Dataset produced by Silberman et al. [21] to provide data for training and evaluating scene understanding algorithms serves as our source of observations about real-world scenes. This dataset contains 1,449 RGBD images of 464 indoor scenes from three cities. The scenes appear in their natural, messy condition. They employed the Amazon Mechanical Turk crowdsourcing Internet marketplace to annotate individual objects in the images as well as the support relationships between objects. We use the annotated information to train our Clutterpalette.

Although the NYU Depth Dataset contains both color and range images, the range data was not used in obtaining the annotations—only the color images were provided to Amazon Mechanical Turk. A dataset of similarly annotated color images can thus be used to provide additional training data for the Clutterpalette.

A sample image from the NYU Depth Dataset is shown in Fig. 6. We trained the Clutterpalette on five scene types from the dataset—bedrooms, kitchens, living-rooms, offices, and classrooms. For each scene type, we identified the most common clutter types (e.g., book, bottle, box, etc.). As shown in Table 1, several dozen clutter types were identified for each scene type. Some of the object types annotated in the NYU Depth Dataset were not used due to a lack of

TABLE 1
Properties of the NYU Depth Dataset [21] Used to Train the Clutterpalette

	Number of training images	Number of clutter types	Number of support relations
Bedroom	383	85	5,843
Kitchen	225	67	4,067
Living-room	221	66	3,712
Office	78	53	1,245
Classroom	49	62	1,315

available 3D models. For each clutter type that was used, we collected four or five object instances from the Trimble 3D Warehouse and from other online sources of free 3D models. In total, our 3D object database contains 176 clutter object types and 786 object instances.

6 Suggestion Generation

When the user points to a location in the scene, the Clutterpalette must determine which clutter types to suggest and in what order. In essence, it must answer the following question: "If there was an additional clutter item at this location, what would that item most likely be?" This is done by evaluating the likelihood that a given clutter type would appear at the identified location. The types with the highest likelihood are presented to the user in rank order.

The Clutterpalette evaluates the likelihood of a given clutter type conditioned on the type of the scene, the type of the supporting object, and the clutter objects that are already present on this supporter. Specifically, denoting $Y = \{Y_i\}$ as the set of all available clutter types, the Clutterpalette evaluates the following probability for each clutter type Y_i:

$$P(x = Y_i | w, s, \{n_j\}), \tag{1}$$

where x is the hypothetical new clutter item type, w is the scene type ('kitchen', 'bedroom', etc.), s is the supporting object type ('floor', 'desk', 'wall', etc.), and n_j is the number of clutter objects of type Y_j that are already present on s, for each type Y_j for which $n_j > 0$. Note that Y_j may be equal to Y_i; for example, when the Clutterpalette considers the probability of adding a second monitor on a desk. To evaluate the probability (1) in terms of empirical statistics that we can extract from the training data, we use the naive Bayes model

$$\begin{aligned} &P(x = Y_i | w, s, \{n_j\}) \\ &\quad = \frac{P(x = Y_i) P(w, s, \{n_j\} | x = Y_i)}{P(w, s, \{n_j\})} && (2) \\ &\quad \propto P(x = Y_i) P(w, s, \{n_j\} | x = Y_i) && (3) \\ &\quad = P(x = Y_i) P(w | x = Y_i) P(s | x = Y_i) \\ &\qquad \prod_j P(n_j | x = Y_i). && (4) \end{aligned}$$

Equation (2) follows from Bayes' theorem. Equation (3) holds because we are interested only in the relative ordering of the likelihoods for different clutter types Y_i, and $P(w, s, \{n_j\})$ does not vary with i. Equation (4) holds by the naive Bayes assumption. We now describe how we estimate each of the probabilities in (4).

Prior $P(x = Y_i)$. There are several ways to approximate the prior, the simplest being a uniform approximation. We use a more informed empirical prior. To estimate $P(x = Y_i)$, we enumerate all clutter instances from all classes in all input scenes and compute the fraction that belong to class Y_i.

Conditional scene probability $P(w|x = Y_i)$. We estimate the conditional scene probability by enumerating all clutter instances from class Y_i in the training set and computing the fraction that occur in scenes of type w:

$$P(w | x = Y_i) = \frac{P(w, x = Y_i)}{P(x = Y_i)}. \tag{5}$$

Fig. 7a illustrates the effect of the scene probability on the presented suggestions.

Conditional supporter probability $P(s|x = Y_i)$. Similarly, we estimate the conditional supporter probability by enumerating all clutter instances from class Y_i in the training set and computing the fraction that occur on supporters of class s:

$$P(s | x = Y_i) = \frac{P(s, x = Y_i)}{P(x = Y_i)}. \tag{6}$$

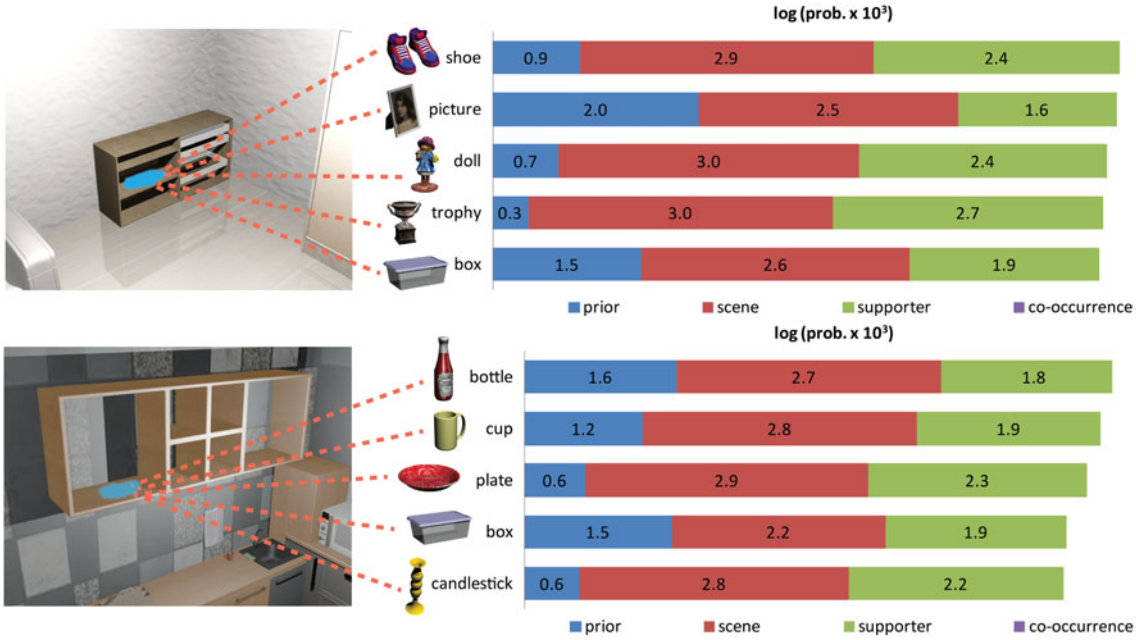
Fig. 7b illustrates the effect of the supporter probability.

Co-occurrence probability $P(n_j|x = Y_i)$. This is the probability that there are currently at least n_j instances of class Y_j on the given supporter, conditioned on x being a new instance of class Y_i. Note that Y_j can equal Y_i, in which case the instance x is not included in the number n_i; thus, n_i is the number of other instances from class Y_i that are already on the supporter. Note also that $P(n_j|x = Y_i)$ is interpreted as the probability that there are at least n_j instances of class Y_j on the current supporter (not counting x), because when we consider adding x to the scene, we do not know what other clutter objects will be subsequently added by the user. Fig. 7c illustrates the effect of the co-occurrence probability.

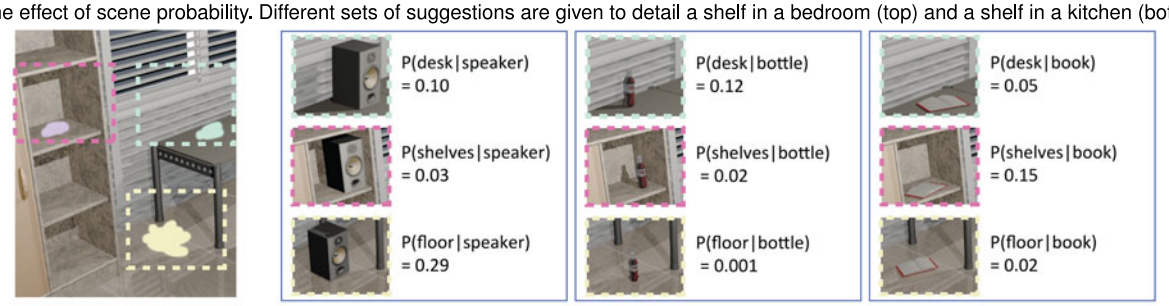
We can evaluate $P(n_j|x = Y_i)$ by enumerating all supporters in which at least one instance of Y_i appears and computing the fraction of these supporters in which at least n_j instances of Y_j appear. In the special case of $Y_j = Y_i$, we take the fraction of the scenes in which at least $n_i + 1$ instances of Y_i appear. Only pairs of clutter objects within a distance threshold of 1.5 meters from each other are considered in the co-occurrence estimation (both at runtime and during training). This is done in order to deal with larger supporters, such as floors, which can cover a large area with numerous clusters of clutter.

To alleviate data fragmentation, we quantize n_j to three possible values: 'zero', 'some', and 'many'. To determine the quantization boundaries, in a preprocessing step we enumerate the objects of class Y_j that occurred together on individual supporters in all scenes in the training set. Let m_j be the median of this set. We set n_j to 'zero' if there are no items of class Y_j on the current supporter, to 'some' if there are from 1 to m_j such items, and to 'many' if the number of such items is greater than m_j.

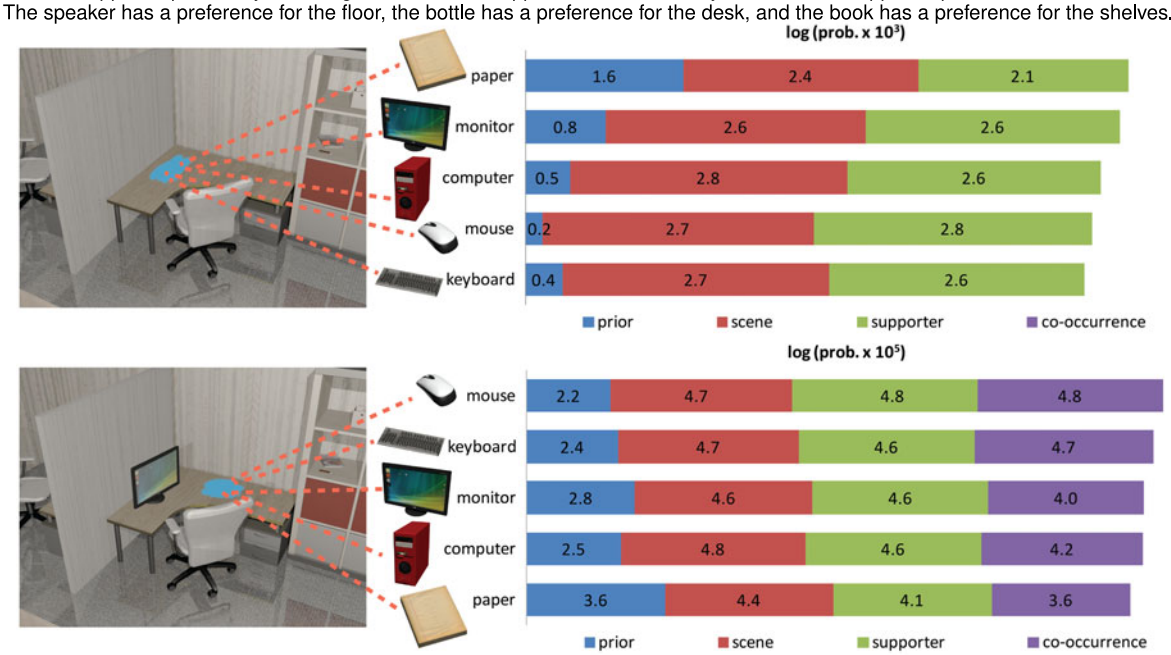
Precomputation. In practice, the probabilities on the right-hand side of (4) can be precomputed, by enumerating all



(a) The effect of scene probability. Different sets of suggestions are given to detail a shelf in a bedroom (top) and a shelf in a kitchen (bottom).



(b) The effect of supporter probability. Detailing three different supporters in the scene yields different supporter probabilities for the different clutter objects. The speaker has a preference for the floor, the bottle has a preference for the desk, and the book has a preference for the shelves.



(c) The effect of co-occurrence probability. When the desk is first detailed (top), the paper and monitor are the most likely clutter types. After a monitor is placed (bottom), the mouse and keyboard become the most likely clutter types to add.

Fig. 7. The effect of different probability terms on clutter type likelihood.

possible combinations of object types and scene types available in the training data. The probabilities are stored in matrices. For example, in the matrix of conditional scene probabilities, each entry represents the probability of a scene type given an object type. During the interaction stage, (4) can be evaluated by quickly retrieving the

Fig. 8. Modeling experiments. (a) Initial scenes given to the participants. (b) Detailed scenes produced by the participants using the Clutterpalette.

corresponding probabilities stored in the matrices and multiplying them together to compute $P(x = Y_i|w, s, \{n_j\})$.

Note that we use uniform small-sample correction in all empirical estimates to avoid zeroing out the probabilities. This is done by adding a default, small observation count σ when estimating each of the probabilities on the right-hand side of (4). We use $\sigma = 1$ in our experiments.

7 Performance and Evaluation

The Clutterpalette was implemented in C++, and our experiments were run on a 3.33 GHz Intel Xeon machine with 12 GB of RAM. In a preprocessing training stage, the required empirical statistics were calculated in a total of about 9 minutes with an unoptimized Matlab implementation. Thanks to the closed-form formulations, a probability of the form (1) is computed in less than 0.1 milliseconds at runtime. Therefore, the Clutterpalette presents its suggestions almost instantaneously, as is demonstrated in the accompanying video, which also includes a demonstration of the interactive detailing of an office scene using the suggestions provided by the Clutterpalette.

To evaluate the utility of the Clutterpalette, we recruited 20 university students and staff, including 12 male and eight female volunteers, to detail indoor scenes.

Fig. 9. Screenshot of the interface of a commercial interior design software system, AutodeskHomestyler, which uses a conventional clutter-object selection menu. To detail a scene, the user manually chooses objects from an alphabetical list.

All participants were frequent computer users and most reported some experience in using simple visual authoring interfaces intended for the general public, such as those used in video games or in photo editing applications.

Each participant was given a brief 5-10-minute tutorial on the Clutterpalette interface. We then let the participants experiment freely with the interface until they felt comfortable with its functionality. Our simple interface is demonstrated in the accompanying video: It allows users to add, delete, move, and rotate clutter objects. All participants reported becoming comfortable with the functionality within 15 minutes.

Each participant was assigned one of the scene types—bedroom, kitchen, living-room, office, or classroom. The participant was shown several images of real-world scenes of this type collected from Google Images. The participant was then asked to use the Clutterpalette to create a realistic scene of the given type, starting from an initial scene containing only the basic furniture. The initial scenes, which we had manually created, are shown in Fig. 8a.

The participants used the Clutterpalette to detail the scene to a level of realism with which they were satisfied. No time limit was imposed; each participant decided on their own when they were done. Modeling times ranged from 6 to 25 minutes. We assigned each participant randomly to one of the following two settings:

In Setting 1, the Clutterpalette presented suggestions ranked according to the model described in Section 6. The participants could alternatively browse in a separate tab an alphabetical list of available clutter types and choose objects from that list. Each clutter type was accompanied with a thumbnail to ease browsing. The participants could easily toggle between the two tabs.

In Setting 2, the Clutterpalette presented only the alphabetical list. The suggestion engine described in Section 6 was deactivated. Note that this setting is similar to conventional clutter object selection menus used in interior design or game level design software.[1] Fig. 9 shows an example.

TABLE 2
Average Time (in Seconds) Spent on Different User Interactions between Object Additions in Setting 1 (Suggestions + Alphabetical) and Setting 2 (Alphabetical Only)

	Navigate	Select	Move	Rotate	Total
Setting 1	8.95	7.72	3.82	1.76	22.25
Setting 2	9.05	17.36	4.94	1.82	33.17

The participants were not informed about the different settings. Ten scenes were produced in each of the two settings. The number of scenes of each type was the same for both settings. The scenes produced in Setting 1 (suggestions & alphabetical) are shown in Fig. 8b.

7.1 Modeling Time

In Setting 1 (suggestions & alphabetical), participants added 34.1 objects to the scene on average, whereas in Setting 2 (alphabetical only) they added 30.8 objects on average. In Setting 1, the average time between object addition was 22.25 seconds, whereas in Setting 2 it was 33.17 seconds. Our suggestion engine thus resulted in a 33 percent improvement in modeling speed.

Table 2 shows the average time spent on different user interactions between object additions—navigating the scene, selecting an object to add, translating an added object, and rotating an added object. Users tended to select objects faster in Setting 1 than in Setting 2. In Setting 1, users also tended to spend slightly less time on moving objects. This can be due to the fact that in Setting 2 some users tended to browse through the alphabetical list to search for a relevant object to add to the scene without caring about the location of the object in the first place. After adding the object, the user would then drag it to a desirable location in the scene. Therefore more time is needed for moving the object. By contrast, in Setting 1 when a user points to a location to add objects, the Clutterpalette makes suggestions that already fit within the scene context; hence, comparatively less time is needed for moving objects.

Compared to the other user interactions, rotating objects takes less time on average. This may be due to the fact that the objects are already in an upright orientation, and because many clutter objects (e.g., bowls) lack any frontal orientation and need not be rotated.

7.2 Usage Pattern

For each of the 10 users in Setting 1 (suggestions + alphabetical), Fig. 10 indicates the percentage of time that the Clutterpalette's suggestions were used relative to the alphabetical list. Clearly, users had a strong preference for employing the Clutterpalette's suggestions; on average, the suggestions were used 87 percent of the time, while the alphabetical list was used only 13 percent

1. Examples include AutodeskHomestyler, IKEA Home Planner, Asset Preview for Unity, etc.

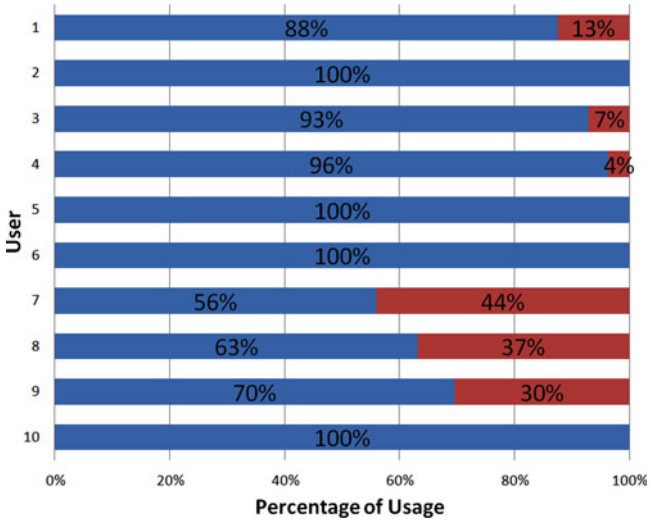


Fig. 10. Usage of the Clutterpalette suggestions (blue) versus the alphabetical list (red) by each user in Setting 1.

of the time. Table 3 further reports the usage percentages of the suggestions in each ranking. When users decided to add objects from the suggestion menu, they usually (77 percent of the time) chose the suggestions ranked within the top 5. Note that suggestions ranked from 6 to 10 are still relevant to the scene context, albeit with lower probabilities.

7.3 Scene Realism

We performed two experiments to evaluate the relative realism of the initial scenes, the scenes produced by users in Setting 1, and the scenes produced by users in Setting 2.

In the first experiment, our goal was to evaluate the relative realism of the scenes before and after detailing via the Clutterpalette. To this end, we recruited through social media a separate group of 25 evaluators. The evaluators were shown pairs of images. Each evaluator had to indicate which image in each pair is the more realistic, or to indicate no preference. The images in each pair were shown side by side and the left-right order was randomized. Each pair of images was of the same scene type. One image showed a scene produced by participants in Setting 1 while the other showed the scene in its initial condition (Fig. 8a). The evaluators were not told how the scenes were produced and in what way they were related to each other. In total, 500 pairwise comparisons were performed. Table 4 summarizes the results of this experiment. The evaluators overwhelmingly preferred the scenes detailed using the Clutterpalette. In informal exit interviews, the evaluators who voted for the uncluttered scenes remarked that empty classrooms are often encountered in the real world, e.g., when class is not in session, so they regarded empty classroom scenes to be realistic.

In the second experiment, we evaluated the relative realism of scenes produced in Setting 1 (suggestions + alphabetical) and scenes produced in Setting 2 (alphabetical only). We recruited through social media a separate group of 32 evaluators, distinct from the prior groups of participants and evaluators. The evaluators performed

TABLE 3
Usage of the Suggestions for Each Ranking in Setting 1

Ranking	1	2	3	4	5	6 to 10
Usage	28.07%	21.05%	12.28%	7.02%	8.77%	22.81%

TABLE 4
Independent Evaluators Comparing the Realism of Scenes Before and After Detailing Using the Clutterpalette

Preferred Before Detailing	Preferred After Detailing	No Preference
4.1%	94.6%	1.3%

TABLE 5
Independent Evaluators Comparing the Realism of Scenes Created in Setting 1 and Setting 2

Setting 1 Preferred	Setting 2 Preferred	No Preference
47.6%	36.5%	15.9%

randomized pairwise image comparisons. One of the images of each pair was of a scene created in Setting 1 and the other was of a scene of the same type created in Setting 2. The evaluators performed 1,280 comparisons, summarized in Table 5. According to a two-tailed t-test, the evaluators had statistically significant preference for the scenes created in Setting 1 ($p-\text{value} < 0.05$).

8 Discussion and Future Work

Real-world data and user-created content are becoming increasingly available. Such "big data" creates good opportunities for devising data-driven tools for computer graphics applications. A key idea in our work is training the Clutterpalette on a dataset of real-world scenes in order to address the problem of modeling virtual indoor scenes. We expect this idea to have broader applicability in training a variety of intelligent visual authoring tools based on the semantics of real-world scenes.

In the application of such data-driven approaches, annotation is often necessary to exploit the full potential of the training data. The NYU Depth Dataset [21], from which we selected five scene types for our experiments, provides annotated data for 26 common indoor scene types with substantial amounts of annotation collected by leveraging manual human effort through the Amazon Mechanical Turk. In principle, our model can be trained through automatic annotation approaches. The automatic annotation of big datasets—for example, the automatic semantic labeling of indoor scenes [22]—is an actively researched topic. Recent research suggests that one can simultaneously infer the furniture types and scene types through a top-down/bottom-up inference algorithm [23], [24]. Our approach can potentially incorporate such an algorithm so that, rather than being preset, the scene type of the room being detailed can be automatically inferred from the types of objects being added by the user.

We believe that there remain many possibilities for further developing interactive tools for enhancing the detail of scenes. In our current approach, we operate on a purely semantic level. While this enables our Clutterpalette to be trained on sparsely-annotated images that are easy to acquire for a variety of domains, the lack of geometric information is a major drawback. In its current level of development, our Clutterpalette does not reason about the careful placement or orientation of objects, thus necessitating user effort that can be demanding on a large scale. The tool also does not take into account the appearance of objects. It would be interesting to investigate scalable approaches to training models that can also reason about finer-grained geometric properties, orientation properties, and aesthetic qualities from real-world scenes.

As Table 2 shows, a considerable amount of time is usually spent navigating the scene in search of a suitable region for detailing. A semi-automated technique for reasoning about and suggesting good regions to detail will make an interactive modeling tool even more convenient.

9 Conclusion

We introduced the Clutterpalette, an interactive tool for enhancing the level of detail of indoor scenes. Our experiments demonstrate that people find scenes detailed with objects to be substantially more realistic than their stark counterparts. The adaptive suggestion generation functionality of the Clutterpalette was found to facilitate the creation of realistic scenes and improve modeling time. We expect the Clutterpalette to be useful in a variety of scene modeling applications, including video game level design interfaces, home modeling and remodeling applications, and virtual set-dressing for use in motion pictures.

Acknowledgments

The authors would like to thank Vladlen Koltun for discussing ideas, Ariel Shamir and Qiming Hou for their valuable comments, Yibiao Zhao for helping with the evaluation, and Michael S. Brown for narrating the demonstration video. This research was supported in part by Singapore University of Technology and Design (SUTD) StartUp Grant ISTD 2011 016, by SUTD-MIT International Design Center Grant IDG31300106, Singapore MOE Academic Research Fund MOE2013-T2-1-159, National Research Foundation, Prime Minister's Office, Singapore, under its IDM Futures Funding Initiative, administered by the Interactive and Digital Media Programme Office, and by University of Massachusetts Boston StartUp Grant P20150000029280 and by the Joseph P. Healey Research Grant Program provided by the Office of the Vice Provost for Research and Strategic Initiatives & Dean of Graduate Studies of the University of Massachusetts Boston. The work reported herein was done in part when L.-F. Yu was at the University of California, Los Angeles, and at the Singapore University of Technology and Design. L.-F. Yu is the corresponding author.

References

[1] M. E. Newell and J. F. Blinn, "The progression of realism in computer generated images," in *Proc. Annu. Conf.*, 1977, pp. 444–448.

[2] K. Xu, J. Stewart, and E. Fiume, "Constraint-based automatic placement for scene composition," in *Proc. Graph. Interface*, May 2002, pp. 25–34.

[3] Set dresser. Wikipedia: The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Set_dresser, Feb. 2014.

[4] R. W. Bukowski and C. H. Séquin, "Object associations: A simple and practical approach to virtual 3d manipulation," in *Proc. Symp. Interactive 3D Graph.*, 1995, pp. 131–138.

[5] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun, "Interactive furniture layout using interior design guidelines," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 87:1–87:10, Jul. 2011.

[6] L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher, "Make it home: Automatic optimization of furniture arrangement," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 86:1–86:12, Jul. 2011.

[7] M. Fisher and P. Hanrahan, "Context-based search for 3d models," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 182:1–182:10, Dec. 2010.

[8] M. Fisher, M. Savva, and P. Hanrahan, "Characterizing structural relationships in scenes using graph kernels," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 34:1–34:12, Jul. 2011.

[9] Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan, "Synthesizing open worlds with constraints using locally annealed reversible jump mcmc," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 56:1–56:11, Jul. 2012.

[10] M. Fisher, D. Ritchie, M. Savva, T. A. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3d object arrangements," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 135:1–135:11, Nov. 2012.

[11] K. Xu, K. Chen, H. Fu, W.-L. Sun, and S.-M. Hu, "Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 123:1–123:15, Jul. 2013.

[12] L. Majerowicz, A. Shamir, A. Sheffer, and H. H. Hoos, "Filling your shelves: Synthesizing diverse style-preserving artifact arrangements," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 11, p. 1, Oct. 2013.

[13] T. Igarashi and J. F. Hughes, "A suggestive interface for 3d drawing," in *Proc. 14th Annu. ACM Symp. User Interface Softw. Technol.*, 2001, pp. 173–181.

[14] S. Chaudhuri and V. Koltun, "Data-driven suggestions for creativity support in 3d modeling," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 183:1–183:10, Dec. 2010.

[15] S. Chaudhuri, E. Kalogerakis, L. J. Guibas, and V. Koltun, "Probabilistic reasoning for assembly-based 3d modeling," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 35:1–35:10, Jul. 2011.

[16] M. G. Chajdas, S. Lefebvre, and M. Stamminger, "Assisted texture assignment," in *Proc. ACM SIGGRAPH Symp. Interactive 3D Graph. Games*, 2010, pp. 173–179.

[17] A. Jain, T. Thormählen, T. Ritschel, and H.-P. Seidel, "Material memex: Automatic material suggestions for 3d objects," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 143:1–143:8, Nov. 2012.

[18] N. Umetani, T. Igarashi, and N. J. Mitra, "Guided exploration of physically valid shapes for furniture design," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 86:1–86:11, Jul. 2012.

[19] F. Bao, D.-M. Yan, N. J. Mitra, and P. Wonka, "Generating and exploring good building layouts," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 122:1–122:10, Jul. 2013.

[20] L.-F. Yu, S. K. Yeung, D. Terzopoulos, and T. F. Chan, "Dressup!: Outfit synthesis through automatic optimization," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 134:1–134:14, Dec. 2012.

[21] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. 12th Eur. Conf. Comput. Vis.-Volume Part V*, 2012, pp. 746–760.

[22] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 564–571.

[23] Y. Zhao and S.-C. Zhu, "Scene parsing by integrating function, geometry and appearance models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3119–3126.

[24] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese, "Understanding indoor scenes using 3d geometric phrases," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 33–40.

Lap-Fai Yu received the BEng (first class honors) and MPhil degrees in computer science from the Hong Kong University of Science and Technology (HKUST), in 2007 and 2009, respectively, and the PhD degree in computer science from the University of California, Los Angeles (UCLA), in 2013. He is an assistant professor at the University of Massachusetts, Boston. He received the Cisco Outstanding Graduate Research Award at the University of California, Los Angeles (UCLA). He was a postdoctoral research fellow at the Singapore University of Technology and Design (SUTD), a visiting scholar at Stanford University, and a visiting scientist at the Massachusetts Institute of Technology (MIT). His research interests include computer graphics, computer vision, and computational photography. He served as a program co-chair of the *Workshop on Vision Meets Cognition: Functionality, Physics, Intentionality and Causality (FPIC)*, organized in conjunction with the IEEE CVPR 2014 Conference. He is a member of the IEEE.

Sai-Kit Yeung received BEng degree (first class Honors) in computer engineering in 2003 and the MPhil degree in bioengineering in 2005 from the Hong Kong University of Science and Technology (HKUST), and the PhD degree in electronic and computer engineering from the HKUST, in 2009. He is an assistant professor at the Singapore University of Technology and Design (SUTD), where he leads the Vision, Graphics and Computational Design (VGD) Group. He was also a visiting assistant professor at MIT and Stanford University. Before joining SUTD, he was a postdoctoral scholar in the Department of Mathematics, University of California, Los Angeles (UCLA). He was also a visiting student at the Image Processing Research Group at UCLA in 2008 and at the Image Sciences Institute, University Medical Center Utrecht, the Netherlands, in 2007. His research expertise is in the areas of computer vision and computer graphics. His current research focus is on scene acquisition, scene understanding, functional realistic scene remodeling, and computational fabrication. He is a member of the IEEE and the IEEE Computer Society.

Demetri Terzopoulos, graduated from McGill University in honours electrical engineering and received the PhD degree in EECS from the Massachusetts Institute of Technology (MIT) in 1984. He is a professor of computer science at the University of California, Los Angeles, he is the distinguished professor and directs the UCLA Computer Graphics & Vision Laboratory. Among his many awards are an Academy Award for Technical Achievement from the Academy of Motion Picture Arts and Sciences for his pioneering work on physics-based computer animation, and the inaugural Computer Vision Distinguished Researcher Award from the IEEE for his pioneering and sustained research on deformable models and their applications. ISI and other indexes list him among the most highly-cited authors in engineering and computer science, with more than 300 published research papers and several volumes, primarily in computer graphics, computer vision, medical imaging, computer-aided design, and artificial intelligence/life. He has given hundreds of invited talks around the world about his research, including more than 100 distinguished lectures and keynote/plenary addresses. Prior to joining UCLA in 2005, he held the Lucy and Henry Moses Endowed Professorship in Science at New York University and was a professor of computer science and mathematics at the NYU's Courant Institute of Mathematical Sciences. Previously, he was a professor of computer science and electrical and computer engineering at the University of Toronto. Before becoming an academic in 1989, he was a program leader at Schlumberger corporate research centers in California and Texas. He is or was a Guggenheim fellow, a fellow of the ACM, IEEE, Royal Society of Canada, and Royal Society of London, and a Member of the European Academy of Sciences, the New York Academy of Sciences, and Sigma Xi.