MATTHEW FISHER RESEARCH STATEMENT

Supporting creative tasks with computational tools is challenging because desirable solutions cannot be delineated by simple algorithms or rules. Performing a task such as "create a 3D scene of a suburban house" requires the answer to thousands of questions such as "what objects belong in a refrigerator?" or "how should the objects on this desk be arranged?". *Explicitly* specifying the answer to all these questions is rarely tractable. The central idea of my research is that we can *implicitly* learn the answer to many of these questions by studying a relevant knowledge database, such as a database of 3D houses or a collection of photographs. We can use this knowledge to make the original task easier: when performing frequent actions, such as arranging the objects on a desk, we can assist the user by borrowing from the collective knowledge and creativity of everyone who has contributed to the underlying database.

My research applies techniques from computer vision and machine learning to address the challenge of supporting these creative tasks. We develop solutions to fundamental problems such as comparing arrangements of objects. These solutions enable useful tools such as an example-driven generative model for scenes that can produce diverse, plausible content. Moving forward, I am applying these techniques to a broader spectrum of problems, such as using a database of examples to learn how to play a game.

USING LARGE DATABASES TO UNDERSTAND 3D SCENES

When performing creative tasks, humans draw from their experiences to create content that is both functional and aesthetically pleasing. Unfortunately, computers lack such experience, without which concepts such as functionality are difficult to express. To overcome this lack of experience, researchers are developing ways to learn such concepts from a database of examples. The area I have chosen to focus on is the modeling of large scenes such as houses, restaurants, and schools, a domain we refer to as *scene modeling*. My long-term research goal is to apply database-driven scene understanding to augment or even partially automate creative endeavors.

To explore example-based learning for scene modeling, I started by developing a system that takes a location in a scene the user is modeling and answers the question, "what objects belong at this location?" [1]. This system is backed by the Google 3D Warehouse database, a large online collection of 3D scenes. Responding to such queries requires answers to a surprisingly diverse set of questions:

- How can we segment the input scenes into meaningful objects?
- What is the best way to compare two different objects? How do we decide the relative weight of different types of similarity, such as geometric similarity versus textual similarity?
- How do we compare two different arrangements of objects to see what parts of the database are most likely to contain relevant results?
- What is the right way to evaluate our search results and compare them against existing model search techniques?

My work is the first to develop approaches to several of these questions and apply them to scene modeling. In creating this system, two important lessons were learned. First, it is possible: properly processed, existing scene databases can serve as a proxy for a human modeler's real-life experience. Second, it is hard: automatic segmentations are very error prone, it is hard to compare models in the absence of high-quality tags, categories, or geometric features, and the distance between two sets of objects is ill-defined. Nevertheless, this initial foray into the domain was promising and served as an important foundation for all our future work.

Moving forward, my group took a hard look at the question of how to compare collections of objects. This is a fundamental operation and has many analogs to data-comparison problems in fields such as computer vision and computational biology, where researchers need to compare regions from two different images or binding sites on two different proteins. Our work applies ideas from these fields to 3D scene understanding by reducing the input scenes to *relationship graphs* and applying a graph kernel technique adapted from computer vision to compare two such graphs [2].

A relationship graph encodes the models in a scene and the semantic relationships between them. This format represents relationships such as whether one object is "on top of" or "to the side of" another object. Graph kernels present an efficient algorithm based on dynamic programming that can compare two such relationship graphs, and our work shows how this approach can be applied to a variety of practical problems. A simple application is scene retrieval, where scenes in the database are ranked according to how similar their object arrangements are to an input scene. Figure 1 shows an example of another application, where our system allows the user to request models with a specific relationship to other objects in the scene.

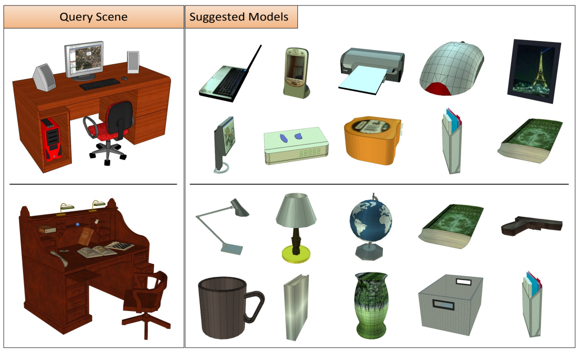

Figure 1: Graph kernels defined over 3D scenes can answer queries such as "what object belong on top of this desk I am modeling?"

APPLYING SCENE UNDERSTANDING TO MODELING

Creating large scenes is a time-consuming and multi-stage task. It requires repeatedly modeling, searching for, placing, scaling, and orienting hundreds to thousands of objects. To alleviate the tedium of this task, my group applied our database-driven scene understanding approach to develop a system that synthesizes object arrangements from examples [3]. This method can fully automate the generation of scenes such as desks or dining tables from as few as four examples of the desired type.

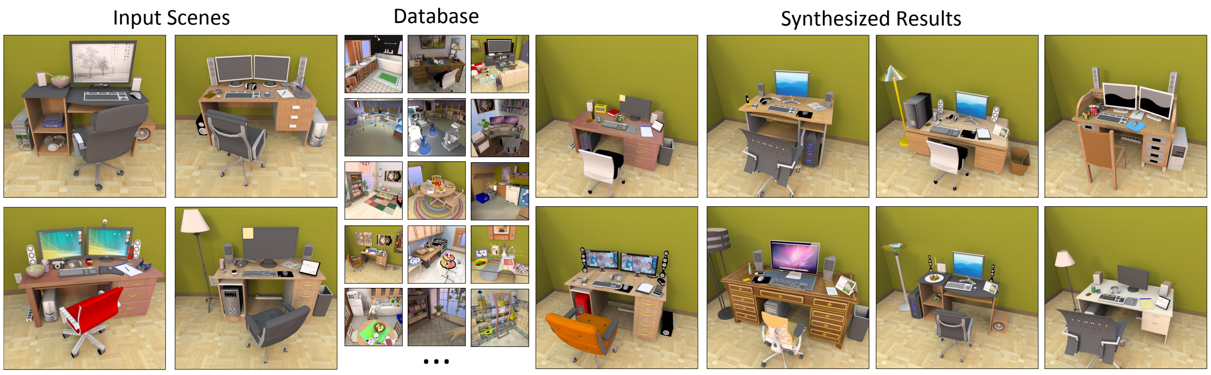

Figure 2: An artist provides the four example desks on the left, and our algorithm generates many plausible scenes of a similar type, incorporating object composition and arrangement information from a database of 3D scenes to increase the variety of its results.

We define a probabilistic model over scenes, drawing from ideas in probabilistic graphical models, and used it to learn a distribution from examples which we sample to generate diverse and plausible results. We train our probabilistic model on a mix of user-provided examples and relevant scenes retrieved from the database, relying upon our graph kernel scene comparison technique to understand which entries in the database contain the most relevant information [2]. Figure 2 shows several scenes sampled from our model when trained on four desks. When humans evaluate the quality of our results at least 80% of synthesized scenes are not distinguishable from hand-created scenes. This has very promising ramifications for reducing the time it takes to make this type of content — our example scenes each took around 15 minutes to make by hand, but our system can generate plausible results in only a few seconds.

UPDATING THE GRAPHICS PIPELINE

Looking beyond scene modeling, I have successful research experiences in a variety of areas in computer graphics, ranging from geometry [4] to graphics pipeline design [5]. Graphics covers a large number of topics, and having a deep understanding of different areas can be rewarding: it is often surprising how changes in one area can have significant ramifications on other seemingly unrelated areas. One area I follow closely is the changing geometric representations used in graphics pipelines. The choice of geometric representation, ranging from point clouds to triangle meshes to vector-displaced subdivision surfaces, has dramatic implications on virtually every area of computer graphics, especially content generation and modeling.

My interest in the graphics pipeline started long ago through a series of internships with the Direct3D kernel team at Microsoft. This team has a unique perspective and influence on the graphics pipeline and, in collaboration with teams at NVidia and ATI, developed the specification for the modern hardware patch tessellation system. Although the D3D11 real-time tessellation system is efficient, for many practical scenes it produces tessellations that are significantly lower quality than those produced by offline rendering systems such as the Reyes pipeline used by Pixar's RenderMan. Specifically, D3D11 tessellations often under-tessellate the input surface (degrading image quality) or over-tessellate it (degrading performance).

My knowledge of Direct3D and my contacts in the industry proved invaluable when my group at Stanford sought to overcome the limitations of existing tessellation methods. We created a tessellation system called *DiagSplit* that is fit for implementation in the real-time graphics pipeline and matches the quality of Reyes tessellations [5]. The key insight which enables efficient parallelism is to allow patch splits to occur along non-isoparametric domain lines; we refer to such edges as diagonal edges, giving rise to the algorithm's name. This allows the split patches to independently make consistent tessellation decisions about the shared edge without the need for complex inter-patch communication and without restricting all splits to be a power-of-two (a tessellation scheme we call *BinarySplit*, which results in significant overtessellation). Figure 3 visualizes the average micropolygon area of these algorithms on three different scenes. An ideal micropolygon tessellation would be entirely green, and in each case our algorithm matches or outperforms all existing approaches in both performance, quality, and ease of parallelization.

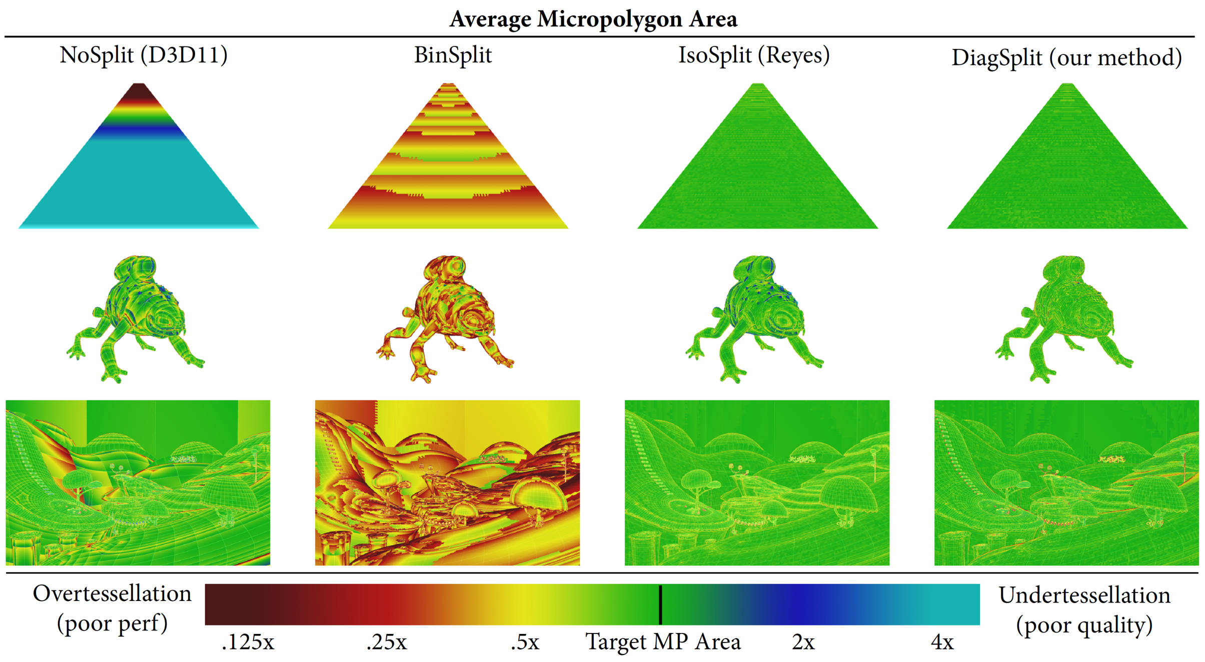

Figure 3: DiagSplit can be easily implemented in a real-time graphics pipeline and also produces high quality tessellations that match the quality of offline rendering systems (Reyes). It avoids the overtessellation and undertessellation seen in existing high-performance tessellation algorithms (D3D11, binary splitting).

Since its publication, a variation on the DiagSplit approach to tessellation has been implemented in the rendering pipeline of Dreamworks Animation.

RESEARCH AGENDA

The central theme that motivates my current and future work is to take challenging tasks where humans rely heavily on experience to guide them, and show that we can develop algorithms that use examples as a proxy for such experience. To date, I have focused on this problem in the context of static scene modeling tasks. Moving forward, I am taking this idea beyond synthesizing stationary geometry. Real environments are dynamic — people and vehicles move, objects hanging over on the edge of a desk can easily fall down, and eggs placed at the bottom of a grocery bag can break. Environments are also functional — light switches turn on lights, power outlets provide electricity to computers, and fans that are powered and turned on provide airflow. Modeling such dynamism and functionality can steer the model away from unrealistic behavior and improve the richness and fidelity of the generated results.

I am excited to explore the idea of using examples to mimic experience in domains beyond scene modeling. Humans rely upon past experience to accomplish almost everything we do, whether it's driving a car or writing a novel. In many of these domains it is difficult to experiment (driving a car) or it is challenging to quantify success (writing a novel). The domain I am currently focusing on is learning games from examples. This domain strikes a balance between accessibility (games are closed systems where success can be evaluated and quantified) and complexity (games encode interesting and complex behavior that is challenging to capture). In *general game learning*, a computer has access to the frame buffer of a digital game and attempts to learn a model for the game through experimentation and observing examples of humans playing the game. A preliminary example-driven system I designed for the popular game Starcraft 2 demonstrates the feasibility of the idea. The project was very well received by the programming community, and the associated YouTube video has been watched over 80,000 times.

In sum, my research shows how a computer can learn from a database of examples, and apply that understanding to solve real problems. This research is relevant to a wide spectrum of domains and works towards making human experience accessible to machine intelligence.

REFERENCES

[1] **Matthew Fisher** and Pat Hanrahan. Context-Based Search for 3D Models. In Proceedings of *ACM SIGGRAPH Asia 2010.* Seoul, South Korea, 2010.

[2] **Matthew Fisher**, Manolis Savva, and Pat Hanrahan. Characterizing Structural Relationships in Scenes Using Graph Kernels. In Proceedings of *ACM SIGGRAPH 2011.* Vancouver, British Columbia, 2011.

[3] **Matthew Fisher**, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based Synthesis of 3D Object Arrangements. In Proceedings of *ACM SIGGRAPH Asia 2012.* Singapore, 2012.

[4] **Matthew Fisher**, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. Design of Tangent Vector Fields. In Proceedings of *ACM SIGGRAPH 2007.* San Diego, California, 2007.

[5] **Matthew Fisher**, Kayvon Fatahalian, Solomon Boulos, Kurt Akeley, Bill Mark, and Pat Hanrahan. DiagSplit: Parallel, Crack-Free, Adaptive Tessellation for Micropolygon Rendering. In Proceedings of *ACM SIGGRAPH Asia 2009.* Yokohama, Japan, 2009.