ShapeExplorer: Querying and Exploring Shapes using Visual Knowledge

Tong Ge[1], Yafang Wang[1]*, Gerard de Melo[2], Zengguang Hao[1], Andrei Sharf[3], Baoquan Chen[1]
[1]Shandong University, China; [2]Tsinghua University, China; [3]Ben-Gurion University, Israel

ABSTRACT

With unprecedented amounts of multimodal data on the Internet, there is an increasing demand for systems with a more fine-grained understanding of visual data. ShapeExplorer is an interactive software tool based on a detailed analysis of images in terms of object shapes and parts. For instance, given an image of a donkey, the system may rely on previously acquired knowledge about zebras and dogs to automatically locate and label the head, legs, tail, and so on. Based on such semantic models, ShapeExplorer can then generate morphing animations, synthesize new shape contours, and support object part-based queries as well as clipart-based image retrieval.

Keywords

Shape Knowledge Harvesting, Shape Matching, Shape Segmentation, Shape Synthesis

1. INTRODUCTION

In recent years, we have seen an explosion in the availability of multimodal data on the Internet, driven mostly by the ubiquity of mobile devices and online sharing platforms. Despite great advances in tasks such as object detection and tracking and multimedia retrieval, we still lack systems that provide more fine-grained semantic analyses of visual data.

In their widely noted work, Deng et al. [4] introduced ImageNet, a hierarchical organization of visual knowledge in raw images, according to semantic categories and relations. We take a further step in this direction and utilize the semantics of individual parts, subparts, and their shapes to facilitate their interpretation and manipulation. We present ShapeExplorer, an interactive software tool that analyzes images of objects and locates and labels specific object parts. For instance, given an image of a donkey, it can draw on previously analyzed images of related objects, e.g. of zebras or even just of dogs, to infer the location and labels of likely parts such as the head, legs, tail, and so on. An analysis in terms of parts is motivated by extant evidence from cognitive research on human vision showing that shape parts play an important role in the lower stages of object recognition [9]. Seeing a small part of an object often suffices for a human to be able to recognize the object, provided that the part is sufficiently unique [3, 2]. Still, fine-grained shape understanding remains a challenging problem in computer vision. It appears that richer data is necessary so that systems can be equipped with the required background knowledge.

Independently from the developments in computer vision, there has been considerable progress on automatically constructing knowledge bases (KB), utilizing textual information to extract relational facts and attributes. Examples include YAGO [10, 12], DBpedia [1], Freebase (www.freebase.com), ConceptNet [7], and WebChild [11]. Often, the backbone of such KBs is a taxonomy of entity types or of part-whole relationships (e.g., Head *isPartOf* Horse).

In our work, we have constructed a visual knowledge base called PartNet, for object parts and their shapes. PartNet semantically describes objects in terms of their classes, parts, and visual appearance. Unlike regular KBs, it gathers examples of the shape contours of objects and object parts.

Based on this, ShapeExplorer provides several higher-level operations, including (partial) shape querying, semantic morphing, shape synthesis, and part-based image retrieval using cliparts.

*Corresponding author: yafang.wang@sdu.edu.cn

2. FRAMEWORK

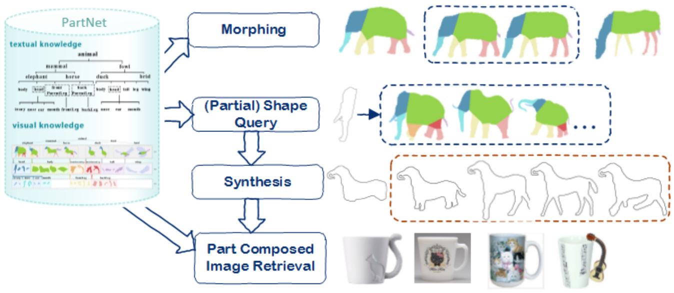

Figure 1: Flow diagram

Hierarchical Part Exploration. Figure 1 provides an overview of ShapeExplorer's operational flow. The system is based on the PartNet knowledge repository, which users can explore hierarchically. This knowledge is also used in several applications such as morphing and querying.

PartNet is organized according to taxonomic categories (animals, dinosaurs, home appliances, etc.), sub-categories (mammals, brontosauruses, chairs, etc.), and their part decompositions. At each level, the system presents corresponding shapes that the user may select and analyze. Internally, these are stored as subject-predicate-object triples, similar to regular knowledge bases, but including multimodal items.

There are two main user interfaces. Figure 3 presents a screenshot of the primary control center for part knowledge exploration. On the left side, the user can explore the knowledge in a convenient hierarchical tree based on categories (animals, home appliances, etc.), sub-categories (mammals, chairs, etc.), and their part decompositions. The right side of the screen serves as a working area. The users drags shapes into the bottom part of that area and can then select from several operations. These include an image analysis to infer the segmentation and labeling, which we describe below, as well as higher-level applications such as querying, morphing, and automated synthesis and completion (described in Section 3). Another user interface, shown in Figure 4, is used for part-based image retrieval. Users may compose a clipart-style query based on object parts and the system retrieves matching images from the database (see Section 3).

Image Analysis. ShapeExplorer's image analysis is based on a joint inference procedure for joint classification, segmentation, and labeling, leveraging visual knowledge from previously analyzed images. In order to bootstrap this process in a particular domain, a small number of manually annotated seed images need to have been fed to the system initially. For those initial seeds, the user manually provides an image label, a segmentation, and part labels for the segments. The labels are chosen from the WordNet taxonomy [5]. For instance, the user could mark the image as portraying an *elephant*, and then the individual segments can be annotated with part labels such as *head*, *tail*, etc.

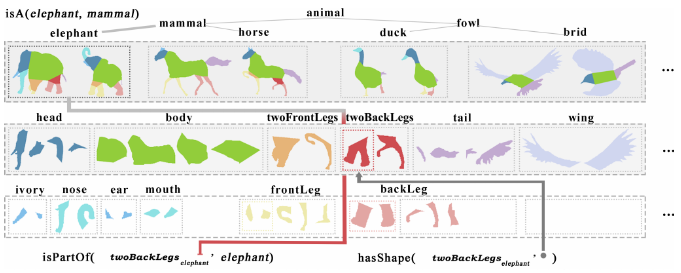

Figure 2: ShapeExplorer's hierarchical organization

After that, one can progressively augment ShapeExplorer's collected knowledge by adding new images, which the system analyzes using a transfer learning strategy. The system first generates a raw set of segmentation candidates, considering merely the image geometry using the short cut strategy [8]. It then matches the image with similar, previously seen ones using the inner-distance method [6], described in more detail later on in Section 3. From the top-3 matching images, we transfer additional candidate segmentations based on the contour alignments. The resulting set of candidate segmentations is pruned using semantic constraints.

Next, we determine label hypotheses for the image and for the parts based on the top-5 matching images in terms of the inner-distance method. For each cut/label candidate, we compute a confidence score based on the probability of the label within the top-5 matches and based on the cost of the contour point alignment. Finally, we perform a joint optimization step, relying on an Integer Linear Program to maximize the sum of confidence scores for the chosen candidates subject to compatibility and cardinality constraints.

In Figure 2, we illustrate the kind of knowledge that ShapeExplorer collects. The general taxonomy comes from WordNet, while image representations are derived from the seeds and the subsequent joint inference procedure for new images.

Implementation Details. ShapeExplorer is implemented as a web application with a JavaScript-driven browser interface that users can access via their web browsers. The back-end is implemented in Java and includes the powerful PartNet shape part knowledge base as well as images indexed using Lire[1].

3. APPLICATIONS

Based on the core framework, ShapeExplorer implements algorithms for several higher-level applications.

3.1 (Partial) Shape Queries

Users may provide an input image with an unknown shape and then ShapeExplorer attempts to match it against known shapes, retrieving top-k matches from the repository. The input image may be partial (i.e., with occlusions or missing parts). Thus, Alg. 1 considers subsets of the parts of every shape class as possible candidates. This is restricted to parts that are adjacent and sufficiently large, in order to avoid a combinatorial explosion. We use the inner-distance method [6] for similarity computation, which we found to be efficient, rotation-invariant, and robust. Given two shapes A and B, described by their contour point sequences $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_m$, respectively, we use χ^2 statistics to compare point histograms, resulting in a cost value $c(p_i, q_j)$. Then we solve for the optimal matching between A and B, denoted as $\pi : (p_i, q_{\pi(i)})$ using dynamic programming. We compute the minimum cost value as $C(\pi) = \sum_{i=1}^{n} c(i, \pi(i))$ and the number of matching points as $M(\pi) = \sum_{i=1}^{n} \delta(i)$, where $\delta(i) = 1$ if $\pi(i) \neq \emptyset$, and 0 otherwise. Finally, given a best matching shape, we define a segment cut, denoted as $\text{cut}_A(p_i, p_j)$, as the 2D line connecting contour points p_i,p_j in A. We use the computed shape matching π to map $\text{cut}_A(p_i, p_j)$ onto the input shape B as $\text{cut}_B(q_{\pi(i)}, q_{\pi(j)})$. Thus knowledge from existing images is transferred onto new ones to classify them and annotate their parts.

Algorithm 1 (Partial) Shape Querying

Input: connected input part and shape database, candidate object classes $\mathcal{C} = \{c_0, c_1, \ldots, c_{n_\mathcal{C}}\}$,number of results k
Output: top-k matching shapes

```
1: for each class $c_i \in \mathcal{C}$ do
2:     for each part $p_j$ of $c_i$ do
3:         $partialShape[] \leftarrow$ set of relevant combinations of parts of $p_j$
4:         for each part $ps_z \in partialShape[]$ do
5:             $cost[ps_z] \leftarrow$ matching cost $\mathcal{C}(\pi(part, ps_z))$
6: $shape[] \leftarrow$ ranking of shapes in $cost[]$ according to cost values
7: return top-k entries in $shape[]$
```

In Figure 3, the user selects an elephant head (blue) and a horse body (green) and synthesizes them together in the bottom right input region. This new shape is converted into a simple contour and given as input to ShapeExplorer.

[1]http://www.lire-project.net/

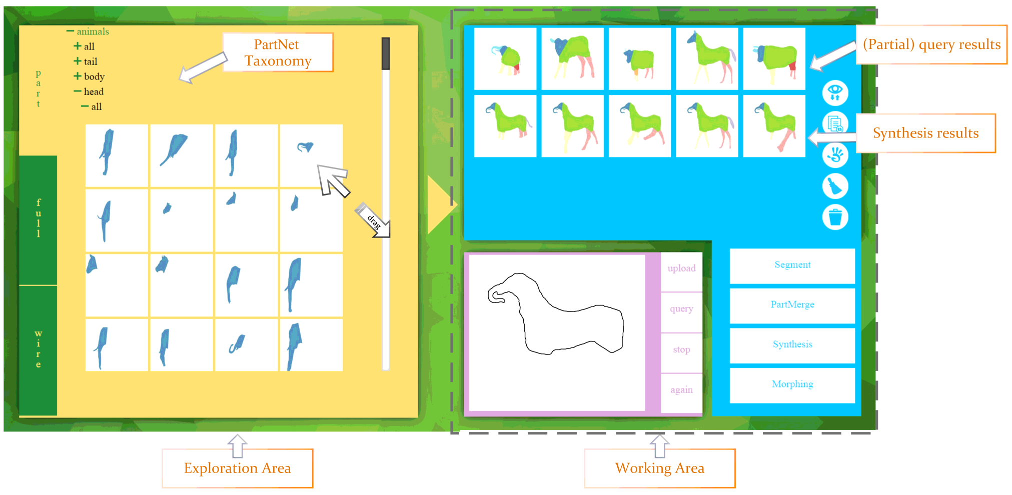

Figure 3: Screenshot for part knowledge exploration

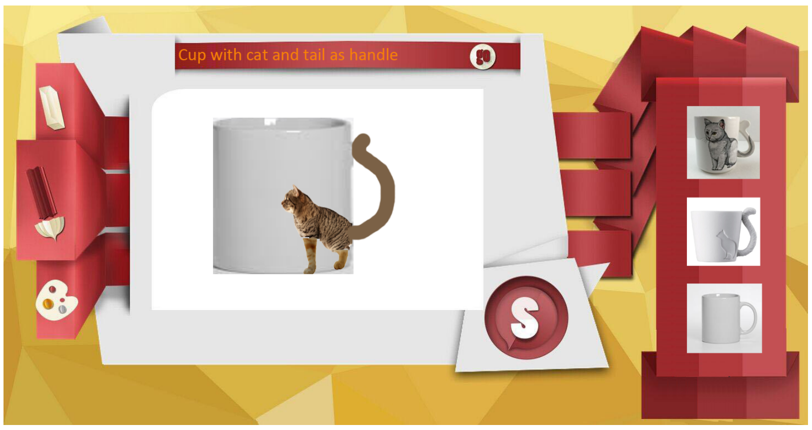

Figure 4: Screenshot for part-based clipart image retrieval

In partial query mode, the top-5 most similar results with respect to their partial similarity are shown in the top row on the right. These can be used to retrieve images with objects showing similar shapes.

3.2 Semantic Morphing

Given two images, ShapeExplorer can automatically perform a semantic form of morphing by classifying and performing a conformal joint segmentation of the two images. Alg. 2 first produces a segmentation of the two images into the same meaningful parts. This task is accomplished by searching for the lowest common cuts in the part hierarchies (e.g., a joint segmentation of a horse and elephant will return heads without the unique trunk of the elephant).

Algorithm 2 **Part Based Morphing**

```
Input: $shape_1$ and $shape_2$, part labels $\mathcal{L} = \{l_0, l_1, ..., l_n\}$
Output: morphing animation sequence
1: $part_1[] \leftarrow shapeSegmentation(shape_1)$
2: $part_2[] \leftarrow shapeSegmentation(shape_2)$
3: align $part_2[]$ to $part_1[]$ using common parts
4: for each label $l_i \in \mathcal{L}$ do
5:     $point_1[l_i][] \leftarrow samplePoints(part_1[l_i])$
6:     $point_2[l_i][] \leftarrow samplePoints(part_2[l_i])$
7: return $morphing(shape_1, shape_2, point_1[][], point_2[][])$
```

Having a full part correspondence between the two shapes, our system generates a morphing sequence which gradually interpolates from one to the other. See the morphing sequence in Figure 1 for an example of morphing from elephant to horse. This is accomplished by performing a per-part morphing while maintaining connectivity between adjacent parts during the deformation.

The morphing algorithm generates animations by smoothly interpolating transitions between corresponding parts in each shape (see morphing sequence in Figure 1). During the animation, users may pause and resume it to review intermediate frames, which can also serve as new inputs for querying and synthesis operations. Additionally, the user may select the intermediate frames as new inputs, which may further be queried, synthesized, and used to retrieve images.

3.3 Shape Synthesis and Completion

In shape synthesis mode, ShapeExplorer starts with a new user-provided partial shape and then uses best matching shapes from the repository to synthesize the missing parts so as to obtain a complete image.

Given an unknown shape, Alg. 3 first finds the top-1 best matching shape in the repository using the (Partial) Shape Query method. The segmentation and labels of the matching shape are transferred to the input image. Then, missing parts in the input shape with respect to the matched shape are detected. We synthesize the missing parts by transferring them from the matched shape onto the input image. Specifically, we subtract from the retrieved shape the parts in common with the input one and gracefully translate, scale, and rotate the shape and its parts so that they fit to their adjacent ones in the unknown one. Please note that *body* might have many part-cut labels, such as head, leg and tail. Therefore, line 6 means whether p_i and p_j have matching part-cut labels. For example, head can match body.

Algorithm 3 Shape Synthesis

```
Input: query shape, shape database
Output: new shape
1: $rParts[] \leftarrow$ parts of top-1 partial query result for $shape$ (Alg. 1)
2: $parts[] \leftarrow$ labeled segmentation of $shape$ via Alg. 1
3: $mParts[] \leftarrow rParts[] - parts[]$
4: for each part $p_i \in mParts[]$ do
5:     for each part $p_j \in parts[]$ do
6:         if $label(p_i)$ matches $label(p_j)$ then
7:             transfer from $p_i$ to $p_j$ in $shape$
8: return $shape$
```

In Figure 3, the user selects an elephant head (blue) and a horse body (green) and synthesizes them together in the bottom right input region. Synthesis results are displayed in the second row of the results region in the work area. These results can be used for retrieving similar images.

3.4 Clipart-based Image Retrieval

Another option is to perform image retrieval from a large image repository by composing a clipart-like query using parts or sketches. Our multimodal retrieval interface is composed of four components (see Figure 4): a control panel on the left, a text query field on the top, the working canvas in the middle, and the retrieval results on the right. Figure 5 illustrates the workflow of the part-based clipart image retrieval system. Users can issue textual queries to retrieve parts of interest from the PartNet knowledge repository. They can explore the results and drag parts of interest into the working area so as to craft a query image. This query image may be composed of parts stemming from different objects in the repository. Users also are able to modify the composition by drawing sketches using a pencil tool in conjunction with a color selection interface. This can be used to add additional items to the image, or to modify the original colors and textures of the parts coming from PartNet. An eraser tool is also provided for cleaning.

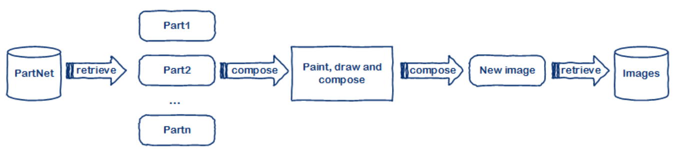

Figure 5: Part-based clipart image retrieval

Finally, the query image is used to retrieve similar images from the database. In the example in Figure 4, the user seeks to find images of cups with cat tails as handles. With standard image retrieval tools, it is hard to find such images unless they have sufficient textual metadata. With ShapeExplorer, the user can first issue a query for the word "cup" to find the cup body in the PartNet repository. This is then dragged from the results area onto the canvas. Similarly, the user finds the body and legs of a cat in PartNet and incorporates them into the query canvas. In order to ensure that the handle looks like a cat tail, the user can pick the color brown via the color selection interface and use the pencil tool to sketch a brown handle. This results in a sort of clipart image that can be used to retrieve matching real images from the database. For image matching, ShapeExplorer relies on a set of image features, combining JCD (Joint Composite Descriptor) and edge histograms. Thus, we can find cups with cat-like handles. Figure 4 shows the top-3 results, and the user may scroll to obtain further images.

4. CONCLUSION

In this paper, we have presented ShapeExplorer, a system aimed at fine-grained analyses of images in terms of object parts that captures multimodal knowledge in more detail than previous work. We see that explicit semantic representations of the parts enable several novel applications, including novel forms of querying, semantics-driven morphing, and synthesis.

Acknowledgments

We thank the anonymous reviewers for their valuable comments, and Kang Feng and Wei Wu et al. for their hard work preparing the dataset. This project was sponsored by National Natural Science Foundation of China (No. 61503217), Shandong Provincial Natural Science Foundation of China (No. ZR2014FP002), and The Fundamental Research Funds of Shandong University (No. 2014TB005, 2014JC001).

5. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. DBpedia: A nucleus for a web of open data. In *Proc. ISWC*, 2007.

[2] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[3] T. O. Binford. In *Proc. IEEE Conf. Systems & Control*.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. CVPR.*, 2009.

[5] C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[6] H. Ling and D. Jacobs. Shape classification using the inner-distance. *IEEE PAMI*, 2007.

[7] H. Liu and P. Singh. ConceptNet: A practical commonsense reasoning toolkit, 2004.

[8] L. Luo, C. Shen, X. Liu, and C. Zhang. A computational model of the short-cut rule for 2D shape decomposition. *CoRR*, abs/1409.2104, 2014.

[9] D. Marr. Early processing of visual information. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 275(942):483–519, 1976.

[10] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *Proc. WWW*, 2007.

[11] N. Tandon, G. de Melo, F. Suchanek, and G. Weikum. WebChild: Harvesting and organizing commonsense knowledge from the web. In *Proc. ACM WSDM*, 2014.

[12] Y. Wang, B. Yang, L. Qu, M. Spaniol, and G. Weikum. Harvesting facts from textual web sources by constrained label propagation. In *Proc. CIKM*, 2011.