EUROGRAPHICS 2018
K. Hildebrandt and C. Theobalt
(Guest Editors)

Volume 37 (2018), Number 2
STAR – State of The Art Report

Functionality Representations and Applications for Shape Analysis

R. Hu[1] [†] [‡] M. Savva[2] [†] O. van Kaick[3] [†]

[1]Shenzhen University [2]Princeton University [3]Carleton University

Abstract

A central goal of computer graphics is to provide tools for designing and simulating real or imagined artifacts. An understanding of functionality is important in enabling such modeling tools. Given that the majority of man-made artifacts are designed to serve a certain function, the functionality of objects is often reflected by their geometry, the way that they are organized in an environment, and their interaction with other objects or agents. Thus, in recent years, a variety of methods in shape analysis have been developed to extract functional information about objects and scenes from these different types of cues. In this report, we discuss recent developments that incorporate functionality aspects into the analysis of 3D shapes and scenes. We provide a summary of the state-of-the-art in this area, including a discussion of key ideas and an organized review of the relevant literature. More specifically, the report is structured around a general definition of functionality from which we derive criteria for classifying the body of prior work. This definition also facilitates a comparative view of methods for functionality analysis. We focus on studying the inference of functionality from a geometric perspective, and pose functionality analysis as a process involving both the geometry and interactions of a functional entity. In addition, we discuss a variety of applications that benefit from an analysis of functionality, and conclude the report with a discussion of current challenges and potential future works.

CCS Concepts
•Computing methodologies $\rightarrow$ Shape analysis; Spatial and physical reasoning;

1. Introduction

One of the goals of computer graphics is to provide tools for designing and simulating real or imagined artifacts, such as man-made objects. Such artifacts are usually functional, and thus an understanding of functionality is paramount for simulating and validating different designs of artifacts. The latter two tasks are especially important for enabling the virtual prototyping and mass customization of artifacts in the context of fabrication, which has become increasingly popular in recent years. Moreover, a functional understanding can benefit not only the analysis of individual objects, but also of other types of geometric datasets, such as object parts or scenes composed of multiple objects. These observations behoove computer graphics researchers to work on representing, analyzing, and synthesizing the functionality of artifacts.

Functionality typically refers to the purpose of an object or, more specifically, how the object can be used to accomplish a specific goal, e.g., see Figure 1. Thus, understanding the functionality of an artifact may involve analyzing and understanding several qualities of the object, ranging from its geometry, to its interactions with other objects, to physical properties of the object. However, this also implies that the analysis of functionality can be a challenging problem, given the variability that exists in the geometry of artifacts, and the many factors that have to be taken into consideration to reveal the functionality of a geometric dataset (see Figure 2). Nevertheless, computer graphics is in a strategic position to provide key contributions to the analysis and representation of functionality, given the expertise of the community with a variety of topics such as geometric modeling, design tools, physical simulation, and applied machine learning.

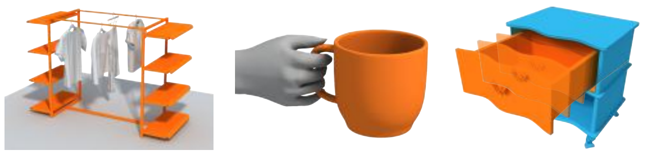
Figure 1: *Examples of objects with a variety of functionalities. From the left: coat rack, mug, nightstand drawer. The static and dynamic relations between the functional entity (in orange) and other entities characterize the functionality of the object or shape part.*

In recent years, there has been an increasing interest in considering high-level and structural information for the analysis of 3D geometric datasets of objects and scenes. These works have shown that going beyond pure geometry with the use of information at a

[†] All the authors have equal contribution.
[‡] Corresponding author: R. Hu (ruizhen.hu@gmail.com)

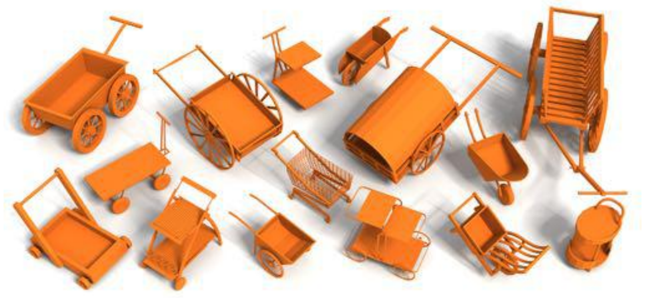

Figure 2: *Challenges in functionality analysis: all of the objects shown here possess similar functionalities, although they vary significantly in their geometry, or the arrangement of their parts.*

more semantic level can greatly benefit several applications in computer graphics, e.g., with structure-aware processing [MWZ*13] or data-driven shape analysis approaches [XKHK17]. Functionality is another form of semantic information, which goes beyond part labels and object structures. One could argue that the end-goal of several previous works involving semantics has been a functional understanding of the objects involved. Functionality-aware processing, the processing of geometric datasets while involving an understanding of functionality, is still a relatively new idea in graphics. However, interest in this topic has grown given the potential that a functional understanding of objects can have in facilitating computer graphics applications.

The goal of this report is to provide a comprehensive survey of functionality analysis in computer graphics, and discuss the state-of-the-art methods in the field. The report is structured around a general definition of functionality which we use to derive criteria for classifying the body of prior work, and for facilitating a comparative view of methods for functionality analysis. In this definition, we focus on the inference of functionality from the geometry of the datasets via shape analysis methods, which is the manner in which functionality analysis has been approached by most works in the literature. Note that we do not consider other important physical properties which are tied to functionality, such as the appearance and materials of the objects. We discuss this aspect of functionality in more detail in the perspective of future work.

Previous reports. Our focus domain is related to structure-aware and data-driven shape analysis approaches, which were covered by the reports of Mitra et al. [MWZ*13] and Xu et al. [XKHK17], respectively. The key difference of our report compared to these previous surveys is that we focus on classifying the types of functionality representations in the literature, with a particular focus on the extraction and representation of the functionality of objects. In addition, our report discusses recent functionality-centered works that have not received attention in these previous reports.

Functionality analysis in computer vision and robotics. In this report, we focus on functionality analysis works published mostly in the computer graphics literature, discussing methods that analyze *geometric datasets* such as 3D objects, object parts, and scenes composed of multiple objects. There has also been considerable work on functionality analysis in other areas of visual computing, such as computer vision and robotics. We include in our discussion the works from these fields that involve the use of 3D or depth information, such as RGB-D images. We consider the discussion of works that do not involve geometric datasets to be outside the scope of our report, for example, methods involving the analysis of digital images composed only of color channels without depth.

Nevertheless, for the reader interested in the use of functionality in these other contexts, our definition of functionality and derived classification scheme can also be used as a framework to study the related literature. For example, a class of methods for affordance analysis in RGB images combines the idea of human pose estimation and object detection [GSEH11, YFF12, YMFF13, ZFFF14], being similar to the agent-based (GA) methods discussed in our report. Another class of computer vision works follow a labeling approach, where input images are segmented and labeled with affordance labels [DR13, ZZ13, RT16], similarly to the geometry-only (G) methods in our report. A few methods combine human poses and affordance labels with additional knowledge such as natural language semantics, involving verb-noun relations [CWMD15], or encode the information as a knowledge graph [ZFFF14]. Finally, rather than using human pose to estimate affordances, some works focus on specific applications and employ pose information directly for scene understanding and reconstruction [ZZ13, FDG*14].

Organization. We first introduce our formal definition of functionality, which leads to the classification of previous works. We also discuss various definitions of functionality proposed in previous works, and show how they can all be seen as a special case of our general definition. We then frame our discussion of the works in the literature based on this induced classification scheme. Moreover, we present a series of applications that benefit from functionality-aware processing. Finally, we conclude the report with a discussion of current challenges and possible future works on functionality-aware analysis.

2. Definition of functionality and classification criteria

In common speech, *functionality* typically refers to the *use* or *purpose* of an object. For example, the Merriam-Webster dictionary defines *function* in layman terms as "*the action for which a person or thing is specially fitted or used, or for which a thing exists (purpose)*". In a more technical context, Bogoni and Bajcsy [BB95] define functionality as "*the application of an object in a specific context for the accomplishment of a particular purpose*". These definitions give an intuitive explanation of functionality as the practical *use* for which an object is intended or designed. However, such definitions do not serve well as a means of classifying technical works in terms of the aspects of functionality that they analyze. Here, we give a constructive definition of functionality, which more easily serves as a classification guide for existing works in the computer graphics literature.

2.1. Our definition of functionality

We start from the assumption that our goal is to understand the functionality of an *entity*, which we refer to as the *functional entity*.

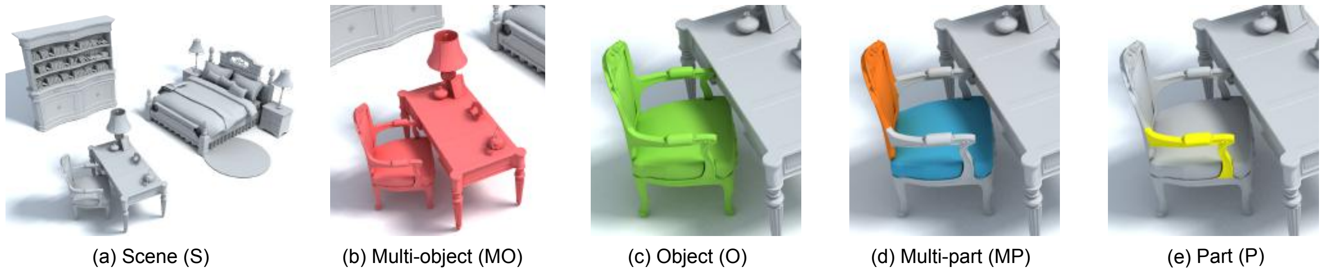

Figure 3: *Our analysis of functionality covers a spectrum of different levels of functional entities: entire scenes, multi-object regions, distinct objects, multi-part sets, and distinct parts.*

An entity can be an animate or inanimate object, e.g., a static object such as a table, a moving mechanism or part such as a sliding drawer, or an agent such as a human being. We define the functionality of the entity as:

$$\text{Functionality} = \text{geometry} + \text{interaction}. \tag{1}$$

Geometry refers to the shape or 3D spatial structure of the functional entity itself, given that the geometry of an object often provides valuable knowledge about the purpose of the object, as captured by the maxim "*form follows function*" [Sul96]. For example, a disk-like shape provides a cue that the object may be rotated and thus may be used as a wheel. However, geometry alone is often not sufficient to characterize the functionality of an entity. A concrete realization of functionality involves the *use* of the entity, which requires some form of interaction to occur between entities.

Thus, *interaction* in the definition refers to the interaction between the functional entity and other entities in a context, which we refer to as *interacting entities*. An interaction can involve any type of relation that provides cues on the use or functionality of the functional entity. Examples of relations are spatial proximity, static support, allowed relative motions, etc. Thus, the idea behind this definition of functionality is that we can obtain valuable knowledge about the functionality of an entity with an analysis that considers both the geometry of the entity and how it interacts with other entities in a relevant context.

We define the interaction component of the definition as a set of *atomic interactions*. We propose this compositional definition of interaction since recurring patterns of the functionality of different objects imply that there exists a set of atomic building blocks for functionality. These atomic building blocks are the individual interactions of the entity with each of the other entities in its context. Specifically, an atomic interaction is a relation between two entities which enables the functionality of the functional entity, where the relation can take a variety of forms. Thus, we denote an atomic interaction as a tuple of the form:

$$\text{Atomic interaction} = \langle\text{functional entity, relation, interacting entity}\rangle. \tag{2}$$

In the following subsections, we will discuss the components of these definitions in more detail, while also analyzing previous works on functionality according to the framework induced by this formalism. We will first give concrete examples of possible entities and relations for Eq. (2). Then, we will see how Eq. (1) serves as a common framework to understand and discuss previous works, also revealing unexplored areas of research. Note that, since much prior work in graphics has focused on the representation and analysis of geometry, our focus in this report will be on the representation and analysis of interaction, and its relation to functionality.

2.2. Atomic interactions

An atomic interaction is characterized by its entities and the relation between the entities. We classify entities and relations according to their characteristics as follows, and list them with the keywords and abbreviations that we use in Table 1.

Type of entity: relates mainly to the dynamics of the entity.

- **Static entity:** the entity is a rigid body that can be moved but usually does not deform during motion, e.g., a mug cup.
- **Dynamic entity:** the entity may deform or articulate when undergoing a motion, e.g., scissors while cutting paper.
- **Human(-oid) agent:** this is a special case of a static or dynamic entity that is typically the performer of actions. Since agents are an important component in some functionality models, we classify them separately from other types of entities. Note that, although humanoid agents are inherently dynamic, certain works may model them as static entities for practical purposes.

Level of entity: relates to the level of organization at which the entity appears, illustrated in Figure 3.

- **Scene:** the entity is a set composed of multiple objects, e.g., a living room with a couch, table, and TV. Note that a scene can be composed of multi-object entities which locally allow for different functionalities, e.g., a living room with a TV stand, a piano, and a dining table.
- **Multi-object:** the entity is a set of objects that are functionally connected and jointly used, e.g., the combination of an office desk, chair, and computer, which can be used for performing office tasks.
- **Object:** the entity is an entire object, e.g., a table.
- **Multi-part:** the entity is composed of multiple object parts, e.g., a combination of a seat and back of a chair that provide support for a human to sit.
- **Part:** the entity is an object part, e.g., table leg.

Type of relation: relates mainly to the dynamics of the relation.

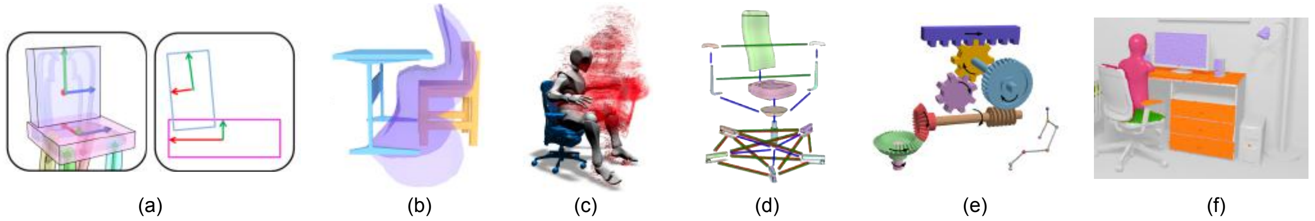

Figure 4: *Different representations of relations in interactions: (a) Spatial arrangement (SA) [FAvK*14], (b) Boundary representation (BR) [ZWK14], (c) Dense volume feature (VF) [PKH*17], (d) Gestalt and symmetry grouping (SG) [WXL*11], (e) Mechanical relations (MR) [MYY*10], (f) Humanoid actions (HA) [FSL*15].*

- **Static:** the relation between the functional and interacting entities does not change over time, e.g., a plate resting on a table.
- **Dynamic:** the relation changes over time time, e.g., someone using a cup to drink a liquid, or scissors to cut a piece of paper. Note from these examples that the entities involved in dynamic relations can be static or dynamic.

Representation of the relation: categorizes how the relation between the entities is represented and what information is captured. Examples of relations are shown in Figure 4.

- **Spatial arrangement (SA):** includes spatial relations that capture the arrangement between the two entities. Examples include relations such as relative position, co-occurrence and adjacency, gravitational support, attachment, enclosure, and the Relation-augmented Image Descriptor (RAID) [GMW16].
- **Boundary representation (BR):** represents the boundary between the geometry of the two entities, to capture richer geometric information about the interaction when compared to simpler measures such as relative position. Examples include the Interaction Bisector Surface (IBS) [ZWK14], and the IBS combined with Interaction Regions (IR) [HZvK*15]. Note that the boundary between the entities in these works is in fact a portion of space between the entities, which can be empty when the entities are far apart, or which can possess an infinitesimal thickness when the entities are touching.
- **Dense volume feature (VF):** represents the space containing the two entities, capturing scalar or vector features of the interaction at a dense set of points in the containing volume. An example of this representation are Interaction Landscapes [PKH*17].
- **Gestalt and symmetry grouping (SG):** includes relations that capture translational and rotational symmetry groupings of the entities, and repeated geometric patterns formed by the entities. Much of the earlier work in shape analysis focuses on this relation type. A prominent example is the symmetry hierarchy representation [WXL*11], where the functional entities are defined at the part level.
- **Mechanical relations (MR):** includes relations such as force drivers (e.g., motors), joints, gears, and other dynamic mechanisms formed by the entities. These relations are common in methods focusing on fabrication of articulated objects [LOMI11, KLY*14], or visualization of mechanical assemblies [MYY*10, XLX*16].
- **Human(-oid) actions (HA):** the relations include actions such as gazing, grasping, holding, pushing, pulling, sitting, and lying down, which involve two entities. Note that the latter two are actions combined with forms of gravitational support. Action relations are the focus of human-centric analysis methods [SCH*14, FSL*15, SCH*16, MLZ*16].

2.3. Representations of functionality

Given the definition of functionality introduced in Eq. (1), we group models of functionality built by different methods into three categories, depending on the components of the definition that are taken into consideration. These categories are illustrated in Figure 5.

Geometry-only (G): only the geometry of the functional entity and possibly its structure are used to derive knowledge about the entity's functionality. Note that the analysis of the structure of an entity may involve the analysis of relations between subcomponents of the entity, e.g., an analysis of the relations among object parts to infer the functionality of an object. The relations between subcomponents are usually stored in a tree or graph structure, which allows one to establish a distance metric between region or scene contexts and retrieve or classify entities. One may argue that given our definition of functionality, such approaches do not really analyze functionality, since no form of interaction with entities at the same level is considered. However, we classify these works as a special case which implicitly falls within the scope of functionality analysis. Examples of works in this category include:

- **Object co-analysis:** methods that characterize the validity of objects based on the configuration of their parts, such as the meta-representation [FAvK*14] or co-constrained handles [YK14], where the geometry of an object and its part composition gives an indication of the validity of the object, in terms of the probability that it belongs to a certain class of objects. The learned validity of an object is closely related to functionality, although it can also include aesthetic properties of the objects.
- **Scene synthesis:** methods that represent scenes as graphs with objects at the nodes, and edges capturing spatial relations between the objects (e.g. graph kernels for 3D scene similarity estimation [FSH11]).

Geometry + interaction (GI): both the geometry and interaction with other entities are used to establish the functionality of the entity. The atomic interactions can be organized in different ways, to provide a summary of the entity's functionality:

- **Rooted tree:** similar to the tree or graph structure mentioned

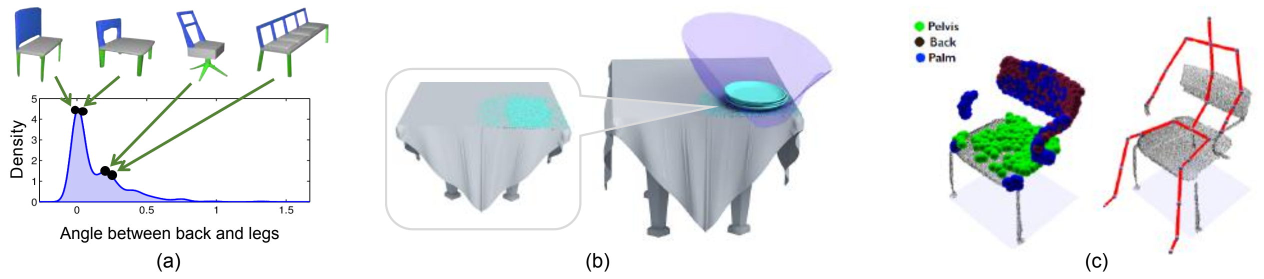

Figure 5: *The three categories of functionality representations, according to our generalized definition of functionality: (a) Geometry-only (G) [FAvK*14], (b) Geometry + interaction (GI) [HZvK*15], (c) Geometry + interaction with humanoid agent (GA) [KCGF14].*

above, but in this case the focus is on describing the functionality of a *single* functional entity and its interactions with other interacting entities at the same level. The functional entity corresponds to the *root* of the tree, and can be called *central* or *focal* object in some works. Examples include the use of objects as focal points [XMZ*14] or central objects in a context [HZvK*15].

- **Dynamic field:** the time-varying nature of a given atomic interaction is captured in a volume containing the interacting entities. For example, the Interaction Landscapes descriptor [PKH*17] represents the dynamic evolution of a relative position interaction as a scalar or vector field over the volume. Note that, as a special case, one of the interacting entities can be a human agent.

Geometry + interaction with human(-oid) agent (GA): we classify models that involve interactions with agents as a special case of the previous classification, given their importance in the analysis of functionality. Examples of approaches making use of interactions with agents include:

- **Agent-centric:** interactions are represented in terms of the structure of the acting agent. Most commonly, this implies modeling of the human pose at the body part level, and associating body parts to objects, e.g., the human pose representation used by PiGraphs [SCH*16].
- **Geometry-centric:** interactions are represented in terms of the geometry (typically the surface) of the objects with which the agent interacts. An example of this representation are the action maps or affordance maps used by SceneGrok [SCH*14].

2.4. Additional classification criteria

Besides the components of the functionality definition we discussed above, we consider some additional criteria that are helpful for classifying previous works.

Model type: type of model produced by the method, which is relevant to the target task. We classify the model type into one of:

- **Discriminative:** the model can be used to classify instances into one or more functionality categories, or to induce features and descriptors for classification or retrieval of functionality.
- **Generative:** the model can be used for classification but also to generate novel domain instances.

Approach: nature of the approach used by method to acquire knowledge on the problem, including different types of learning methods. We classify the approach as one of:

- **Supervised:** the method is data-driven, with the functionality-related knowledge being learned from training data.
- **Unsupervised:** the method is data-driven, but with knowledge being extracted in an unsupervised manner.
- **Handcrafted:** the knowledge used by the method is based on a handcrafted model, possibly consisting of a user-created rule-based system, and no learning is involved.

Input data representation: the encoding of the main type of input data handled by method. Note that we consider mainly approaches that use some type of 3D shape representation. We classify the representation into one of:

- **RGB-D image:** an image representing a single frame of a depth scanner input.
- **Point cloud:** an already-reconstructed point cloud, possibly arising from fused RGB-D frames or from sampling a mesh.
- **Mesh:** a polygonal mesh, including triangle soups and manifold meshes.

2.5. Example definitions of functionality

The general definition of functionality and terminology that we present above allows us to classify prior work and carry out a comparative discussion of existing models of functionality, independently of whether the models were explicitly defined in the works, or only implicitly used. Here, we provide a few prominent examples to illustrate the use of our classification scheme.

Slippage Analysis [GG04]: this method analyzes a shape to discover its slippable motions, that is, rigid motions that slide a transformed version of the shape against a stationary version of the shape without forming any gaps. The approach reveals portions of the shape that are kinematic surfaces, such as planes, cylinders, spheres, and surfaces of revolution. Although the method is used primarily for segmentation of the shape into primitive surfaces, the analysis reveals the slippable portions of the shape and the associated motions, which could be explored for inferring functional characteristics of the shape.

Following our classification criteria, this work proposes a geometry-only (**G**) method representing *static* symmetry relations (**SG**) of *static parts* of a shape, by modeling the slippage characteristics of the parts. In addition, the slippage analysis is based on a prespecified mathematical definition of the various surfaces. Thus, the approach is *handcrafted*, and is used *discriminatively* to provide segmentations of a shape as the output, taking *polygonal meshes* as input.

ICON [HZvK*15]: this method encodes the interactions between objects in a scene with the Interaction Bisector Surface (IBS) [ZWK14], which is a subset of the Voronoi diagram that captures the spatial boundary between the objects. In addition, portions of the central object that triggered the creation of the IBS are captured by the Interaction Regions (IR), which characterize the shape portions that are directly involved in the interactions. By organizing the interacting objects in a scene into a tree structure, according to the similarity of their IBSs and IRs, the representation can be used to compare the functionality of the central objects reflected in the corresponding context.

In our classification, we characterize this approach as a geometry + interaction **(GI)** method, representing the *static* boundary **(BR)** between the interacting objects and the central object, in a rooted tree. Thus, the method works at the *object* level, and is *discriminative* and *handcrafted*, applied to *point cloud* representations.

SceneGrok [SCH*14]: the method takes as input dense 3D reconstructions of real-world scenes and tracks people as they interact with the environment. This information is used to train a classifier that can predict the probable interactions that unobserved 3D scenes support, which are encoded as action maps over the scenes.

According to our classification, this is a geometry + human agent **(GA)** method representing spatial arrangement **(SA)** and human action **(HA)** relations between human agent entities and *static 3D scenes*, using the geometry-centric "action map" representation. The model is *discriminative* and trained in a *supervised* fashion from *polygonal mesh* representations of reconstructed 3D scenes.

In the following sections, we discuss individual methods that analyze the functionality of geometric datasets. We start the discussion by first examining works that are related to functionality analysis, but which do not exactly fit our definition as they do not explicitly aim to analyze the functionality of shapes. Next, we organize the discussion of the more closely-related approaches according to the three representations of functionality that we defined in Section 2.3.

3. Shape analysis works related to functionality

In recent years, several approaches that incorporate semantic considerations into the analysis of shapes have appeared, such as structure-aware [MWZ*13] and data-driven [XKHK17] approaches. Although these methods do not explicitly infer or create a model of the functionality of shapes, they can be seen as precursor works in this direction. Many of these approaches have objectives related to shape understanding, and form the building blocks of later methods that perform explicit functionality analysis. In this section, we discuss representative methods of this category, grouped by the main problems that they address, and explore the principles followed by these works that are also relevant to functionality analysis tasks.

3.1. Consistent part correspondence

A variety of techniques have been proposed for obtaining a consistent part correspondence of a set of shapes. The objective of these techniques is to segment the input shapes into *parts*, while at the same time establishing a *correspondence* among the parts of all shapes, so that parts that likely possess the same semantics are mapped to each other. For example, given a set of human models, the goal would be to segment the models into body parts, and assign a common label to similar parts, e.g., all hands get assigned the same numeric label. Different solution approaches were proposed to solve this problem, which can be roughly classified into methods that first extract candidate parts and then establish a correspondence among the parts (*segmentation* $\rightarrow$ *correspondence* methods), and methods that follow the inverse approach by first establishing correspondences between the entire shapes and then consistently segmenting the shapes into parts based on the correspondences (*correspondence* $\rightarrow$ *segmentation* methods).

In the *segmentation* $\rightarrow$ *correspondence* category, one group of methods is based on first performing an over-segmentation of the shapes using geometric criteria, to yield a partitioning of the shapes into candidate building blocks of parts, also referred to as *super-faces*, similar to super-pixels in computer vision. Next, a *co-analysis* is performed where the super-faces are represented by a set of shape descriptors and clustered according to the similarity of these descriptors. The clusters then provide the correspondence between parts, which can cause super-faces within the same cluster that are adjacent on a shape to be fused together into larger semantic parts. Example approaches of this group use similar descriptors but perform the clustering with a variety of methods, such as spectral clustering [SvKK*11], subspace clustering [HFL12], multi-label optimization [MXLH13], affinity propagation [WWS*13], and deep learning [SQX*16].

Another group of methods in this category takes the global structure of the shapes into consideration when performing the segmentation and then computing the correspondence. The structure can be represented as a template, hierarchy of parts, or a pattern of part arrangements. For example, the approach of Kim et al. [KLM*13] fits a global box-template to the shapes, which provides a segmentation. A correspondence can then be inferred from parts assigned to the same box of the template. The initial template fitting and correspondence provide feedback for the refinement of the fitting, which can be performed iteratively. The approach of van Kaick et al. [vKXZ*13] organizes candidate parts into hierarchies which are consistent across the shapes in the set, providing a hierarchical segmentation and correspondence of the parts. Zheng et al. [ZCOAM14] first compare pairs of candidate parts across shapes in a set to discover recurrent part arrangements. Then, this information is used to define the final shape parts and their correspondence.

In the *correspondence* $\rightarrow$ *segmentation* category, a correspondence is first established among shapes in the set according to a global matching technique, such as shape alignment [GF09], functional maps [HWG14], or structure matching [FvKBCO16]. Next, the shapes can be partitioned into parts consistently according to the adjacency of the geometric primitives on the shapes and their correspondence across shapes [GF09, HWG14], or by transferring a manual segmentation defined for only a few key shapes to all the

		Representation of geometry			Additional method classification criteria		
Works	Functional entity level	Sub-component relation involved	Relation type	Relation representation	Input representation	Approach	Model type
Geometry-only (G)							
Xu et al. [XSF02]	Scene	Yes: object-level	Static	SA	Mesh	Handcrafted	Generative
Merrell et al. [MSL*11]	Scene	Yes: object-level	Static	SA	Mesh	Handcrafted	Generative
Yu et al. [YYT*11]	Scene	Yes: object-level	Static	SA	Mesh	Supervised	Generative
Fisher et al. [FSH11]	Scene	Yes: object-level	Static	SA	Mesh	Handcrafted	Discriminative
Fisher et al. [FRS*12]	Multi-object	Yes: object-level	Static	SA	Mesh	Supervised	Generative
Zhao et al. [ZWK14]	Multi-object	Yes: object-level	Static	BR	Point cloud	Handcrafted	Discriminative
Zhao et al. [ZHG*16]	Multi-object	Yes: object-level	Static	BR	Mesh	Supervised	Generative
Zheng et al. [ZCOM13]	Object	Yes: part-level	Static	SG	Mesh	Handcrafted	Generative
Mitra et al. [MYY*10]	Object	Yes: part-level	Static	SG	Mesh	Handcrafted	Discriminative
Xu et al. [XLX*16]	Object	Yes: part-level	Static	SG	RGB-D image	Handcrafted	Discriminative
Fish et al. [FAvK*14]	Object	Yes: part-level	Static	SA	Mesh	Supervised	Generative
Yumer et al. [YK14]	Object	Yes: part-level	Static	SA	Mesh	Supervised	Generative
Pechuk et al. [PSR08]	Part	Yes: part-level	Static	SA	RGB-D image	Supervised	Discriminative
Gelfand et al. [GG04]	Part	No	–	–	Mesh	Handcrafted	Discriminative

		Representation of atomic interactions			Additional method classification criteria		
Works	Functional entity level	Interacting entity type	Relation type	Relation representation	Input representation	Approach	Model type
Geometry+interaction (GI)							
Hu et al. [HZvK*15]	Object	Static entity	Static	BR	Point cloud	Handcrafted	Discriminative
Hu et al [HvKW*16]	Object	Static entity	Static	BR	Point cloud	Supervised	Discriminative
Pirk et al. [PKH*17]	Object	Dynamic entity	Dynamic	VF	Mesh	Handcrafted	Discriminative
Myers et al. [MTFA15]	Part	Static entity	Static	SA	RGB-D image	Supervised	Discriminative
Kim et al. [KS14]	Part	Static entity	Static	SA	RGB-D image	Supervised	Discriminative
Laga et al. [LMS13]	Part	Static entity	Static	SA+SG	Mesh	Supervised	Discriminative
Hu et al. [HLK*17]	Part	Static entity	Dynamic	SA+BR	Point cloud	Supervised	Discriminative
Geometry+agent (GA)							
Grabner et al. [GGVG11]	Scene	Agent	Static	HA	Mesh	Supervised	Generative
Savva et al. [SCH*14]	Scene	Agent	Static	SA+HA	Mesh	Supervised	Discriminative
Zhu et al. [ZJZ*16]	Scene	Agent	Static	SA	Mesh	Supervised	Generative
Jiang et al. [JKS13]	Multi-object	Agent	Static	SA	RGB-D image	Supervised	Discriminative
Wang et al. [WLY17]	Multi-object	Agent	Static	SA+HA	Mesh	Supervised	Discriminative
Fisher et al. [FSL*15]	Multi-object	Agent	Static	SA+HA	Mesh	Supervised	Generative
Savva et al. [SCH*16]	Multi-object	Agent	Static	SA+HA	Mesh	Supervised	Generative
Ma et al. [MLZ*16]	Multi-object	Agent	Dynamic	SA+HA	Mesh	Unsupervised	Generative
Zheng et al. [ZLDM16]	Object	Agent	Static	SA	Mesh	Handcrafted	Generative
Kim et al. [KCGF14]	Object	Agent	Static	SA	Mesh	Supervised	Generative
Bar-Aviv & Rivlin [BAR06]	Object	Agent	Static	SA+HA	Mesh	Handcrafted	Discriminative
Zhu et al. [ZZCZ15]	Object	Agent	Dynamic	SA+HA	RGB-D image	Supervised	Discriminative
Zhao et al. [ZCK17]	Object	Agent	Dynamic	SA+HA	Mesh	Handcrafted	Discriminative
Lee et al. [LCL06]	Object	Agent	Dynamic	SA	Mesh	Supervised	Generative

Table 1: *All the literature reviewed in this report, classified according to our definition of functionality. We group first by the representation of functionality (indicated in the row titles), then by aspects of the representation of atomic interactions for capturing the functionality, and finally by properties of the method presented by each work.*

remaining shapes in the set, with the use of a propagation mechanism that relies on the correspondences [FvKBCO16].

Relation to functionality analysis. Given that the methods discussed reveal semantics of the shapes based on the similarity of their geometry, they can also reveal functional aspects of the shapes, since that often functionality also relates to the shape of an artifact. Specifically, these methods can provide a correspondence between parts that *likely* possess the same functionality. Thus, these approaches can potentially provide a preliminary analysis of functionality, which could then be leveraged by further analysis.

On the other hand, a limitation of these approaches is that they reveal parts that *likely* possess the same functionality. However, correspondences between parts with different functionality may be established based on their geometric similarity, and it may be difficult to separate them from true functional matches. In addition, although parts with similar functionality are likely matched to each other, the methods do not *name* or *categorize* the type of functional relation that the parts possess, which poses difficulties in generalizing the knowledge for creating a model of functionality. Moreover, these methods reveal corresponding functionality only at the *part level*. They do not provide a model of functionality that can be used to evaluate the functionality of the entire object formed by the parts. Finally, as discussed in detail above, the potential functionality of the parts is revealed based only on an analysis of the static geometry and structure of the shapes, and the methods do not consider additional knowledge such as object-to-object or agent interactions.

3.2. Symmetry detection

The use of analysis methods for detecting symmetries in shapes has also received much attention in recent years [MPWC12]. Methods for revealing reflectional and rotational symmetries have been proposed [MGP06, PMW*08], as well as methods that make use of the detected symmetries for applications such as segmentation [WXL*11] and correspondence [THW*14]. Since symmetries are essentially self-correspondences of shapes, symmetry detection methods can be used to perform a preliminary analysis of the shapes to reveal corresponding parts that likely possess the same functionality, similarly to the methods that compute consistent part correspondences.

3.3. Segmentation and labeling

The objective of methods discussed in this section is also to segment the shapes into semantic parts. The main difference to the methods previously discussed is that the goal also includes the labeling of the parts with pre-assigned category names, such as *hand* or *leg* for the category of human models. For example, the approach of Kalogerakis et al. [KHS10] learns a classifier from samples of segmented, labeled meshes of the same category, based on descriptors that capture the geometry of mesh faces. The classifier is then used to segment and label unknown shapes of the same category. The approach of van Kaick et al. [vKTS*11] also considers correspondence information in addition to geometric features when performing the labeling.

The advantage of these methods in comparison to methods that perform a consistent part labeling is that, in addition to the correspondence among parts that possess similar semantics, we also obtain a *name* for the semantic part. Although these methods provide additional information with the semantic segmentation in the form of part labels, if our goal is to apply the methods to functionality analysis, they possess limitations similar to the previously discussed approaches, such as not being able to separate functionality from other semantics of the shape parts.

4. Geometry-only (G) methods

The methods examined in this section derive knowledge about the functionality of an entity based only on the geometry of the entity. The analysis of the geometry may involve an analysis of structure, involving relations among sub-components of the entity. However, no interactions with other entities are considered, and all of the entities and relations considered are static. For example, some methods involve the analysis of pairwise relations between shape parts, for inferring the functionality of a shape. We discuss the methods in this section by grouping them according to the level of the functional entity considered.

4.1. Scene-level functionality

Methods that analyze the functionality of scenes focus on examining the spatial layout of the objects in the scene, since usually objects are placed in some specific arrangement to enable a certain functionality, e.g., a distinctive arrangement of a TV device, TV stand, and couch enables a person to watch TV comfortably. In the works discussed in this section, only relations of sub-components of a scene are considered, i.e., relations among the objects. Relations to other scenes or interactions with human agents are not incorporated into the analysis. We organize the methods discussed in this part by the type of relation considered.

Spatial arrangement (SA): Most of the previous works that analyze the affordance or functionality of scenes encode the spatial arrangement between the objects. The mapping from specific arrangements to functionality is derived either from handcrafted rules or learned from training data.

The method of Xu et al. [XSF02] uses a combination of rule-based placement constraints, pseudo-physics, and a manually-designed semantic database to guide the automatic placement of objects and generate a scene containing multiple objects. Later, Merrell et al. [MSL*11] incorporated interior design guidelines of furniture layouts for generating indoor scenes, where the layouts can be constrained by objects placed by the user. The part-placement rules used by these methods ensure that the generated scenes can be used for certain types of tasks, and thus possess a specific functionality. Both of these methods use *handcrafted* rules to guide the placement of objects, and the intrinsic functionality models are *generative*, as they can be used to synthesize new scenes.

Similar ideas are explored by Yu et al. [YYT*11], but the hierarchical and spatial relationships that furniture objects need to satisfy are learned in a *supervised* manner. The method extracts the object placement rules from a set of training examples and encodes them in priors associated with ergonomic factors, such as visibility and accessibility. The rules provide a *generative* model of object placement that can be used to synthesize scenes with specific functionality. While the three methods for scene synthesis discussed so far can generate plausible and aesthetically-pleasing furniture arrangements, they do not completely synthesize scenes, since they require a user to specify the set of furniture objects to be arranged. To overcome this problem, Fisher et al. [FRS*12] propose an example-based synthesis method that also selects the objects to be placed in the scene. The selection is performed according to a co-occurrence

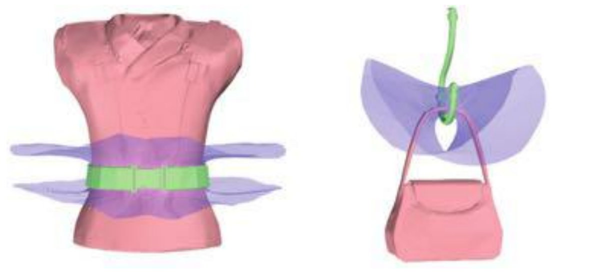

Figure 6: *Examples of two Interaction Bisector Surfaces (IBSs) computed for two different 3D scenes. The IBSs are* geometry-only *(G),* multi-object *functionality representations under our definition. The image is courtesy of Zhao et al. [ZWK14].*

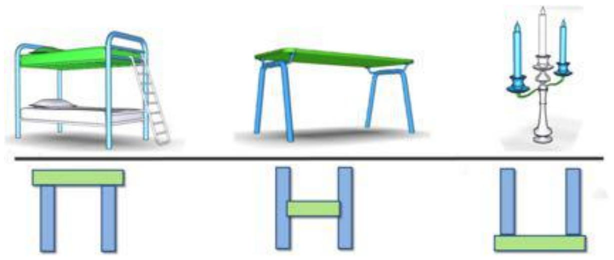

Figure 7: *Examples of three types of Symmetric Functional Arrangements (sFARRs): support, embedding, and placement. Using our functionality classification scheme, sFARRs are* geometry-only *(G)* object-level *representations of functionality. The image is courtesy of Zheng et al. [ZCOM13].*

model, which can be used as a *generative* model to sample and optimize an object layout. The co-occurrence model is learned from example scenes in a *supervised* manner.

Besides the problem of synthesizing object layouts, designing a representation that can be used to compare scenes is also an important research problem. One representative work along this direction is the method of Fisher et al. [FSH11], where scenes are represented as graphs that encode objects and their semantic relationships. The graphs are then compared based on a kernel that takes into account substructures of the graphs. Thus, the approach uses a *handcrafted* graph kernel to provide a *discriminative* approach that allows the comparison of scenes. Similarly, the approach of Xu et al. [XMZ*14] also uses object co-occurrence patterns to discover "focal point" functional regions in a scene for scene retrieval.

Boundary representation (BR): relations that encode the spatial arrangement (SA) between objects can be used to represent static scenes where the placement of objects depends mainly on the adjacency or support among objects. However, these relations cannot encode more complex relationships such as different types of enclosures or entanglements between objects, or interactions with articulated entities such as human bodies or deformable objects. Thus, a more descriptive representation of the boundary between objects can capture interactions between objects more accurately.

A representative approach in this class is the Interaction Bisector Surface (IBS) [ZWK14], which is a subset of the Voronoi diagram that captures the spatial boundary between objects, illustrated in Figure 6. Specifically, the subset of the diagram is represented by a set of geometric descriptors. The IBS can capture more complex interactions between any pair of entities, where the entities can be at any level, e.g., objects, parts, etc. Specifically, Zhao et al. [ZWK14] propose to use the IBS to hierarchically group objects in a scene, providing a hierarchical representation for scene comparison and content-based retrieval. The work defines a measure of closeness for comparison between communities, which can contain a single or set of objects. Thus, the method is *discriminative*, using a *handcrafted* geometric construction to represent the interactions.

To be able to synthesize scenes with complex relations, Zhao et al. [ZHG*16] introduce relationship templates to capture complex relationships between objects in a scene. The templates are derived from example scenes, and combine the interaction bisector surface (IBS) with the space coverage feature (SCF) that encodes the open space surrounding an object. Intuitively, the union of all IBSs is used to subdivide the space into cells, with each cell corresponding to one object in the example scene. The set of cells together with their descriptive features form the relationship template of a given scene. For synthesizing a new scene, objects in different cells are replaced by new objects, where the placement of each new object is guided by feature matching. Thus, the method is *supervised* as the templates are extracted from existing scenes, while the templates provide a *generative* model for synthesizing new scenes.

4.2. Object-level functionality

For methods that analyze the functionality of an entire object, the most common approach found in the literature is to examine the geometry of the shape in terms of its structure, given by relations between sub-components. A common representation of the structure of a shape is a graph of shape parts, where nodes in the graph are connected if the parts are adjacent or close enough on the shape, and the edges of the graph are represented by relations between the parts connected by the edge.

A few works consider special types of structures and relations when building such a graph. For example, Zheng et al. [ZCOM13] employ *handcrafted* rules to detect special groupings of parts called symmetric functional arrangements (sFARRs), which are subsets of symmetric parts that are linked to the functionality of a shape, as illustrated in Figure 7. The method provides a *generative* model of these specific functionalities, since sFARRs can be exchanged between shapes to generate novel plausible shapes, which are well-supported and stable. Moreover, Mitra et al. [MYY*10] introduce a method to illustrate the functioning of mechanical assemblies composed of gears, belts, levers, etc. At the core of the method is an analysis to understand the motion of the assembly, based on detecting a chain of motion from the connection between assembly components. The method detects the motion with a set of *handcrafted* rules based on symmetry relations. Recently, Xu et al. [XLX*16] introduce an approach to recover the functioning of mechanical assemblies from multi-view images, also based on an analysis of the geometry of parts linked together.

Other methods build a graph of shape parts with more general relations. The method of Fish et al. [FAvK*14] takes as input a training set of shapes of a certain semantic class, where the shapes are segmented into parts and labeled. The method then analyzes the configurations of the shapes parts, encoded by multiple relations between pairs of parts, such as angles and relative proportions, and provides a model that characterizes typical shapes in a given class. The model can then be used to test whether unknown shapes are valid shapes of the category by evaluating how well they fit the model. Since most shapes of a known category are modeled after objects that exist and function in the physical world, valid configurations of parts learned from the training data often capture functionality constraints, and thus the model enables to evaluate the functionality of a shape from its geometry.

Yumer and Kara [YK14] also introduce a model to characterize the validity of shapes, which is based on abstracting the input shapes as simple proxies such as cylinders and boxes, and then capturing the common configuration of these proxies. The methods of Fish et al. [FAvK*14] and Yumer and Kara [YK14] are both *supervised*, as they learn from segmented, labeled shapes, and are *generative*, as novel shapes can be created by sampling configurations of parts from the distributions of relations learned for the models.

The main limitation of the last two approaches is that not all the properties possessed by typical models of a class are related to functionality. There are other types of properties, such as aesthetic choices in design, which are mixed with functional properties in the models, and which would need to be filtered out to provide a pure model of object functionality. Structures such as the sFARRs [ZCOM13] discussed above are more strongly tied to functionality, but the approach is limited to discovering a few manually-defined functionality types.

Moreover, there have been works in computer vision that follow a classification approach to recognize the functionality of objects appearing in images, including depth images. These works start by classifying the functionality of parts, and then combine the information about the entire structure of the object to infer the functionality of the object. In earlier work, Rivlin et al. [RDR95] introduce a method that recognizes and labels functional parts of objects according to a hierarchy of parts, where relations between parts are also considered. The functionality of the object is then estimated based on how well the hierarchy is matched by the parts. The part hierarchies are manually defined and the method uses *handcrafted* rules for matching the hierarchies to the data being analyzed. Pechuk et al. [PSR08] provide a supervised version of this approach by extracting the hierarchies from example depth images.

4.3. Part-level functionality

As discussed in Section 2.5, slippage analysis [GG04] is an example of a method that considers only the geometry of an object to infer functionality-related knowledge about the object parts, since the approach segments the surface of the object into contiguous regions, where each region is a kinematic surface composed of points that undergo the same type of motion. The regions describe the possible motions of the shape, and thus provide some indication of the functionality of the shape. Thus, the method operates at the *part-level* and is *discriminative*, as it can be used to distinguish different types of surfaces according to their kinematics. In addition, it is based on a *handcrafted* rule for grouping points into segments.

Note that all of the functionality models discussed in this section are derived solely from the structure of the objects or scenes, and thus certain aspects of functionality, such as the need for regions that support interactions with entities external to the object, e.g., the empty space above a chair seat needed for sitting, are not captured by the analysis approaches and the generated models.

5. Geometry + interaction (GI) methods

In this section, we discuss works where both the geometry of an entity and its interaction with other entities are used to infer its functionality. The key observation used by this group of works is that the functionality of an entity is well reflected by the way that the entity is used when performing its functionality. In these works, the *functional entity* always interacts with other entities, which we call *interacting entities*. The type of interacting entities can be quite general, and works involving the special case of interacting entities which are human agents will be discussed in Section 6. We discuss the methods in this section by first grouping them based on the approach type, and further clustering the works based on interaction type and the level of the functional entity considered.

5.1. Handcrafted functionality descriptors

To understand the functionality of an entity from its usage, the typical approach taken by the methods discussed in this section is to extract and encode the functionality-related information from the context of the entity, and then use this information to derive a functionality descriptor of the functional entity. The input of these methods typically consists of a specified central entity, i.e., the functional entity, and one or more interacting entities that interact with the functional entity in a static or dynamic manner. The output of the methods is commonly a descriptor that provides a structural organization of multiple interactions, or which encodes the dynamic changes of a sequence of interactions.

Static interaction. When encoding multiple interactions that appear in the context of a functional entity into a descriptor of functionality, one of the key challenges is to ensure that the descriptor is insensitive to the variation in the number and categories of interacting objects. For example, a scene composed of a table with several dishes on its top, and another scene composed of a table with one vase on its top, both communicate the supporting functionality of the tabletop. Thus, the key problem to be addressed is to recognize different types of interactions and organize them in a meaningful manner that enables to derive generalized knowledge about the functionality of the entity.

To describe the atomic interactions between the functional entity and each interacting entity, several informative descriptors were introduced, including IBS [ZWK14], IR [HZvK*15], and RAID [GMW16]. As discussed in Section 2.2, both IBS and IR focus on representing the boundary between the geometry of the two entities. IBS is the surface that bisects the free space between the two entities, and IR is the surface region on the entity that is close to and used to determine the IBS. To contrast, RAID encodes complex

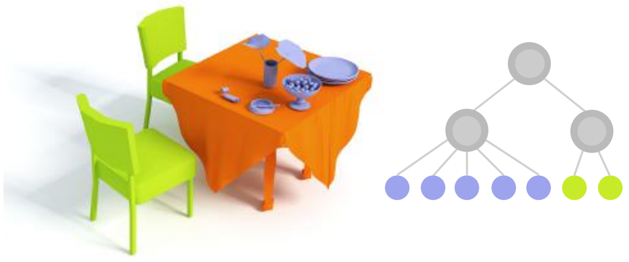

Figure 8: *The ICON descriptor of the central table object shown in orange in the scene on the left. In our definition, the ICON descriptor is an* object-level*,* geometry + interaction *(GI) functionality representation. The image is courtesy of Hu et al. [HZvK*15].*

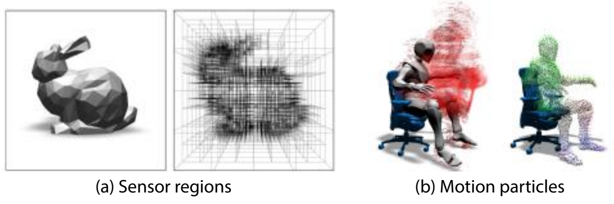

Figure 9: *Components of the interaction landscapes representation of Pirk et al. [PKH*17]. Examples of sensor regions in the domain around the observed shape, shown in (a), and motion particles on the surface of the moving model with direction and speed of movement recorded, shown in (b). In our definition, this is an* object-level*,* geometry + interaction *(GI) functionality representation, which focuses on capturing a single* dynamic *interaction. The image is courtesy of Pirk et al. [PKH*17].*

spatial relationships between two spatial regions as the spatial distribution of simpler point-to-region relationships. Although RAID was first introduced to describe relationships between 2D image regions, the approach can be easily extended to 3D space to encode the spatial relationship between two 3D entities [HLK*17].

To organize multiple atomic interactions, Hu et al. [HZvK*15] group the interactions, which are described both with IBS and IR, into a hierarchical structure based on their feature similarity. The result is a functionality descriptor called ICON, which is essentially a tree where the leaves represent atomic interactions with the central object, and nodes at higher levels of the hierarchy represent general interaction types (e.g., support), since they characterize all the interactions belonging to their subtrees, as illustrated in Figure 8. When comparing the functionality of two central objects appearing in two different scene contexts, their corresponding ICON descriptors are computed and matched by finding a common subtree isomorphism. This provides an meaningful correspondence between the interactions in different scenes, which can be used to derive a measure of functional similarity for the two objects.

Dynamic interaction. The functionality of objects that are used with static interactions, such as tables, can be inferred satisfactorily from static scenes. However, for objects that typically involve dynamic interactions, example sequences of dynamic interactions are needed to fully characterize the functionality of the object. For example, a static object like a cup is typically used with dynamic interactions such as pouring liquid into the cup, or drinking liquid from the cup. While the works mentioned above focus on organizing multiple static interactions of an object, the works that capture dynamic interaction typically focus on describing a single atomic interaction. However, the atomic interaction is dynamic. Since the functional entity remains the same during the interaction process, the main challenge for these approaches is to encode the dynamic change of the atomic interaction. Accordingly, a special setup is needed for capturing the dynamics of how objects move and behave when in functional use, and appropriate descriptors need to be designed to encode and abstract the dynamic characteristics of interactions.

The Interaction Landscapes descriptor [PKH*17] is the first attempt along this direction of describing atomic interactions. To simplify the acquisition and modeling of the interactions, only one interacting entity is considered. During the interaction, the functional entity is kept static and considered as an action receiver, while the interacting entity is considered as a motion driver. To capture the dynamic change of the interaction, the interacting entity is represented as a set of motion particles, and the space around the functional entity is subdivided into sensor regions to track motion particles passing by, as shown in Figure 9. For each sensor region, a vector field is built to encode the dynamic changes of position and velocity of passing-by particles. Thus, each dynamic interaction is represented as a set of vector fields. Attribute histograms are computed for each vector field, and the set of average histograms across all the vector fields are taken as the final dynamic interaction descriptor, called Interaction Landscape. To facilitate the comparison and clustering of interactions, the distance between two Interaction Landscapes is defined as the relative distance between the two sets of global histograms.

5.2. Supervised functionality recognition

The handcrafted functionality descriptors discussed in the previous subsection explicitly describe the functionality of a single object, where the object is given in a static scene context or involved in a dynamic change of interactions. Those descriptors can be used to compare the functional similarity between two functional entities, but cannot be directly used to recognize the actual functionality of the object, especially when the objects are given in isolation. To form a better understanding of different types of functionalities, data-driven methods were introduced to learn the intrinsic properties needed by entities to perform certain functionalities. The following discussion of supervised methods for functionality analysis is divided based on the functional entity level, as different types of methods are used depending on the level of the entity. Since objects are not always guaranteed to appear in a scene demonstrating their functionality, it is important to be able to recognize the functionality of objects in isolation. On the other hand, shape parts are typically given in conjunction with their containing object, and thus a context for functionality inference is usually available.

Object-level functionality. To determine whether an individual 3D object possesses a specific functionality, a sensible approach is to

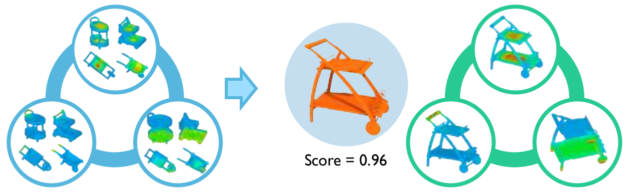

Figure 10: *Example of the "proto-patch"–based category functionality model of Hu et al. [HvKW*16]. Based on the model learned for the handcart category (left), the functionality score of the shape given in isolation (middle) is relatively high, i.e., 0.96, with the corresponding predicted functional patches shown on the right. This is an* object-level*,* geometry + interaction *(GI) functionality representation according to our definition. The image is courtesy of Hu et al. [HvKW*16].*

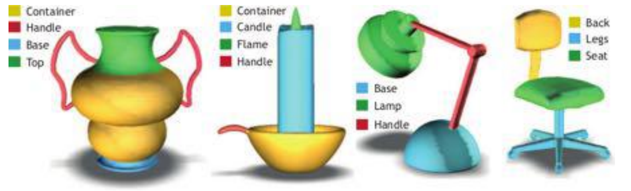

Figure 11: *Segmentation and semantic labeling of 3D objects based on geometric features and part context by Laga et al. [LMS13]. This method leverages a* part-level*,* geometry + interaction *(GI) functionality representation. The image is courtesy of Laga et al. [LMS13].*

first learn what functionality-related properties are commonly possessed by 3D objects that have this functionality, and then verify whether all the properties are satisfied by the object. Typically the properties are learned in a supervised manner from a set of objects belonging to the same functional category.

For GI methods, these functional properties are derived from the interactions of the object. Thus, each training object is not provided in isolation, as is the case for the test object, but in a scene context that reflects the functionality of the object, as for example in the setting for functionality descriptor computation of Hu et al. [HZvK*15]. By co-analyzing the interactions involving objects from a specific category, the method of Hu et al. [HvKW*16] localizes the analyzed properties to object locations, specifically, surface patches, that support specific types of interactions. Then, these patch-level properties are integrated to form a *category functionality model*. More specifically, the learned category functionality model is composed of *proto-patches*, along with their pairwise relations, which together summarize the functional properties of all the patches that appear in the input object category. The basic use of the functionality model is to predict whether an unknown shape supports the functionality of the category. However, unknown shapes often appear in isolation, not interacting with other objects in the context of a scene. Thus, an optimization is solved to simultaneously find the patches on the testing shape that correspond to the proto-patches in the model, and to compute a score of how well the shape and its patches support the model functionality. Each testing shape is then assigned a functionality score to indicate how well it supports the category functionality, as shown in Figure 10.

Part-level functionality. For parts given in an object context, analysis based on intra-object part relations can provide a more complete functional understanding than relying on geometry only. The most common approach for static functionality analysis at the part level is to segment a shape into parts and assign a corresponding semantic label to the parts [LMS13]. For dynamic functionality, part mobility, i.e., how parts move relative to each other when an object is functioning, is the focus of recent work [HLK*17].

The context of a part within a 3D shape provides important cues for learning the semantics of shapes. Thus, the method of Laga et al. [LMS13] follows a supervised segmentation and labeling approach that, besides geometric features, also incorporates the context of parts as additional information to label an object, as illustrated in Figure 11. Given a 3D shape pre-segmented into several parts, their method captures contextual part information with a graph where nodes correspond to parts and edges correspond to structural relationships between parts, such as intra-part symmetry and adjacency. The context of a node is then defined as a substructure of the graph, which is characterized by graph walks of specific length that start at this node. Then, a context-aware similarity measure between two parts can be defined using graph kernels [FSH11], and used to derive a part correspondence between shapes. In particular, this method is able to assign functionality-related labels to parts, recognizing parts as *graspable* or *non-graspable*, independently of the semantic category of the shapes. The training is performed by providing examples of segments that are considered graspable parts, along with counter-examples.

The method of Kim et al. [KS14] labels point clouds to recognize parts that can be useful in the context of robotics, where the labels denote affordances such as "graspable", "liftable", and "pushable". The labeling is based on geometric features of the segments, which include relations between adjacent pairs of segments. Similarly, Myers et al. [MTFA15] focus on understanding everyday tools and propose a method to learn part affordances from local shape and geometric primitives, labeling parts with affordances such as "cut", "scoop", "contain" , "pound", "support" , "grasp", and "wrap-grasp". A part could be assigned with multiple affordances since a tool can be used in different ways.

Dynamic functionalities of articulated 3D objects can be characterized by the motion of one or more of their parts, which we denote by *part mobility*. The presence of part mobilities is ubiquitous in our daily lives, e.g., the opening/closing of the drawers in a chest, or the rotation of the cap of a bottle or the seat of a swivel chair. There have been some previous works focusing on discovering object mobility from an input scene with repeated instances of objects and parts [SHL*14], or discovering part mobility of an object from a sequence of RGBD scans of the dynamic motion of articulated models [LWL*16]. In these works, the input data provides different states of the motion and the key technical problem is motion fitting to the sequences. However, in our daily lives, we, humans, apply motion inference to objects all the time. In general, we can predict the functionality of unseen objects by prior experience or knowledge on similar objects. In the same spirit,

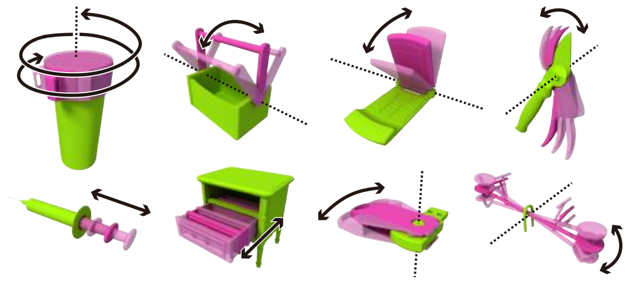

Figure 12: *Examples of motion prediction results using the part mobility model of Hu et al. [HLK*17]. This approach uses a* geometry + interaction *(GI),* part-level *functionality representation to analyze dynamic interactions involving part motions. The image is courtesy of Hu et al. [HLK*17].*

Hu et al. [HLK*17] show that it is possible to infer part mobilities of a static 3D object by learning from motion sequences of parts from different classes of objects. By learning a motion-dependent *snapshot-to-unit distance measure*, the learned part mobility model can predict mobilities for parts of a 3D object given in the form of a single static snapshot, where the snapshot reflects the spatial configuration of the object parts in 3D space. The prediction can also be used to transfer the mobility from relevant training data to static shapes, as shown in Figure 12.

6. Geometry + agent (GA) methods

In this section we discuss methods that involve modeling of an agent interacting with objects in order to characterize the functionality of the objects. Agents can be posed humans, articulated robotic hands, or other inherently animate entities. Agents are distinguished from other objects in that they are performing an action, usually highly correlated with the functionality of the functional entity. Thus, an analysis of functionality based on agents can reveal otherwise hard to infer latent functionalities of the functional entities.

Analysis of functionality through what we have called the *geometry + agent* approach has been becoming increasingly popular. Since an embodied understanding of the natural world is a common experience shared by people, a human agent analysis of functionality is a natural direction for research. Interpretation of common human actions in the real world through the lens of "atomic interactions" is by extension a natural view to take.

Here, we summarize prior work that involves modeling of both geometry and agent (GA) using three major axes of classification: a) the focus of the functionality representation, b) the level of the functional entity, and c) the temporality of the interaction relation.

6.1. Functionality representation

First, we divide prior work into two main facets reflecting a difference in focus for the functionality representation: *geometry-centric* vs. *agent-centric* representations.

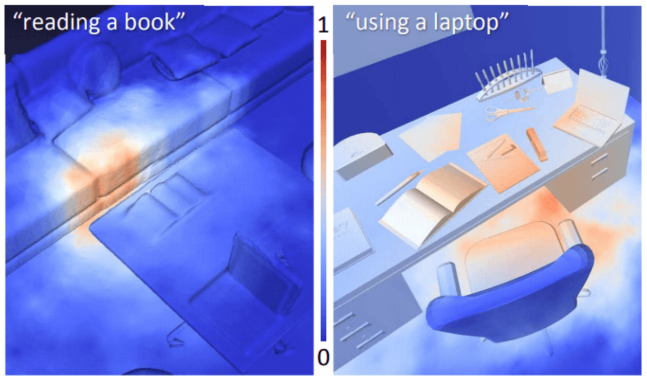

Figure 13: *"Action map" predictions by the SceneGrok system of Savva et al. [SCH*14]. This* geometry + agent *(GA) functionality representation encodes interactions in a geometry-centric view, assigning an action probability to each point on the 3D surface of the input scene. The blue-to-red heatmap visualizes the probability. The images are courtesy of Savva et al. [SCH*14].*

Geometry-centric representations. This group of works primarily uses the geometry of the non-agent entities (typically the geometry of the functional entity) to organize the representation of functionality. Examples in this group include the early work of Bar-Aviv and Rivlin [BAR06] on "Agent Based Simulated Vision" which classifies objects through collision and contact detection with human poses and other agents, and the work of Grabner et al. [GGVG11] on sittability detection through the use of a sitting agent template. Later, the work of Jiang et al. [JKS13] hallucinated human poses in order to improve object detection and predict affordances in RGB-D frames, and the work of Savva et al. on SceneGrok [SCH*14] predicted "action maps" over 3D scenes (see Figure 13). All three of these works leverage some form of human pose prior to evaluating the likelihood that the local geometry can support the pose. More recently, Fisher et al. [FSL*15] demonstrated that annotations of object geometry with human interaction regions can be used for human activity-centric scene synthesis, improving the quality of synthesized 3D scenes. Zhu et al. [ZZCZ15] simulate the dynamics of tool use by humans during specific tasks (e.g. hammering a nail) to select the most appropriate tool and optimal trajectory given the target object of the task. In a similar direction, Zhu et al. [ZJZ*16] simulate the forces between a human pose and geometry in a 3D scene to infer locations in the scene that can comfortably support the human pose. Wang et al. [WLY17] take as input a set of objects and jointly select one of the objects as a container for the rest, and a human pose for carrying the container and containees.

All these works center their representation on the geometry with which the agent is interacting (typically the surfaces of a collection of objects in a 3D scene). In other words, these works cast functionality into a function over the geometry of the functional entities, one example being the "action map" representation of Savva et al. [SCH*14]. These geometry-centric representations are natural for addressing tasks that predict the functionality of newly observed geometry without corresponding agent observations. In practice, it is hard to capture both object geometry and agent poses. This means that the agent is rarely directly observed, making this group of geometry-centric methods appropriate.

Agent-centric representations. On the other hand, one can focus the functionality representation on the agent entity instead. In doing so, functionality is organized mainly around the structure of the agent (as opposed to over the geometry of the functional entity). Note that the representation may still involve geometric properties of the agent, but typically the agent is represented abstractly, for example using the connectivity structure of the human pose. Prominent examples of agent-centric representations are the Shape2Pose priors of Kim et al. [KCGF14] and the PiGraphs interaction priors [SCH*16]. Earlier work in the animation community by Lee et al. [LCL06] has proposed the "motion patches" representation which organizes geometric patches by analysis of motion captured pose data that interacts with the patch. Zheng et al. [ZLDM16] demonstrate an efficient 3D object editing interface that connects deformations of the object geometry to the pose of a person interacting with the object through a set of ergonomics rules. Zhao et al. [ZCK17] have proposed indexing of an IBS-based representation of objects using a human pose in order to enable retrieval of human-object interaction sequences from attributes of the pose.

All the works in this group organize their functionality in an agent-centric fashion. This makes it natural to use such methods when it is necessary to infer what geometry a particular agent configuration (i.e. human pose) can interact with, or to predict plausible agent configurations given the geometry as input. The latter task is related to work in constraint-based animation, and to work in virtual agent behavior synthesis.

6.2. Functional entity level

The second axis over which we classify prior work is the level at which the functional entity exists. The existing works can be split mainly into three target functional entity levels: individual *objects*, *multi-object* regions, and *scenes*.

Object level. This level simplifies the analysis of functionality to distinct objects. This is typically done by looking at the geometry of de-contextualized objects, and analyzing agent-object interactions mainly within a specific category of such de-contextualized objects (e.g., chairs). This simplification partitions the space of possible agent-object interactions and makes functionality analysis more tractable. A prominent example in this group of work is the Shape2Pose system of Kim et al. [KCGF14] which predicts interaction poses given the geometry of a specific instance for a known category of objects. The shape deformation interface of Zheng et al. [ZLDM16] is designed to operate at the level of distinct objects such as benches or chairs. Similarly, the retrieval approach of Zhao et al. [ZCK17] indexes single objects using the human poses that would be interacting with them. The animation synthesis system of Lee et al. [LCL06] also organizes motion captured human pose sequences by the geometric properties of patches at the object-level.

Multi-object level. This level is an extension of the object level, where the agent is now jointly interacting with multiple distinct objects within a functionally-connected region (e.g., a working desk area). This level of functionality analysis is well suited to the work focusing on 3D scene synthesis, as it allows the structure of 3D scenes to be factored into regions with sets of functionally-connected objects. Two example prior works in this space are the activity-centric scene synthesis system of Fisher et al. [FSL*15], and the action-driven 3D scene evolution system of Ma et al. [MLZ*16]. Both of these methods tackle the arrangement of regions containing multiple objects by explicitly representing an agent performing common human actions with the objects. The PiGraphs representation of Savva et al. [SCH*16] is also formulated at the multi-object level, since it connects sets of objects to the human pose through static support and human gaze relations (see Figure 14). Finally, the earlier work of Jiang et al. [JKS13] also predicts object affordances jointly for multi-object sets that are likely to be interacting with a hallucinated human pose. Since the human pose frequently interacts with multiple related objects, the multi-object level appears to be a common choice for analyzing functionality using human agent–based methods.

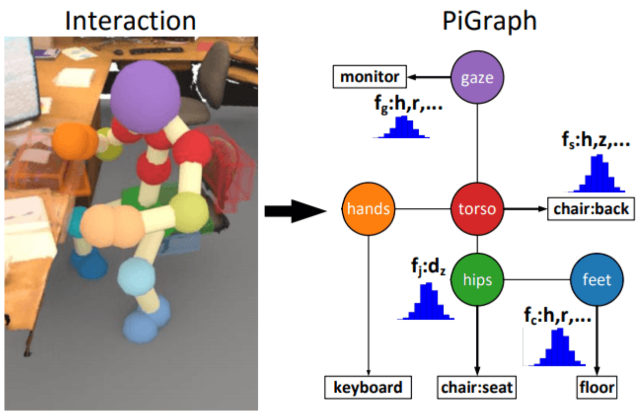

Figure 14: *The PiGraphs [SCH*16] representation is an agent-centric* geometry + agent *(GA) functionality representation. Interactions of the human pose with objects in the scene are encoded based on the connectivity of the human pose. The images are courtesy of Savva et al. [SCH*16].*

Scene level. At the scene level, the functional entity is now a complete scene containing potentially multiple functional regions each consisting of multiple objects. The focus of work at this level shifts mostly to identifying distinct functional regions in the broader context of the entire scene. One line of work that exemplifies this focus is prediction of abstracted affordances such as the "sittability" prediction of Grabner et al. [GGVG11], and the human pose support comfort prediction of Zhu et al. [ZJZ*16]. Another work at this level is the SceneGrok system of Savva et al. [SCH*14] which predicts the probability of more specific actions (e.g. "watching TV") taking place at positions over a 3D scene. This group of work performing functionality analysis at the scene level has focused on such higher-level notions of functionality which are generally connected to human action affordances.

6.3. Type of relation

The third classification axis that we use characterizes whether the relation between the functional entity and the agent exhibits variation along the temporal domain. That is, whether the relation is *static* (an observation at a single point in time) or *dynamic* (a set of observations over time).

Static. Methods in this category are based on analysis of a set of interactions that are observed to hold at a single point in time. A basic assumption in this line of work is that the functionality can be fully characterized by a single observation of the interactions between an agent and the functional entity, or at the very least that a single observation captures a sufficiently large fraction of relevant atomic interactions to characterize the functionality. Most existing methods fall in this category, as it is practically much easier to work with single snapshots rather than sequences of observations. For example all scene level and multi-object level works work with static functional entities and static relations between the agent and the functional entities. The only exception is the work of Ma et al. [MLZ*16] that evolves a 3D scene layout through a series of static configurations exhibiting transitions between related human actions (we have categorized this work as dynamic, but more concretely it builds a transition model between states which are static).

Dynamic. Methods in this category perform functionality analysis that involves interactions over a temporally coherent sequence of observations. This is a natural choice for agent-driven functionality analysis as most agents are inherently dynamic and their actions evolve over the time domain. The work of Lee et al. [LCL06] on motion synthesis is a dynamic method as it represents the dynamic relations of human pose sequences against the geometry of the objects with which the pose is interacting. More recently, Zhao et al. [ZCK17] perform motion classification and retrieval using the interactions of human pose sequences with target objects. Both of these approaches leverage temporally contiguous, coherent observations of an agent interacting with objects. As it is hard to acquire the input data that is necessary for such joint dynamic analysis of agent-object interaction, there are few dynamic agent-based methods. However, this research direction is becoming increasingly relevant with the rise of interactive VR and AR applications.

6.4. Summary

In this section we discussed a variety of geometry + agent (GA) methods that explore a multi-faceted, rich notion of functionality. Key to these methods is the connection between the action of an agent entity with objects, regions and scenes and the functionality that enables the action to take place. This correlation of functionality and agent action can be viewed either from the perspective of the geometry (*geometry-centric* representation), or from the perspective of the agent (*agent-centric* representation). The analysis can span a range of functional entity levels including *objects*, *multi-object* regions, and entire 3D *scenes*. The dynamic nature of agent actions further enriches this notion of functionality, and is starting to be the focus of recent work in the geometry + agent category.

There is significant potential for growth in this research area due to emerging applications in VR and AR, where a human-centric embodied understanding of real spaces is critical for enabling relevant information retrieval, and interaction with 3D spaces. This research area is also connected with the rising domain of fabrication. Many artifacts that are becoming 3D printable (e.g., toys, prosthetics, novel furniture designs) are fabricated for practical use by people. Thus, a human-centric functional understanding of the artifacts to be generated is a critical part of the design process.

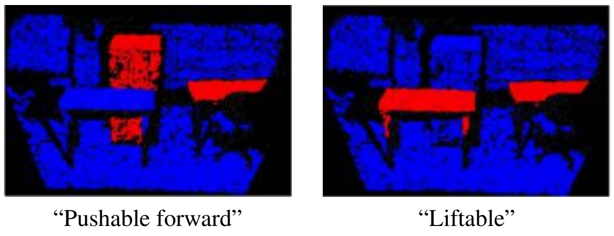

Figure 15: *Examples of depth images labeled with two functional labels. The image is courtesy of Kim and Sukhatme [KS14].*

7. Applications

In this section, we discuss how prior work involving functional analysis of shapes has been applied to a variety of applications. We organize the work firstly by application domain, dividing our discussion into subsections covering each domain.

7.1. Classification, segmentation, and labeling

Classification, segmentation, and labeling are all closely-related tasks. A classification can be performed at different levels of an entity, for example, to classify an entire object into one class out of a set of pre-defined object classes, or to classify parts of the object into classes, which would imply a segmentation of the shape with a labeling of the segments. Thus, we discuss these applications jointly in this section, based on the level of the entity considered.

Part classification. All of the methods discussed in this section classify the shape's primitives, e.g., triangles of a mesh, into different groups, which effectively provides a segmentation of the object into parts and labeling of the parts. The main differences among the works are what types of labels are assigned to the primitives.

The slippage analysis approach of Gelfand and Guibas [GG04] segments a triangle mesh into kinematic surfaces, which indicate regions of the shape that undergo a similar type of motion. Each resulting segment is classified into one of a few types of kinematic surfaces, such as planes, spheres, and cylinders.

Pechuk et al. [PSR08] introduce a supervised method to recognize the functional regions of a shape according to a model of functionality of the shape class. The functional model encodes the geometry of parts and pairwise relationships that should appear in a shape of the class, and thus the detected parts correspond to components of this model. Moreover, the supervised method of Kim and Sukhatme [KS14] classifies regions of shapes with functional labels such as "graspable", "liftable", and "pushable" (see Figure 15). Similarly, Laga et al. [LMS13] introduce a supervised method that

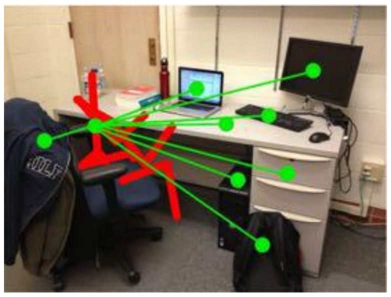

Figure 16: *A human pose predicted for an input scene, along with the scene's context used for object labeling. The image is courtesy of Jiang et al. [JKS13].*

considers geometric features and the context of parts in the shape to label shape parts with labels. The method learns the classification from a set of labeled training examples. In terms of functional labels, the work presents results mainly for labeling parts as "graspable" or "non-graspable".

Moreover, the method of Hu et al. [HvKW*16] defines weight fields over a point-sampled surface based on a learned classifier, where the weight of a point defines the probability of the point belonging to a specific type of *proto-patch*. Proto-patches correspond to functional regions of shapes, such as the "sittable" region of a chair, or "graspable" region of a stroller's handle. In principle, the weight fields define a probabilistic labeling of points rather than a segmentation of the shapes into well-separated parts. However, the weight fields can be exploited to extract functional segments in combination with a segmentation method.

For considering dynamic parts, Hu et al. [HLK*17] classify query pairs of already-segmented static parts into a set of motion classes, defined by the grouping of the training data. The method finds the nearest neighbor example in the training set, which allows to transfer the motion type and motion parameters from the example to the query pair, effectively transferring dynamic motion information to static parts.

Object classification. The method of Grabner et al. [GGVG11] classifies a 3D object as being a chair or not a chair, based on the pose assumed by a humanoid agent while attempting to sit on the object. Thus, the method is demonstrated on one type of functional category, and classifies objects as a whole into the two categories. Moreover, the method of Hu et al. [HvKW*16], besides providing weights fields that define the probability of points belonging to a proto-patch, as discussed in the previous paragraph, can also be used to classify entire objects into functional classes, by integrating the weight fields in a global energy function. Although the classes in the training data have common semantic names such as "chairs", "handcarts", and "tables", the grouping of the shapes follows their functional use rather than geometric similarity.

The methods discussed so far label individual objects mainly based on the geometry of the shapes. On the other hand, the following methods provide a labeling of objects based on their context in a larger scene. Jiang et al. [JKS13] introduce an approach to predict human poses that best fit static 3D scenes. Based on this prediction, the approach then labels objects in the scene according to the context formed by the objects and the fitted human poses (see Figure 16). The method assigns semantic labels to the objects, such as "table", "monitor", etc. Zhu et al. [ZJZ*16] propose an approach with a similar initial goal, where human poses are fitted to static 3D scenes according to a physics-based simulation. However, rather than using the approach for labeling objects in the scene, the approach extracts the *human utility* of objects in the scene, which provides an indication of the level of comfort of a human when interacting with the object, e.g, the level of comfort when sitting on a chair. This is made possible with the physical simulation that derives the forces and pressures that are applied to various body parts of the human agent when interacting with objects in the scene.

Scene classification. The SceneGrok approach [SCH*14] learns a classifier from human actions tracked on scenes, including actions such as sitting, using a computer, and reading a book. Then, the classifier can be used to segment and label query scenes with action labels, providing an *action map* of the scene. The action maps define the probability of regions in the scene being used for the pre-defined human actions.

7.2. Retrieval

The goal of geometric retrieval is to rank geometric datasets in a collection according to their similarity to a query, to facilitate the exploration of the collection. The most common application examples are shape and scene retrieval, where the goal is to rank shapes (or scenes) in a database according to their similarity to a query shape (or scene). This necessitates the definition of a proper similarity measure to compare two entities. Moreover, in the context of functionality-aware processing, the similarity measure should take functional aspects of the shapes or scenes in consideration. We group the discussion of the works in this section by the level of the entity retrieved.

Part retrieval. RAID [GMW16] is a descriptor that captures complex relationships between image regions, such as different forms of containment. Given images segmented into regions, the descriptor can be used to retrieve pairs of regions that have relationships similar to those between a query pair of regions (see Figure 17). RAID is adapted to the 3D setting and used with other functionality-related descriptors in the approach of Hu et al. [HLK*17]. As discussed in the previous section, this approach predicts the possible motion of a static pair of parts with a model of part mobility learned from a training set of part motions. The method can also be used in a retrieval context, where a type of motion is given as a query, and the method retrieves shape parts that likely support the motion type.

Object retrieval. The regions corresponding to proto-patches detected by the approach of Hu et al. [HvKW*16] can be used to define a functional similarity measure between two objects, which is relevant in the context of object retrieval focusing on object functionality. Moreover, the Interaction Landscape of Pirk et al. [PKH*17] describes dynamic interactions between objects. The descriptor, represented as histograms of properties of vector fields

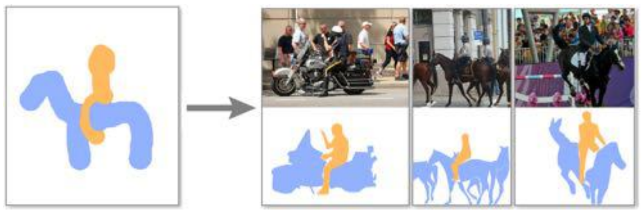

Figure 17: *Image retrieval performed with the RAID descriptor. Left: sketch with the query regions. Right: retrieved regions with similar relationships, and corresponding images. The image is courtesy of Guerrero et al. [GMW16].*

that define the motions of the interactions, can be used for retrieval of similar interactions at the object-level, but independently of the geometry of the objects involved. This is accomplished by directly comparing the similarity of the vector fields according to the histograms of their properties.

Scene retrieval. Fisher et al. [FSH11] introduce an approach to estimate the similarity between two scenes, by comparing objects in the scenes and their contexts with a graph kernel capturing geometric relationships between the objects. The similarity measure can be directly used to perform retrieval of scenes from a collection, and can be further leveraged to search for similar objects according to their contexts in scenes. From a more functional perspective, as discussed in the context of classification, the SceneGrok approach of Savva et al. [SCH*14] predicts action maps for scenes, defining regions that can likely be used for specific actions. A descriptor capturing functional aspects of the scene can then be obtained by integrating the action map corresponding to each individual action. Scene retrieval can be performed with this descriptor, to locate scenes that enable similar types of human actions.

Zhao et al. [ZWK14] build a similarity measure for local communities of scenes based on their Intersection Bisector Surface (IBS), where a local community denotes a group of one or more objects and their interactions with neighboring objects. This provides an approach to perform retrieval of scenes based on the similarity of subsets of objects. Moreover, the ICON descriptor of Hu et al. [HZvK*15] provides an estimate of the functional similarity of two objects given in the context of a scene, based on a hierarchical description of object-to-object interactions. Thus, the descriptor can be directly used for performing object-in-scene retrieval from a collection of scenes. Finally, the focal point approach of Xu et al. [XMZ*14] also enables retrieval of objects in the context of scenes, by emphasizing the focal object and its relations to neighboring objects in the scene.

7.3. Synthesis

Synthesis of 3D content is a prominent application domain for many works involving functionality analysis. Generative models leveraging functionality have been applied to the domain of indoor 3D scene synthesis, and to the domain of human interaction synthesis (i.e. generating human poses and corresponding 3D scenes with which the human pose is interacting).

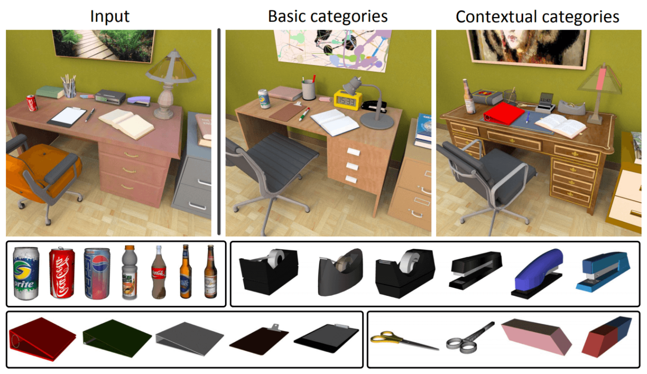

Figure 18: *Fisher et al. [FRS*12] demonstrate contextual object categories inferred from commonalities in the context within which objects are observed (bottom, in black outlines). These automatically extracted categories are used to improve the perceived quality of the synthesized 3D scenes (top). The images are courtesy of Fisher et al. [FRS*12].*

3D scene synthesis. In this application, functional constraints or priors are used to inform generation of 3D scenes, including object selection and layout optimization. Xu et al. [XSF02] use a constraint-based formulation that encodes functional relations between objects in the form of "binding areas" and "offer areas" (representing object co-occurrence priors). Their system speeds up interactive 3D scene composition by offering automatic placement suggestions. On the other hand, Yu et al. [YYT*11] assume that the full set of objects in a scene is given as input, casting the problem as one of automatic layout optimization. The objective function of their optimization encodes several functional priors for furniture arrangement, such as visibility and accessibility. Fisher et al. [FRS*12]'s example-based 3D scene synthesis method learns a probabilistic model of both object occurrence and object layout. Their model is based on a set of object co-occurrence and spatial arrangement relation priors, and a set of automatically extracted "contextual categories", which categorize objects through their observation context (see Figure 18).

Human interaction synthesis. A closely related line of work focuses on synthesis of 3D scenes guided by human interaction priors. For example, Fisher et al. [FSL*15]'s activity-based 3D scene synthesis algorithm encodes human-object interactions to synthesize more plausible 3D scenes, given the types of human actions that they should support. In a similar vein, Ma et al. [MLZ*16]'s action-driven 3D scene evolution system alters a 3D scene through object placement and layout optimization such that specific actions can be performed. Though both these works leverage human-centric priors, they do not generate human poses. Kim et al. [KCGF14]'s Shape2Pose system generates interaction poses given the geometry of an object instance, whereas Savva et al. [SCH*16]'s PiGraphs system jointly generates a human pose and 3D scene composition given a text-based specification of a human interaction with a scene.

7.4. Modeling and editing

Another common application domain for functionality analysis methods is that of modeling and editing 3D geometric representations. In this application domain, an interactive modeling interface is typically augmented by the functionality-based method to allow for more efficient or more advanced editing operations.

Object modeling. The symmetry hierarchy of Wang et al. [WXL*11] encodes relations between parts of an object and allows for efficient symmetry-aware editing of objects (e.g., simultaneously adjusting the shape and position of all candles on a chandelier). Similarly, Zheng et al. [ZCOM13]'s symmetric functional arrangements are used to automatically generate object variations by matching, replacing and adjusting triplets of parts. Both of these methods primarily rely on symmetry relations between parts to enable efficient editing of object geometry. Umetani et al. [UIM12] further propose an interactive modeling framework for efficient exploration of not only geometrically- but also physically-valid furniture designs. Fish et al. [FAvK*14]'s meta-representation also encodes relations between object parts, enabling functionality-aware validation and manipulation of part position and orientation. Similarly, Yumer and Kara [YK14]'s method learns a set of co-constrained handles for deformation of object parts in a shape collection. By relating global structure to functionality, Lun et al. [LKWS16] automatically transfer the style of one shape to another with different functionalities via a sequence of element-level operations. Finally, Zheng et al. [ZLDM16] encode constraints relating the human pose with the part structure of chairs to allow for ergonomics-inspired reshaping of the object geometry (see Figure 19), while Fu et al. [FCSF17] use human poses to guide part assembly and the synthesis of functional hybrid objects.

Scene editing. There are some works that use a functionality-based model to assist the user during 3D scene editing. Merrell et al. [MSL*11] encode a set of interior design guidelines in the system, reflecting functionality patterns such as clearance for navigation. These guidelines are used in an interactive furniture layout interface to automatically suggest optimized furniture layouts given the current state of the scene. Zhao et al. [ZHG*16]'s template-based approach can also implicitly be used for scene editing since it allows for easily generating variations of a given object layout through object swapping. Sharf et al. [SHL*14] use a mobility-tree representation to extract the potential ways in which objects exhibit motion relative to each other in a scene (e.g., chairs around a table). This representation is then used to allow for easy manipulation of groups of objects and to make scene editing more efficient.

7.5. Visualization and fabrication

A rising application domain for functionality analysis methods is in visualizing the assembly procedure of physical objects, or the dynamic behavior of assembled and operating mechanisms. With a similar understanding of mechanical relations, 3D object designs can be optimized to be more easily fabricatable.

Assembly visualization. Mitra et al. [MYY*10] did early work on generating illustrations of how mechanical assemblies work.

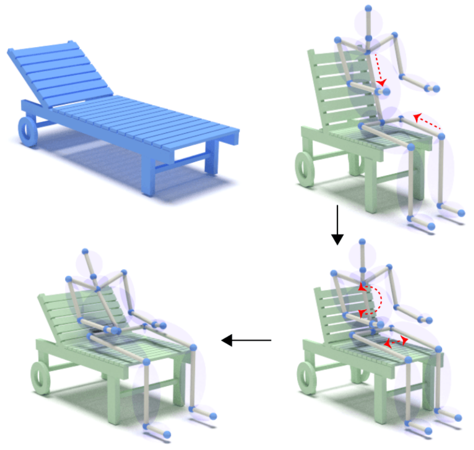

Figure 19: *The ergonomics-based 3D object editing interface of Zheng et al. [ZLDM16] allows for efficient editing and exploration of 3D chair designs that conform to the specified human pose configurations. The image is courtesy of Zheng et al. [ZLDM16].*

Their method discovers the relative motion that different component parts of a mechanical assembly can undergo. More recently, Koo et al. [KLY*14] presented a system for rapid design of "works-like" prototypes of articulated rigid objects such as cupboards. The system relies on a set of part-to-part relations including support and enclosure, and optimizes the size and placement of the parts to ensure that the relations are preserved during articulation of the object's parts. Xu et al. [XLX*16] have recently presented a system that takes as input a set of images of a mechanism, and allows for more efficient extraction of the geometry and motion patterns of the mechanism.

Fabrication. Applying functionality analysis methods for fabrication is a natural extension to applications in object modeling and shape editing. An early example of work in this space is by Lau et al. [LOMI11], whose method converts 3D furniture models into sets of fabricatable parts and connectors, using a part-based analysis involving part–connector–part relations. More recently, Shao et al. [SLR*16] have also presented a system that takes as input multi-step assembly instruction diagrams and reconstructs the part geometry and assembly process. The reconstructed part geometry can then be fabricated, and assembled into the intended object following the generated assembly instructions. This line of work on functionality-aware methods for fabrication is likely to become increasingly important as fabrication technologies become more ubiquitous.

8. Conclusion

In this report, we provide a comprehensive survey of works on functionality representation and their applications in computer graphics. In contrast to previous works involving a semantic understanding of 3D shapes, which focus more on the structural arrangement of parts of an object or objects in a scene, the explicit modeling of functionality can provide a deeper link between those structures and the intended design and purpose of the object or scene. This deeper understanding can facilitate applications such as classification, segmentation, modeling, and synthesis.

By defining functionality as a combination of *geometry and interaction*, we are able to split the characterization of the functionality of an entity into intrinsic/internal properties of the functional entity and extrinsic/external properties that arise from interactions of the entity with other entities in the environment. This definition also provides a natural grouping for studying existing works. Works that consider only the geometry of an entity do not build a direct connection between the intrinsic properties of the entity and the manner in which the functional entity is used, in contrast to works that take interaction into account which makes this connection explicit. Especially when the interacting entities are agents such as simulated humans or articulated robotic hands, directly modeling the agents for simulating interaction and associating the interaction with intrinsic properties of the functional entity can provide a rich understanding of functionality.

Limitations

Our definition of functionality considers geometry and interaction as two key aspects for understanding how an entity functions, or how it is used for a certain purpose. We reviewed the works in the literature according to this definition. However, our definition is by no means a complete or perfect description of functionality. In particular, there are several questions to consider as interesting avenues for extending this definition: Is there a better way of encoding functionality? In other words, are there other fundamental properties of parts, objects, and scenes which should enter into our definition? How far can we go by looking at form or function alone? More concretely, can we directly infer functionality from geometric form, or can we infer geometric form from functional descriptions? There is a broad space of related research problems to be explored.

Current challenges and directions for future work

Functionality analysis is a relatively new topic in computer graphics with a lot of potential for further work. Here, we discuss a list of challenges and open problems for future exploration.

Different levels of functional entities. Functionality can be defined for entities at different levels, as we discussed in the report. There is much similarity among the techniques that analyze functionality at the same level, but not so much similarity when comparing techniques targeted at different levels. Thus, treating multiple levels of functionality, e.g., in a hierarchical representation, remains a challenging problem to solve with the existing state-of-the-art. A multi-level treatment of functionality might need to involve multiple levels of entity-entity or entity-human interactions.

Static vs. dynamic interactions. In most of the works existing in the literature, the analysis is restricted to *static* artifacts and interactions, especially in the case of geometry-only methods, where all the techniques studied in this report consider only static geometry and structural relations. Thus, analyzing *dynamic* geometry and interactions is a research area that would clearly benefit from further work. Moreover, an interesting question towards this direction is whether geometry-only methods need to be restricted to analyzing static interactions. Would an analysis of the motion of a single shape provide information on the shape's functionality? An approach based on this idea would perhaps be relevant for predicting the possible motion of a static object. Nevertheless, for any approach aiming at modeling dynamic interactions, the first step to address is the problem of capturing dynamic 3D data. The difficulties in collecting this type of data may only be justified if dynamic interactions are proven to be superior to static interaction data when modeling a specific type of functionality or enabling certain functionality-related applications.

Interaction/relation representation. Based on the classification of works shown in Table 1, the most common approaches for encoding interaction relations are spatial arrangement (SA) properties, boundary representations (BR), and humanoid actions (HA). An interesting research question is to explore whether there are more informative relations for capturing interactions. Moreover, since different types of relations capture different types of information, an obvious direction for future work is to investigate how the existing models can be combined together, in an efficient and meaningful manner, to improve the understanding of an artifact's functionality, e.g., to develop a method that is able to handle both entity-entity and entity-agent interactions.

Handcrafted vs. data-driven models. In recent years, with the increasing availability of 3D data and advances in learning techniques, using data-driven methods for shape analysis and processing has become quite popular [XKHK17]. However, in most of the works on functionality analysis, although some level of learning is used, the models of functionality themselves are still mostly defined based on prior knowledge and handcrafted features. One likely cause is that a large-scale dataset for functionality analysis does not currently exist. Most of the existing works introduce their own dataset to prove a concept, but the lack of a large amount of data prevents most methods from learning an entire model from scratch, requiring professional guidance for their construction. Establishing a large functionality dataset and making full use of the power of advanced machine learning techniques may be a fruitful direction for future work.

Input data type. Currently, most of the works assume that "clean" data is given as input, especially polygonal meshes with appropriate topology. There are research opportunities for considering other 3D shape representations: volumetric data (uniform voxel grids or octrees), parametric surfaces, constructive solid geometry, etc. However, to apply functionality analysis in acquisition and reconstruction scenarios, ideally the methods would accept raw 3D scans as input. Such data provides new research challenges due to its shortcomings: presence of noise, incomplete / partially observed regions, imperfect registration, etc. In addition, it would be interesting to perform functionality analysis by combining multiple data types. For example, a joint analysis of functionality with both 2D images and 3D datasets could certainly benefit from the different cues provided by these data types.

"Grand challenge" in applications. Applications that use functionality information are likely still in their infancy. By proposing new possible applications that involve an understanding of functionality, we may also discover aspects of functionality that are missing in the analysis of current methods and functionality representations. We explore a few possible applications here.

Functionality-aware shape modeling: Current shape manipulation methods are largely based on the principle of preserving the form, in the hope that preserving the form will lead to a preservation of the functionality of the artifact, which is often not true in many cases. For example, shrinking the width of a chair's seat might break or alter its suitability of being sittable. On the other hand, human-centric approaches are largely designed for particular object categories and have limited use. Thus, the introduction of shape modeling approaches that make use of explicit functionality information appears to be a paramount direction for future work in computer graphics. Moreover, using functionality to guide shape manipulation can also be essential for fabrication. For example, a comprehensive understanding of the functionality of mug cups or chairs could allow a fabrication design tool to suggest novel designs that preserve functionality while allowing the shape to vary in form, appearance, and material properties.

Functionality-guided shape synthesis: One step further from functionality-aware manipulation and editing would be to generate an entire shape from a given class, based only on functional constraints. In recent years, generative models using deep learning have become an active area of research. These approaches could be leveraged for learning functionality-related constraints and for guiding shape generation.

Functionality-driven real-time interaction: In the context of robotics, it is important for a robot to understand the functionality of its surroundings to be able to interact with the environment. Functionality analysis is thus important in this context as well, and will likely also be critical for many emerging applications in AR and VR. For example, when using AR glasses, a recognition of the functionality of drawers and cupboards in a kitchen, indicating that they can be opened/closed and that they can store objects, would allow the AR system to display stored objects, or perform "search and assisted navigation" to objects.

Full understanding of 3D shape. In this report, we focused on the inference of functionality from the geometry of the input. However, other aspects of an artifact could also be tied to its functionality and could be explored during the analysis, e.g., material, weight, or size. Moreover, a full understanding of the functionality of an artifact does not necessarily imply a full understanding of the artifact as a whole, as there are other non-functional aspects, such as style, ornaments, etc., that are part of the shape but not related to its functionality. Thus, methods in shape analysis and design would likely benefit from combining different types of analyses, such as functionality analysis and style analysis, to reach a richer understanding of 3D shapes.

Acknowledgements

We thank all of the reviewers for their valuable suggestions. We also thank the authors of the figures reproduced in this report with permission from the publishers. This work was supported in part by grants from NSFC (61602311, 61528208), 973 Program (2015CB352501), Guangdong Science and Technology Program (2015A030312015), Shenzhen Science and Technology Program (JCYJ20170302153208613, JCYJ20151015151249564), NSERC (2015-05407), and unrestricted gifts from Intel Corporation and Google.

References

[BAR06] BAR-AVIV E., RIVLIN E.: Functional 3d object classification using simulation of embodied agent. In *British Machine Vision Conference* (2006), pp. 307–316. 7, 13

[BB95] BOGONI L., BAJCSY R.: Interactive recognition and representation of functionality. *Computer Vision and Image Understanding 62*, 2 (1995), 194–214. 2

[CWMD15] CHAO Y.-W., WANG Z., MIHALCEA R., DENG J.: Mining semantic affordances of visual object categories. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 4259–4267. 2

[DR13] DESAI C., RAMANAN D.: Predicting functional regions on objects. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops* (2013), pp. 968–975. 2

[FAvK*14] FISH N., AVERKIOU M., VAN KAICK O., SORKINE-HORNUNG O., COHEN-OR D., MITRA N. J.: Meta-representation of shape families. *ACM Trans. on Graph (SIGGRAPH) 33*, 4 (2014), 34:1–11. 4, 5, 7, 10, 18

[FCSF17] FU Q., CHEN X., SU X., FU H.: Pose-inspired shape synthesis and functional hybrid. *IEEE Trans. Visualization & Computer Graphics 23*, 12 (2017), 2574–2585. 18

[FDG*14] FOUHEY D. F., DELAITRE V., GUPTA A., EFROS A. A., LAPTEV I., SIVIC J.: People watching: Human actions as a cue for single view geometry. *International journal of computer vision 110*, 3 (2014), 259–274. 2

[FRS*12] FISHER M., RITCHIE D., SAVVA M., FUNKHOUSER T., HANRAHAN P.: Example-based synthesis of 3D object arrangements. *ACM Trans. on Graph (SIGGRAPH Asia) 31*, 6 (2012), 135:1–11. 7, 8, 17

[FSH11] FISHER M., SAVVA M., HANRAHAN P.: Characterizing structural relationships in scenes using graph kernels. *ACM Trans. on Graph (SIGGRAPH) 30*, 4 (2011), 34:1–12. 4, 7, 9, 12, 17

[FSL*15] FISHER M., SAVVA M., LI Y., HANRAHAN P., NIESSNER M.: Activity-centric scene synthesis for functional 3D scene modeling. *ACM Trans. on Graph (SIGGRAPH Asia) 34*, 6 (2015), 179:1–13. 4, 7, 13, 14, 17

[FvKBCO16] FISH N., VAN KAICK O., BERMANO A., COHEN-OR D.: Structure-oriented networks of shape collections. *ACM Trans. on Graph (SIGGRAPH Asia) 35*, 6 (2016), 171:1–14. 6, 7

[GF09] GOLOVINSKIY A., FUNKHOUSER T.: Consistent segmentation of 3D models. *Comput. Graph. (Proc. SMI) 33*, 3 (2009), 262–269. 6

[GG04] GELFAND N., GUIBAS L. J.: Shape segmentation using local slippage analysis. In *Proc. SGP* (2004), pp. 214–223. 5, 7, 10, 15

[GGVG11] GRABNER H., GALL J., VAN GOOL L.: What makes a chair a chair? In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2011), IEEE, pp. 1529–1536. 7, 13, 14, 16

[GMW16] GUERRERO P., MITRA N. J., WONKA P.: RAID: A relation-augmented image descriptor. *ACM Trans. on Graph (SIGGRAPH) 35*, 4 (2016), 46:1–12. 4, 10, 16, 17

[GSEH11] GUPTA A., SATKIN S., EFROS A. A., HEBERT M.: From 3D scene geometry to human workspace. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2011), IEEE, pp. 1961–1968. 2

[HFL12] HU R., FAN L., LIU L.: Co-segmentation of 3D shapes via subspace clustering. *Computer Graphics Forum (SGP) 31*, 5 (2012), 1703–1713. 6

[HLK*17] HU R., LI W., KAICK O. V., SHAMIR A., ZHANG H., HUANG H.: Learning to predict part mobility from a single static snapshot. *ACM Trans. on Graph (SIGGRAPH Asia) 36*, 6 (2017), 217:1–13. 7, 11, 12, 13, 16

[HvKW*16] HU R., VAN KAICK O., WU B., HUANG H., SHAMIR A., ZHANG H.: Learning how objects function via co-analysis of interactions. *ACM Trans. on Graph (SIGGRAPH) 35*, 4 (2016), 47:1–13. 7, 12, 16

[HWG14] Huang Q., Wang F., Guibas L.: Functional map networks for analyzing and exploring large shape collections. *ACM Trans. on Graph (SIGGRAPH) 33*, 4 (2014), 36:1–11. 6

[HZvK*15] Hu R., Zhu C., van Kaick O., Liu L., Shamir A., Zhang H.: Interaction context (ICON): Towards a geometric functionality descriptor. *ACM Trans. on Graph (SIGGRAPH) 34*, 4 (2015), 83:1–12. 4, 5, 6, 7, 10, 11, 12, 17

[JKS13] Jiang Y., Koppula H., Saxena A.: Hallucinated humans as the hidden context for labeling 3D scenes. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2013), IEEE, pp. 2993–3000. 7, 13, 14, 16

[KCGF14] Kim V. G., Chaudhuri S., Guibas L., Funkhouser T.: Shape2pose: Human-centric shape analysis. *ACM Trans. on Graph (SIGGRAPH) 33*, 4 (2014), 120:1–12. 5, 7, 14, 17

[KHS10] Kalogerakis E., Hertzmann A., Singh K.: Learning 3D mesh segmentation and labeling. *ACM Trans. on Graph (SIGGRAPH) 29*, 3 (2010), 102:1–12. 8

[KLM*13] Kim V. G., Li W., Mitra N. J., Chaudhuri S., DiVerdi S., Funkhouser T.: Learning part-based templates from large collections of 3D shapes. *ACM Trans. on Graph (SIGGRAPH) 32*, 4 (2013), 70:1–12. 6

[KLY*14] Koo B., Li W., Yao J., Agrawala M., Mitra N. J.: Creating works-like prototypes of mechanical objects. *ACM Trans. on Graph (SIGGRAPH Asia) 33*, 6 (2014), 217:1–9. 4, 18

[KS14] Kim D. I., Sukhatme G. S.: Semantic labeling of 3D point clouds with object affordance for robot manipulation. In *Int. Conf. Robotics and Automation (ICRA)* (2014), pp. 5578–5584. 7, 12, 15

[LCL06] Lee K. H., Choi M. G., Lee J.: Motion patches: building blocks for virtual environments annotated with motion data. *ACM Trans. on Graph (SIGGRAPH) 25*, 3 (2006), 898–906. 7, 14, 15

[LKWS16] Lun Z., Kalogerakis E., Wang R., Sheffer A.: Functionality preserving shape style transfer. *ACM Trans. on Graph 35*, 6 (2016), 209. 18

[LMS13] Laga H., Mortara M., Spagnuolo M.: Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes. *ACM Trans. on Graph 32*, 5 (2013), 150:1–16. 7, 12, 15

[LOMI11] Lau M., Ohgawara A., Mitani J., Igarashi T.: Converting 3D furniture models to fabricatable parts and connectors. *ACM Trans. on Graph (SIGGRAPH) 30*, 4 (2011), 85:1–6. 4, 18

[LWL*16] Li H., Wan G., Li H., Sharf A., Xu K., Chen B.: Mobility fitting using 4D RANSAC. *Computer Graphics Forum (SGP) 35*, 5 (2016), 79–88. 12

[MGP06] Mitra N. J., Guibas L., Pauly M.: Partial and approximate symmetry detection for 3D geometry. *ACM Trans. on Graph (SIGGRAPH) 25*, 3 (2006), 560–568. 8

[MLZ*16] Ma R., Li H., Zou C., Liao Z., Tong X., Zhang H.: Action-driven 3D indoor scene evolution. *ACM Trans. on Graph (SIGGRAPH Asia) 35*, 6 (2016), 173:1–13. 4, 7, 14, 15, 17

[MPWC12] Mitra N. J., Pauly M., Wand M., Ceylan D.: Symmetry in 3D geometry: Extraction and applications. In *Eurographics State-of-the-art Report (STAR)* (2012), Eurographics. 8

[MSL*11] Merrell P., Schkufza E., Li Z., Agrawala M., Koltun V.: Interactive furniture layout using interior design guidelines. *ACM Trans. on Graph (SIGGRAPH) 30*, 4 (2011), 87:1–10. 7, 8, 18

[MTFA15] Myers A., Teo C. L., Ferm¨uller C., Aloimonos Y.: Affordance detection of tool parts from geometric features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on* (2015), IEEE, pp. 1374–1381. 7, 12

[MWZ*13] Mitra N., Wand M., Zhang H., Cohen-Or D., Bokeloh M.: Structure-aware shape processing. In *Eurographics State-of-the-art Report (STAR)* (2013), Eurographics. 2, 6

[MXLH13] Meng M., Xia J., Luo J., He Y.: Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization. *Computer-Aided Design 45* (2013), 312–320. 6

[MYY*10] Mitra N. J., Yang Y.-L., Yan D.-M., Li W., Agrawala M.: Illustrating how mechanical assemblies work. *ACM Trans. on Graph (SIGGRAPH) 29*, 4 (2010), 58:1–12. 4, 7, 9, 18

[PKH*17] Pirk S., Krs V., Hu K., Rajasekaran S. D., Kang H., Yoshiyasu Y., Benes B., Guibas L. J.: Understanding and exploiting object interaction landscapes. *ACM Trans. on Graph 36*, 3 (2017), 31:1–14. 4, 5, 7, 11, 16

[PMW*08] Pauly M., Mitra N. J., Wallner J., Pottmann H., Guibas L. J.: Discovering structural regularity in 3D geometry. *ACM Trans. on Graph (SIGGRAPH) 27*, 3 (2008), 43:1–11. 8

[PSR08] Pechuk M., Soldea O., Rivlin E.: Learning function-based object classification from 3D imagery. *Computer Vision and Image Understanding 110*, 2 (2008), 173–191. 7, 10, 15

[RDR95] Rivlin E., Dickinson S. J., Rosenfeld A.: Recognition by functional parts. *Computer Vision and Image Understanding 62*, 2 (1995), 164–176. 10

[RT16] Roy A., Todorovic S.: A multi-scale CNN for affordance segmentation in RGB images. In *European Conf. on Computer Vision* (2016), Springer, pp. 186–201. 2

[SCH*14] Savva M., Chang A. X., Hanrahan P., Fisher M., Niessner M.: SceneGrok: Inferring action maps in 3D environments. *ACM Trans. on Graph (SIGGRAPH Asia) 33*, 6 (2014), 212:1–10. 4, 5, 6, 7, 13, 14, 16, 17

[SCH*16] Savva M., Chang A. X., Hanrahan P., Fisher M., Niessner M.: PiGraphs: Learning Interaction Snapshots from Observations. *ACM Trans. on Graph (SIGGRAPH) 35*, 4 (2016), 139:1–12. 4, 5, 7, 14, 17

[SHL*14] Sharf A., Huang H., Liang C., Zhang J., Chen B., Gong M.: Mobility-trees for indoor scenes manipulation. *Computer Graphics Forum 33*, 1 (2014), 2–14. 12, 18

[SLR*16] Shao T., Li D., Rong Y., Zheng C., Zhou K.: Dynamic furniture modeling through assembly instructions. *ACM Trans. on Graph (SIGGRAPH Asia) 35*, 6 (2016), 172:1–15. 18

[SQX*16] Shu Z., Qi C., Xin S., Hu C., Wang L., Zhang Y., Liu L.: Unsupervised 3D shape segmentation and co-segmentation via deep learning. *Computer Aided Geometric Design 43* (2016), 39–52. 6

[Sul96] Sullivan L. H.: The tall office building artistically considered. *Lippincott's Magazine* (Mar. 1896), 403–409. 3

[SvKK*11] Sidi O., van Kaick O., Kleiman Y., Zhang H., Cohen-Or D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. on Graph (SIGGRAPH Asia) 30*, 6 (2011), 126:1–10. 6

[THW*14] Tevs A., Huang Q., Wand M., Seidel H.-P., Guibas L.: Relating shapes via geometric symmetries and regularities. *ACM Trans. on Graph (SIGGRAPH) 33*, 4 (2014), 119:1–12. 8

[UIM12] Umetani N., Igarashi T., Mitra N. J.: Guided exploration of physically valid shapes for furniture design. *ACM Trans. on Graph 31*, 4 (2012), 86–1. 18

[vKTS*11] van Kaick O., Tagliasacchi A., Sidi O., Zhang H., Cohen-Or D., Wolf L., , Hamarneh G.: Prior knowledge for part correspondence. *Computer Graphics Forum (Eurographics) 30*, 2 (2011), 553–562. 8

[vKXZ*13] van Kaick O., Xu K., Zhang H., Wang Y., Sun S., Shamir A., Cohen-Or D.: Co-hierarchical analysis of shape structures. *ACM Trans. on Graph (SIGGRAPH) 32*, 4 (2013), 69:1–10. 6

[WLY17] Wang H., Liang W., Yu L.-F.: Transferring objects: Joint inference of container and human pose. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2933–2941. 7, 13

[WWS*13] WU Z., WANG Y., SHOU R., CHEN B., LIU X.: Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering. *Computers & Graphics (Proc. SMI) 37* (2013), 628–637. 6

[WXL*11] WANG Y., XU K., LI J., ZHANG H., SHAMIR A., LIU L., CHENG Z., XIONG Y.: Symmetry hierarchy of man-made objects. *Computer Graphics Forum (Eurographics) 30*, 2 (2011), 287–296. 4, 8, 18

[XKHK17] XU K., KIM V. G., HUANG Q., KALOGERAKIS E.: Data-driven shape analysis and processing. *Computer Graphics Forum 36*, 1 (2017), 101–132. 2, 6, 19

[XLX*16] XU M., LI M., XU W., DENG Z., YANG Y., ZHOU K.: Interactive mechanism modeling from multi-view images. *ACM Trans. on Graph (SIGGRAPH Asia) 35*, 6 (2016), 236:1–13. 4, 7, 9, 18

[XMZ*14] XU K., MA R., ZHANG H., ZHU C., SHAMIR A., COHEN-OR D., HUANG H.: Organizing heterogeneous scene collection through contextual focal points. *ACM Trans. on Graph (SIGGRAPH) 33*, 4 (2014), 35:1–12. 5, 9, 17

[XSF02] XU K., STEWART J., FIUME E.: Constraint-based automatic placement for scene composition. In *Proc. Graphics Interface* (2002), vol. 2, pp. 25–34. 7, 8, 17

[YFF12] YAO B., FEI-FEI L.: Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses. *Trans. on Pattern Analysis and Machine Intelligence 34*, 9 (2012), 1691–1703. 2

[YK14] YUMER M. E., KARA L. B.: Co-constrained handles for deformation in shape collections. *ACM Trans. on Graph (SIGGRAPH Asia) 33*, 6 (2014), 187:1–11. 4, 7, 10, 18

[YMFF13] YAO B., MA J., FEI-FEI L.: Discovering object functionality. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2013), pp. 2512–2519. 2

[YYT*11] YU L.-F., YEUNG S. K., TANG C.-K., TERZOPOULOS D., CHAN T. F., OSHER S.: Make it home: automatic optimization of furniture arrangement. *ACM Trans. on Graph (SIGGRAPH) 30*, 4 (2011), 86:1–12. 7, 8, 17

[ZCK17] ZHAO X., CHOI M. G., KOMURA T.: Character-object interaction retrieval using the interaction bisector surface. *Computer Graphics Forum (Eurographics) 36*, 2 (2017), 119–129. 7, 14, 15

[ZCOAM14] ZHENG Y., COHEN-OR D., AVERKIOU M., MITRA N. J.: Recurring part arrangements in shape collections. *Computer Graphics Forum 33*, 2 (2014), 115–124. 6

[ZCOM13] ZHENG Y., COHEN-OR D., MITRA N. J.: Smart variations: Functional substructures for part compatibility. *Computer Graphics Forum (Eurographics) 32*, 2pt2 (2013), 195–204. 7, 9, 10, 18

[ZFFF14] ZHU Y., FATHI A., FEI-FEI L.: Reasoning about object affordances in a knowledge base representation. In *European Conf. on Computer Vision* (2014), Springer, pp. 408–424. 2

[ZHG*16] ZHAO X., HU R., GUERRERO P., MITRA N., KOMURA T.: Relationship templates for creating scene variations. *ACM Trans. on Graph (SIGGRAPH Asia) 35*, 6 (2016), 207:1–13. 7, 9, 18

[ZJZ*16] ZHU Y., JIANG C., ZHAO Y., TERZOPOULOS D., ZHU S.-C.: Inferring forces and learning human utilities from videos. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2016), IEEE, pp. 3823–3833. 7, 13, 14, 16

[ZLDM16] ZHENG Y., LIU H., DORSEY J., MITRA N. J.: Ergonomics-inspired reshaping and exploration of collections of models. *IEEE Trans. Visualization & Computer Graphics 22*, 6 (2016), 1732–1744. 7, 14, 18

[ZWK14] ZHAO X., WANG H., KOMURA T.: Indexing 3D scenes using the interaction bisector surface. *ACM Trans. on Graph 33*, 3 (2014), 22:1–14. 4, 6, 7, 9, 10, 17

[ZZ13] ZHAO Y., ZHU S.-C.: Scene parsing by integrating function, geometry and appearance models. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2013), pp. 3119–3126. 2

[ZZCZ15] ZHU Y., ZHAO Y., CHUN ZHU S.: Understanding tools: Task-oriented object modeling, learning and recognition. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 2855–2864. 7, 13