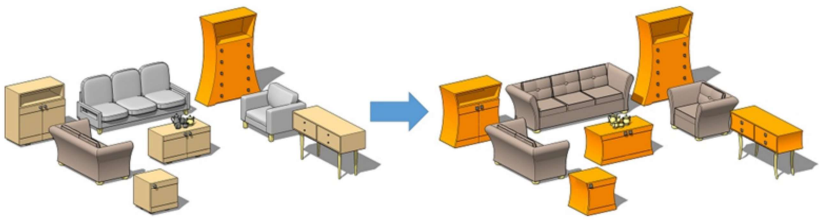
Functionality Preserving Shape Style Transfer

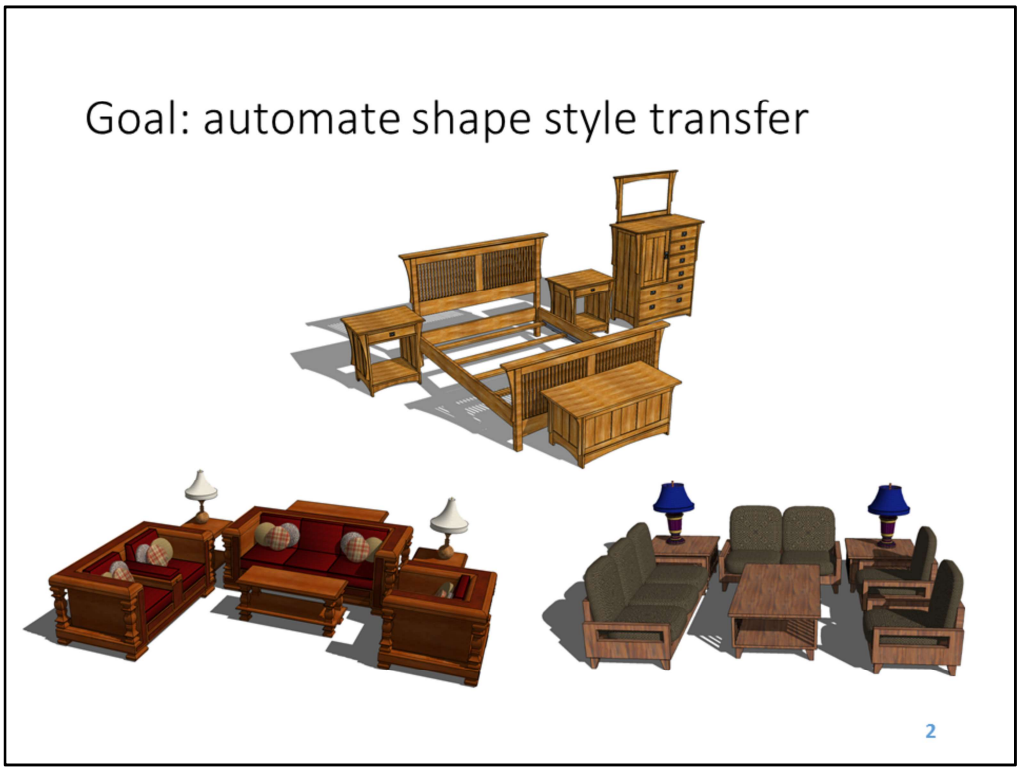


Zhaoliang Lun[1] Evangelos Kalogerakis[1] Rui Wang[1] Alla Sheffer[2]

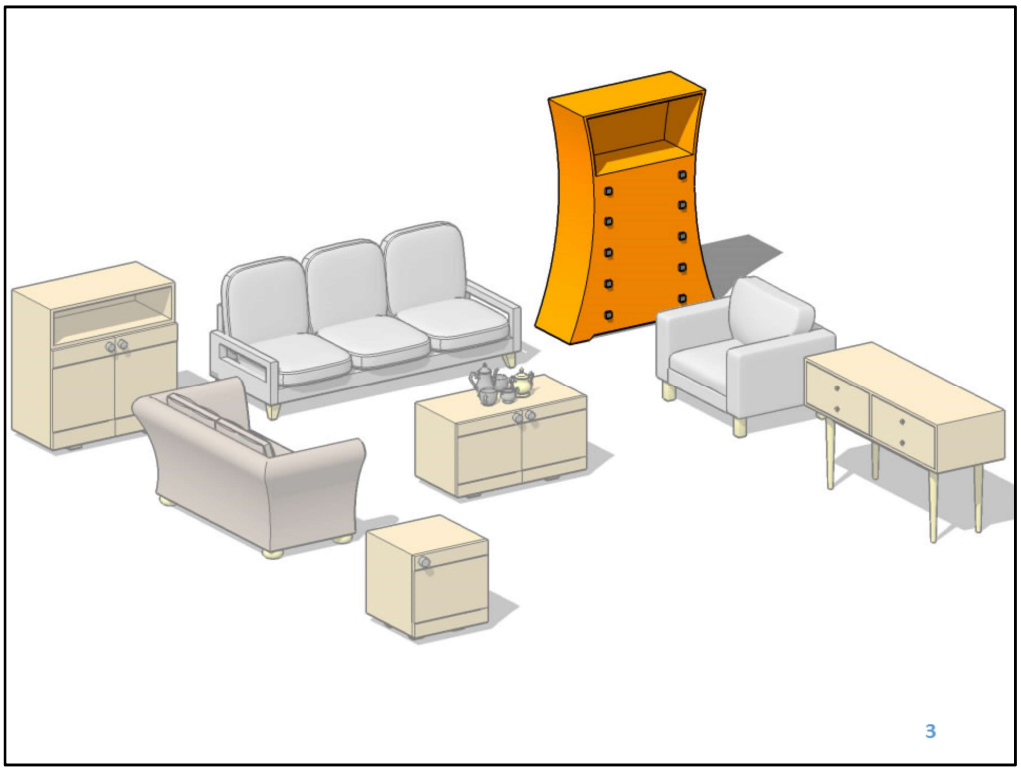
[1]University of Massachusetts Amherst [2]University of British Columbia

1

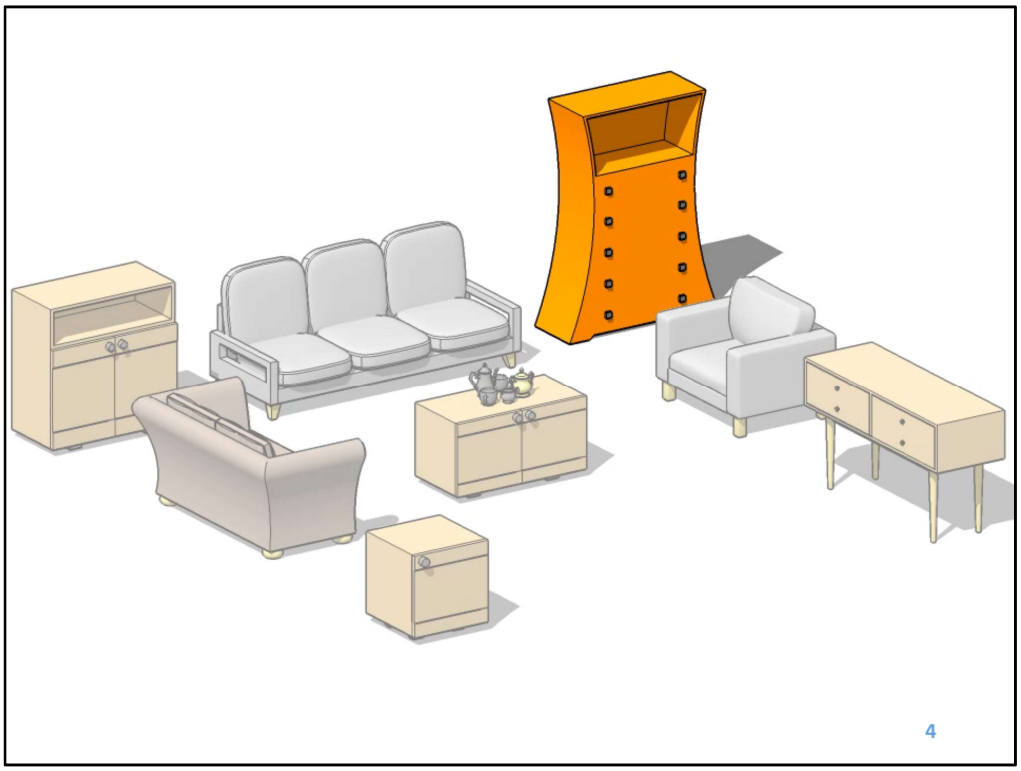
Hi, I am Zhaoliang Lun. Today I will present an algorithm for transferring geometric style between 3D shapes.



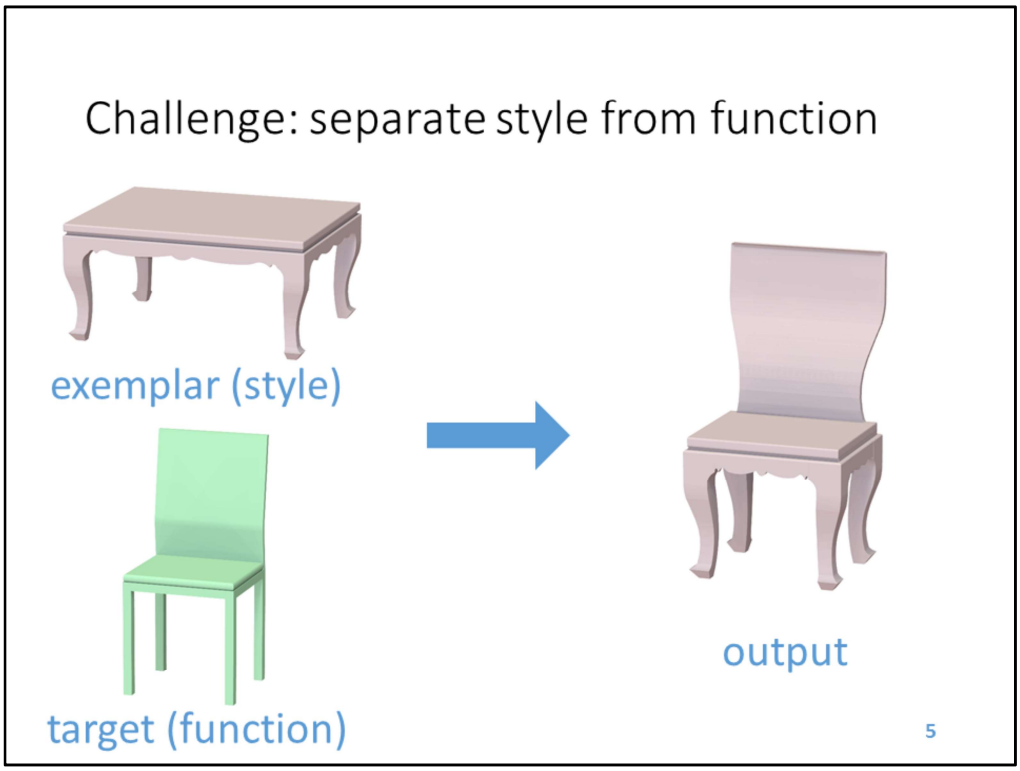
Human living spaces are often populated with shapes in a similar style. However, manually creating the underlying geometric representation of those style-coordinated objects requires lots of artistic expertise and is time consuming.



Our goal is to automate this process. Our algorithm synthesizes shapes by transferring geometric style between man-made objects with different structure and functionality. Users specify the desired style via an exemplar shape, **(CLICK)** such as the orange dresser here, and our method automatically transfers its style to functionally different target objects …



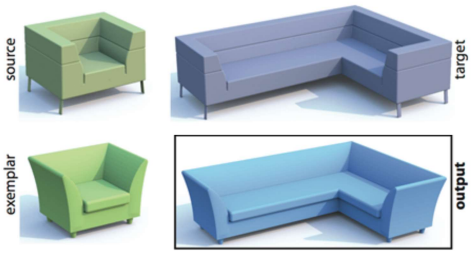
… such as the cabinet, coffee table, side table, and desk … It similarly transfers the style of the love-seat to the sofa and armchair … and the style of the sugar pot to the rest of the coffee set.



The main challenge of this problem is to separate shape style from function. The output shape should be stylistically as close as possible to the exemplar shape, while strictly maintaining the functionality encoded in the target shape.

Related work: style transfer

[Xu et al. 2010]

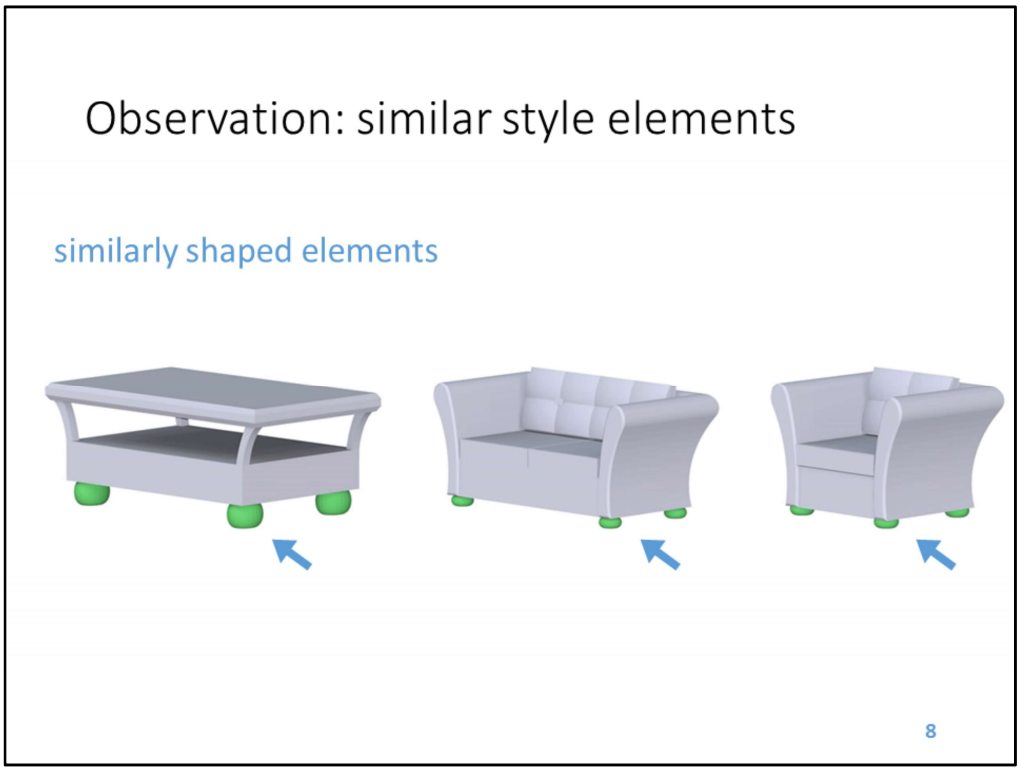


[Ma et al. 2014]

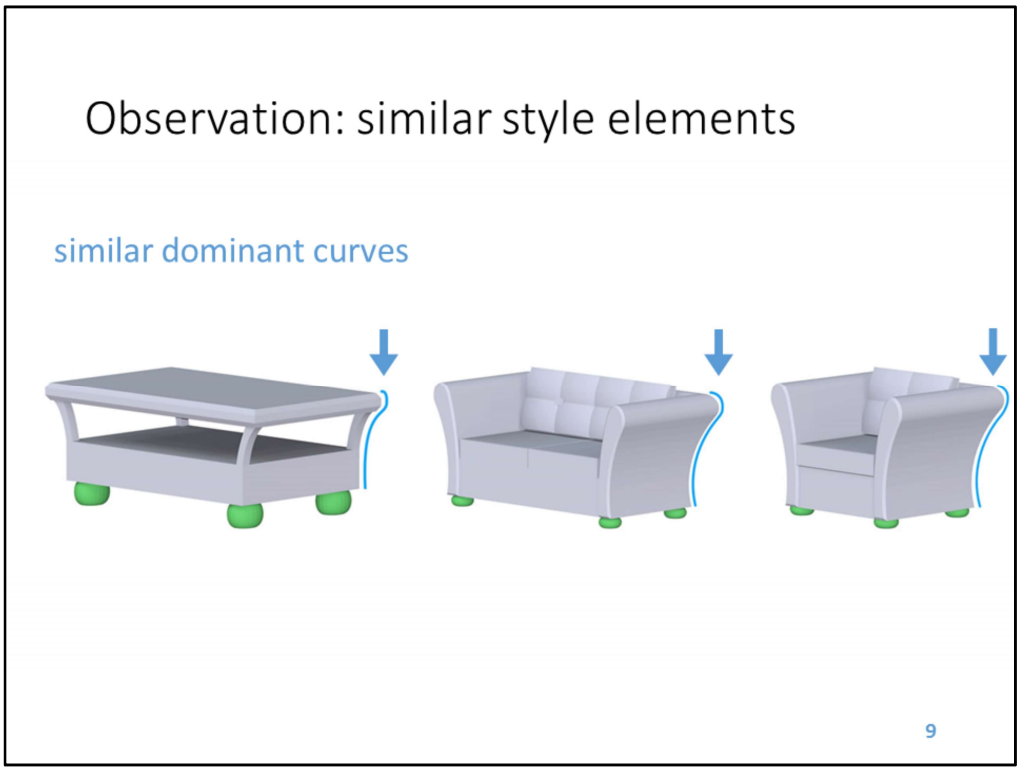
6

There are a few previous works attempting to achieve this goal. However, they addressed only very narrow special cases of 3D style transfer, either by transferring part scales across co-segmented shapes, or employing an extra source model as input for analogy-based style transfer. Instead, our algorithm is much more general than these approaches and requires less user input.

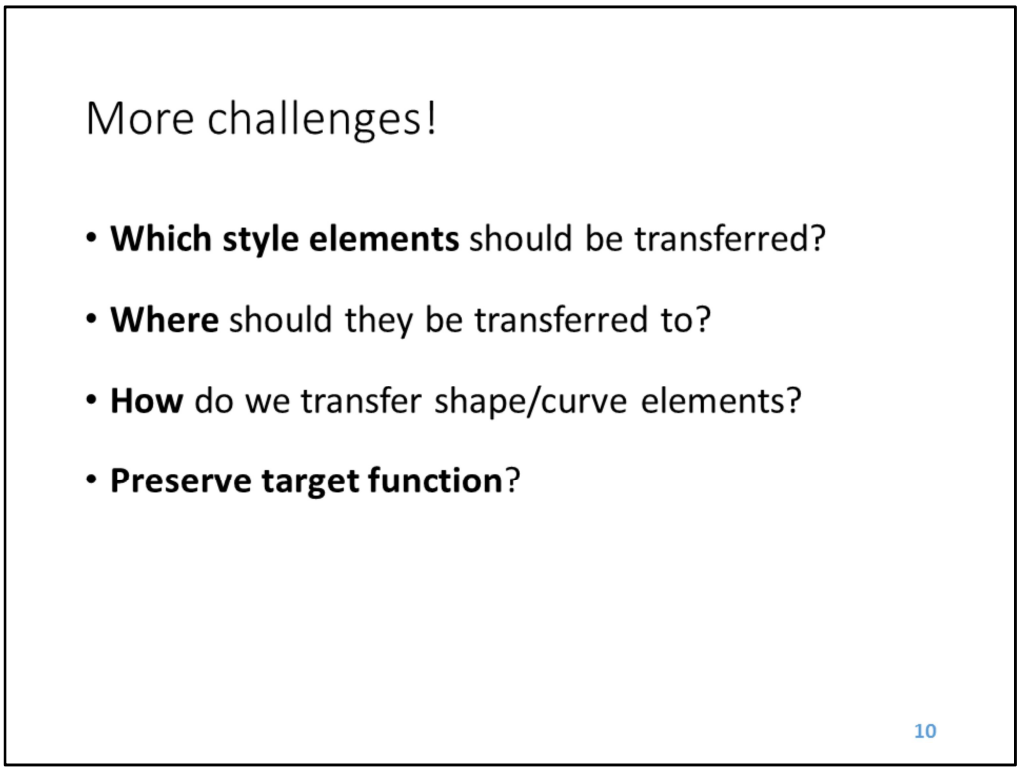
Our algorithm is motivated by observations about human perception of style in art history literature.



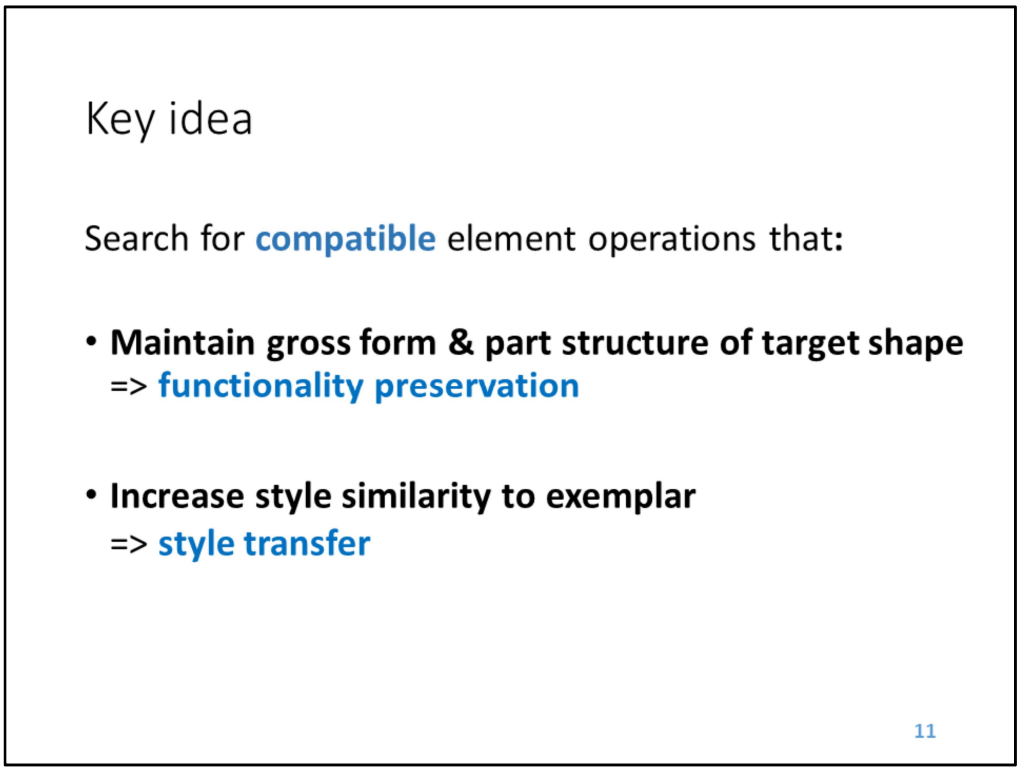
Same style objects frequently have similarly shaped elements such as the legs highlighted in green color here…



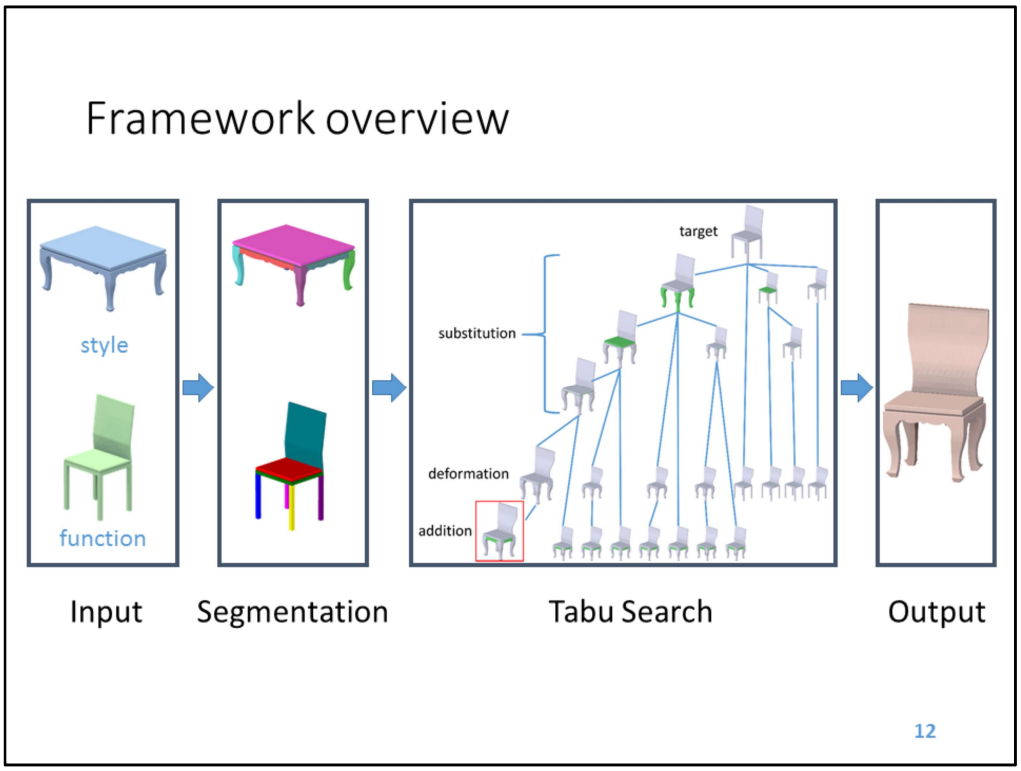
… as well as similar dominant curves such as the blue contour lines along the right side of the shapes



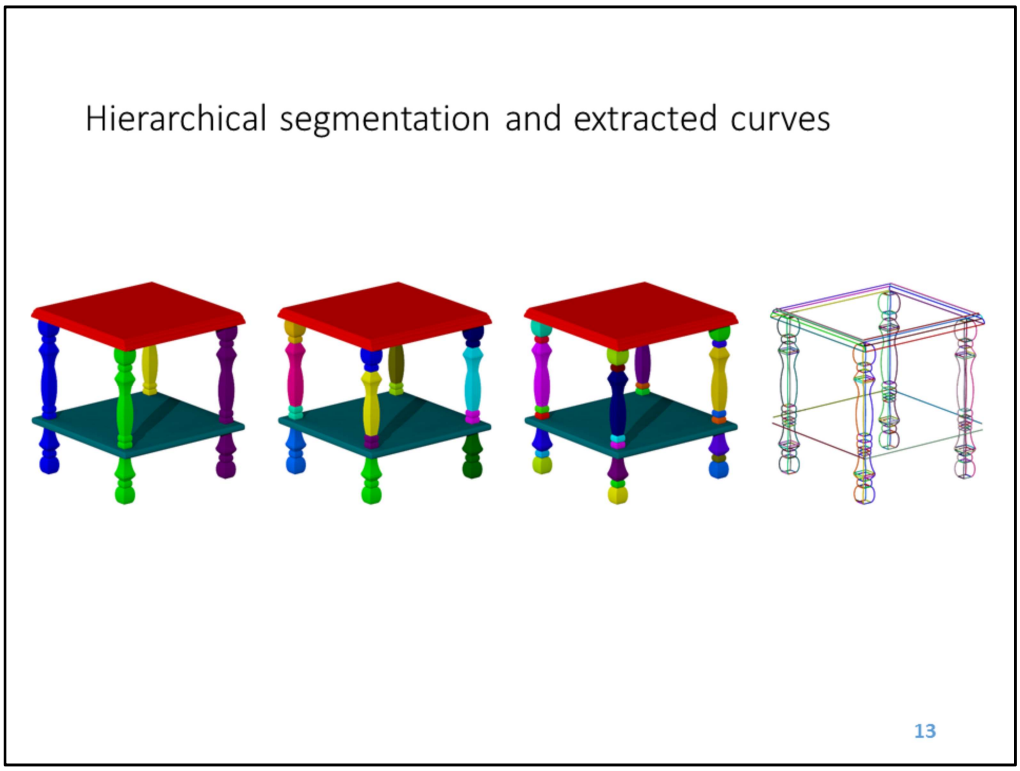
These observations pose even more challenges, making our problem hard to solve! For example, we need to detect candidate style elements in exemplar shapes **[CLICK]**, we need to determine where exactly, in the target shapes, they should be transferred to **[CLICK]**, determine how to transfer "shape-based" and "curve-based" elements **[CLICK]**, and finally we have to ensure that the target function will not break!



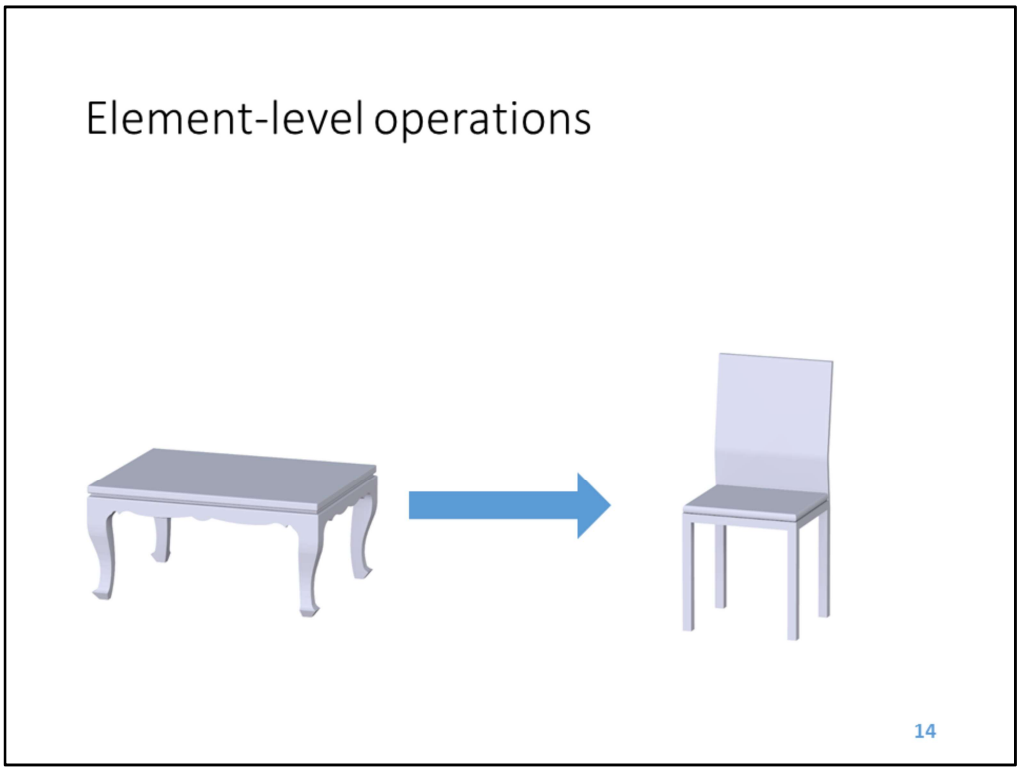
The key idea of our algorithm is to search for **compatible** element operations that have two goals **(CLICK)** first, preserve the functionality of the target shape by maintaining its gross form and arrangement of parts **(CLICK)** second, increase style similarity to the exemplar shape



We designed the following framework to implement these key ideas **(CLICK)** Given an exemplar shape encoding style, such as the blue table here, and a target shape encoding the desired function, such as the green chair here. **(CLICK)** our first step is to extract potential geometric elements through hierarchical segmentation, **(CLICK)** and then transfer elements from the exemplar to the target shape by searching for element-level modifications such as part substitution, addition, removal and curve-based deformation. The key idea of this search is to prohibit solutions that violate target functionality and progressively find solutions that increase style similarity to the exemplar **(CLICK)** The result of this procedure is a set of candidate solutions, such as the output chair that you see on the right.



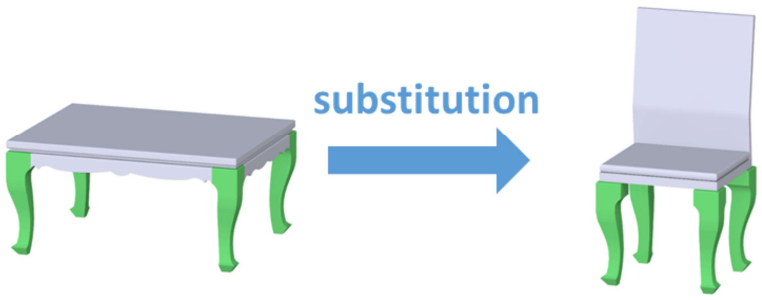
As I mentioned before, the first step of our framework is hierarchical segmentation. We segment input shapes into approximately convex patches at multiple scales, as demonstrated for this table. Then for each element, we compute ridges and valleys as well as occluding contours that serve as candidate curve elements, as you see on the right.



The second step of our framework is to transfer elements. I now describe the compatible element-level operations we use.

Element-level operations

- part substitution

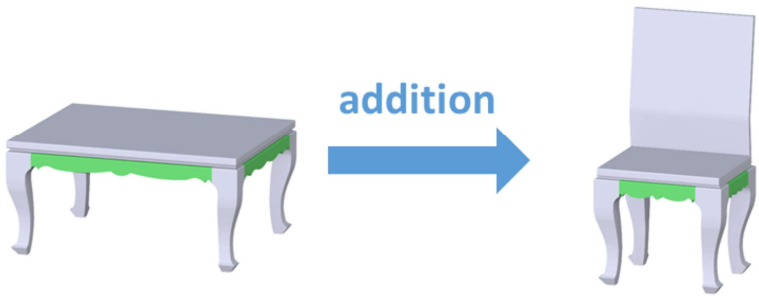


15

Our first type of element-level operations is element substitution such as the substitution of the legs you see here…

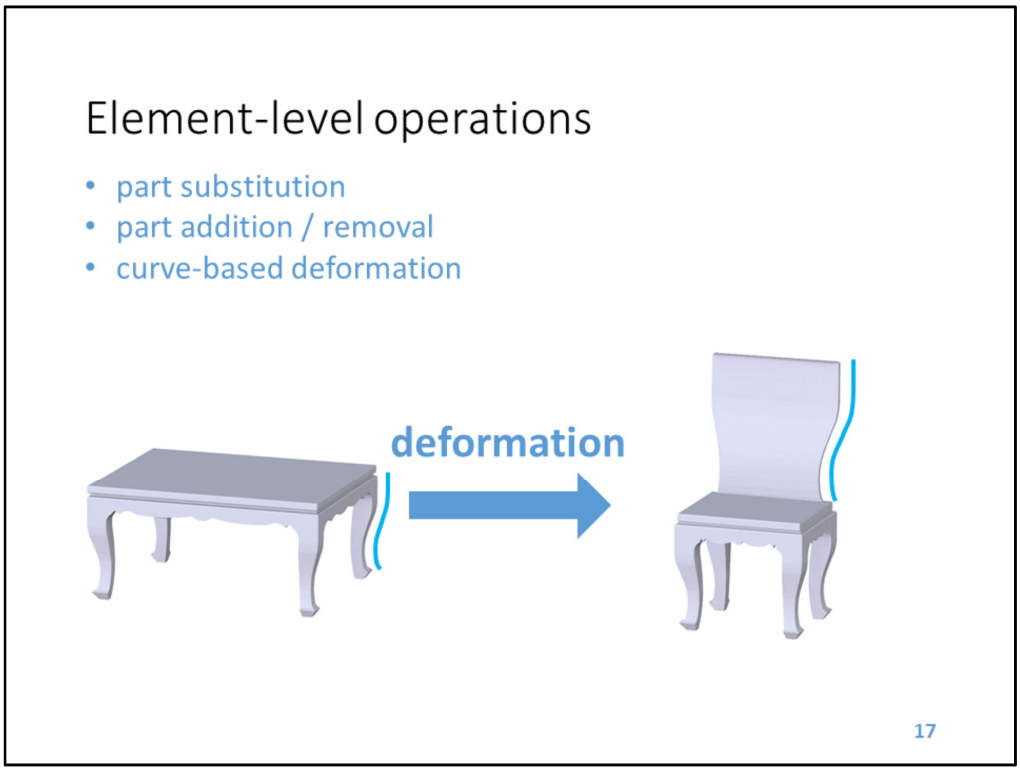
Element-level operations

- part substitution
- part addition / removal

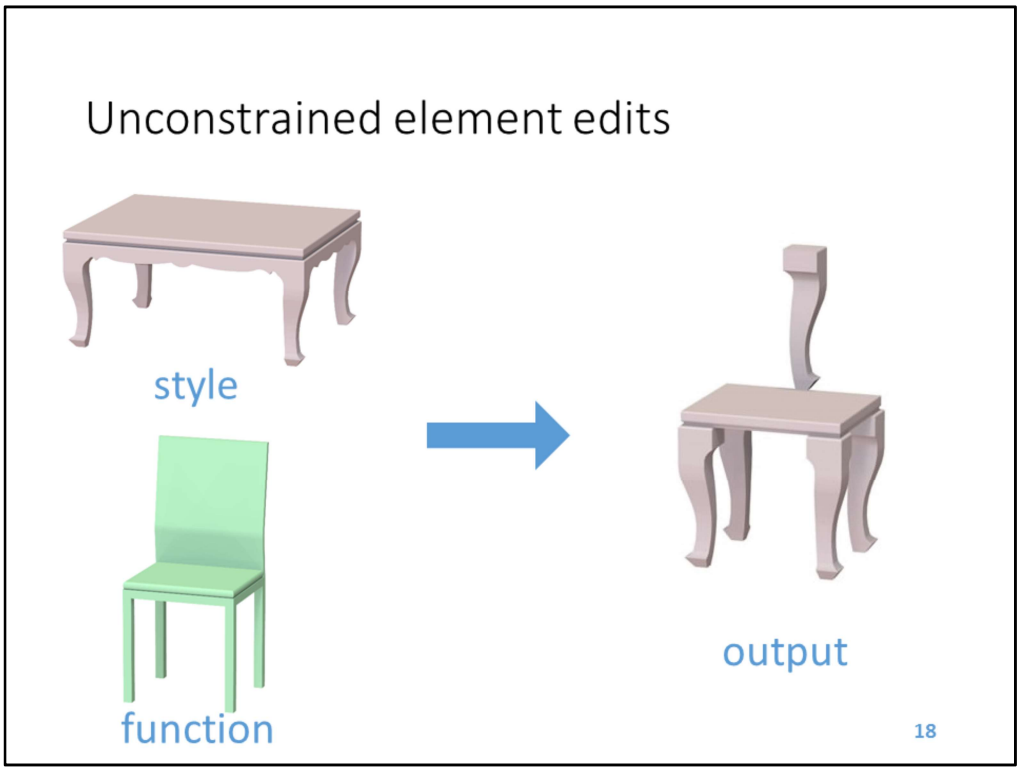


16

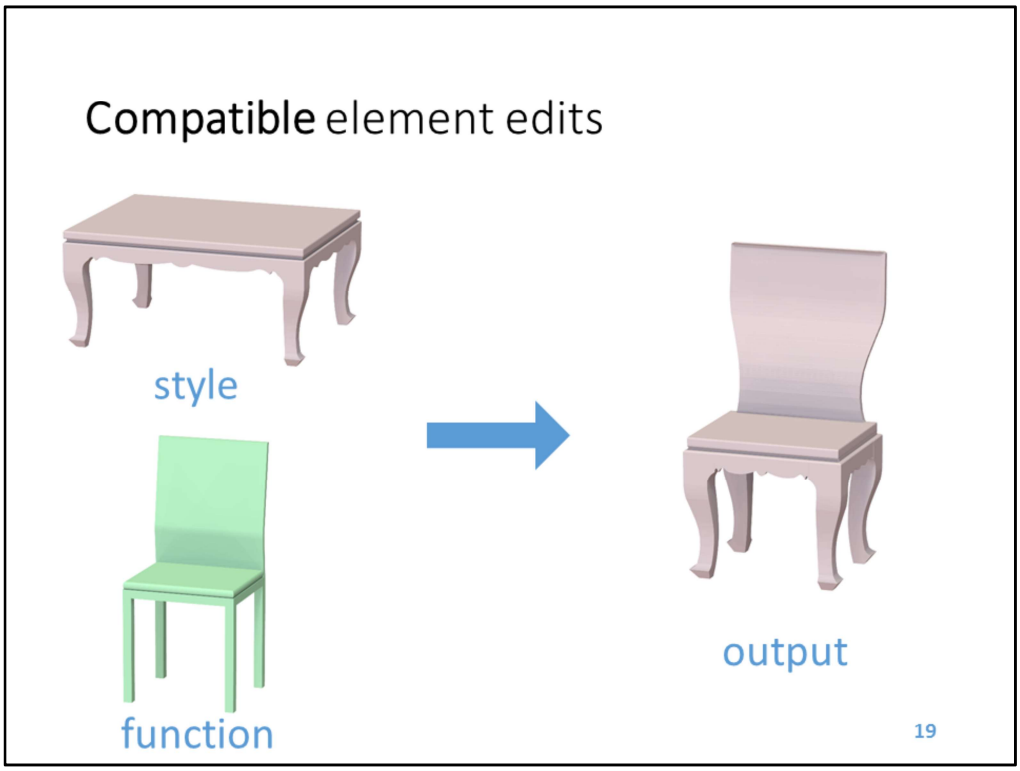
The second type of operations is element addition or removal. For example, here we add the decorative part of the table, shown in green, to the chair.



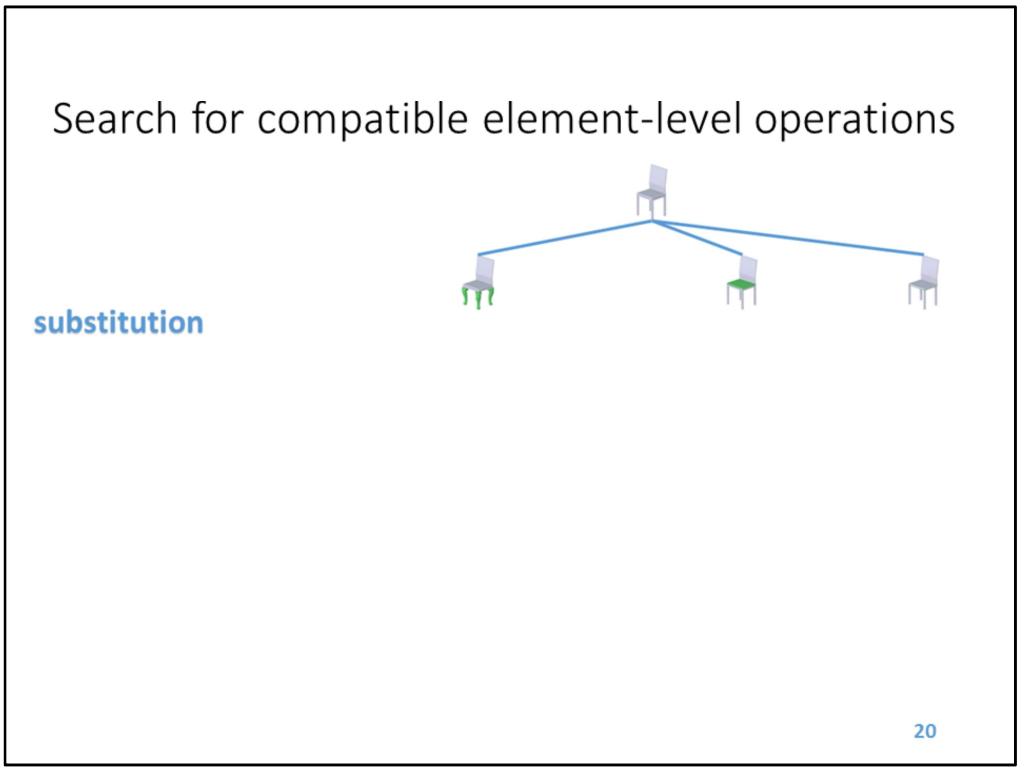
The third type of element-level operations is curve-based deformation. For example, this operation makes the geometry of the seat back match the table's feature curves



To create functional objects, element operations cannot be performed in an arbitrary manner. Arbitrary element edits can easily break the functionality of the output shape. **(CLICK)** For example, transferring the style of the exemplar table to the target chair without constraints yields a dysfunctional chair.



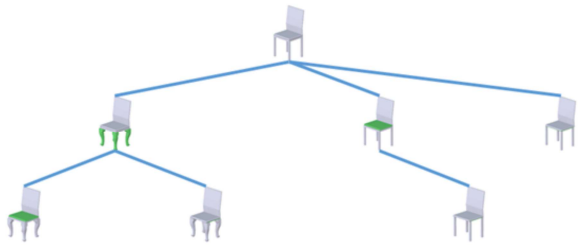
Instead, we need to find compatible editing operations that preserve the target functionality.



We use an optimization procedure to search for compatible element-level operations that strictly maintain target functionality, while progressively increasing style similarity to the exemplar.

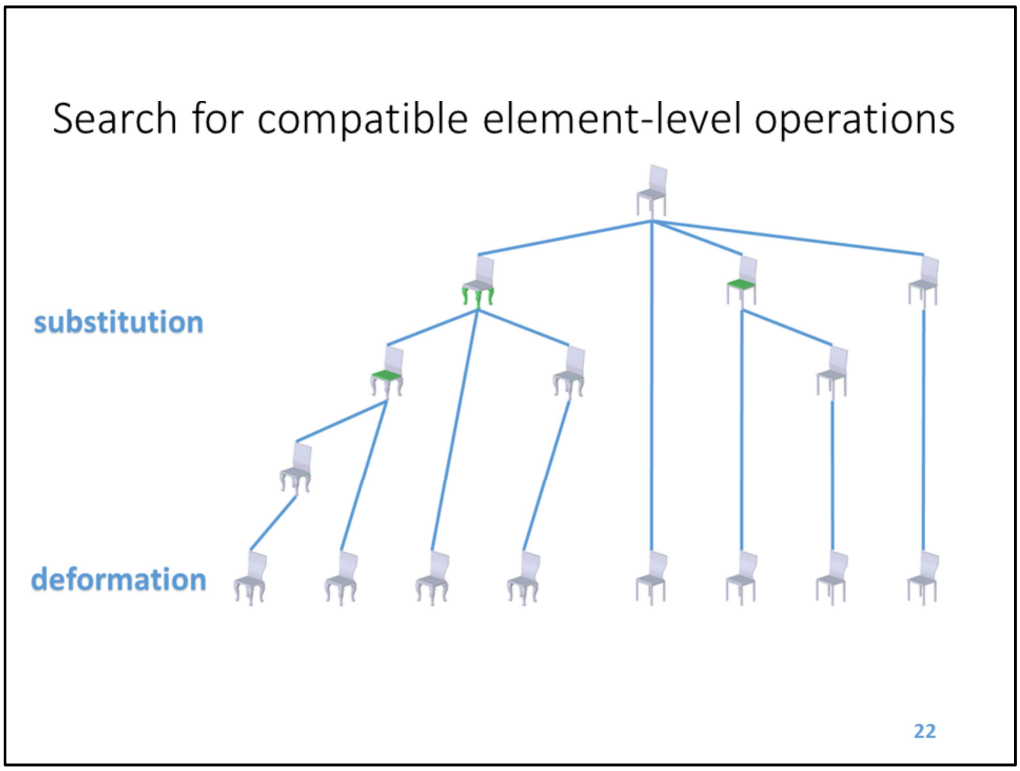
Search for compatible element-level operations

substitution

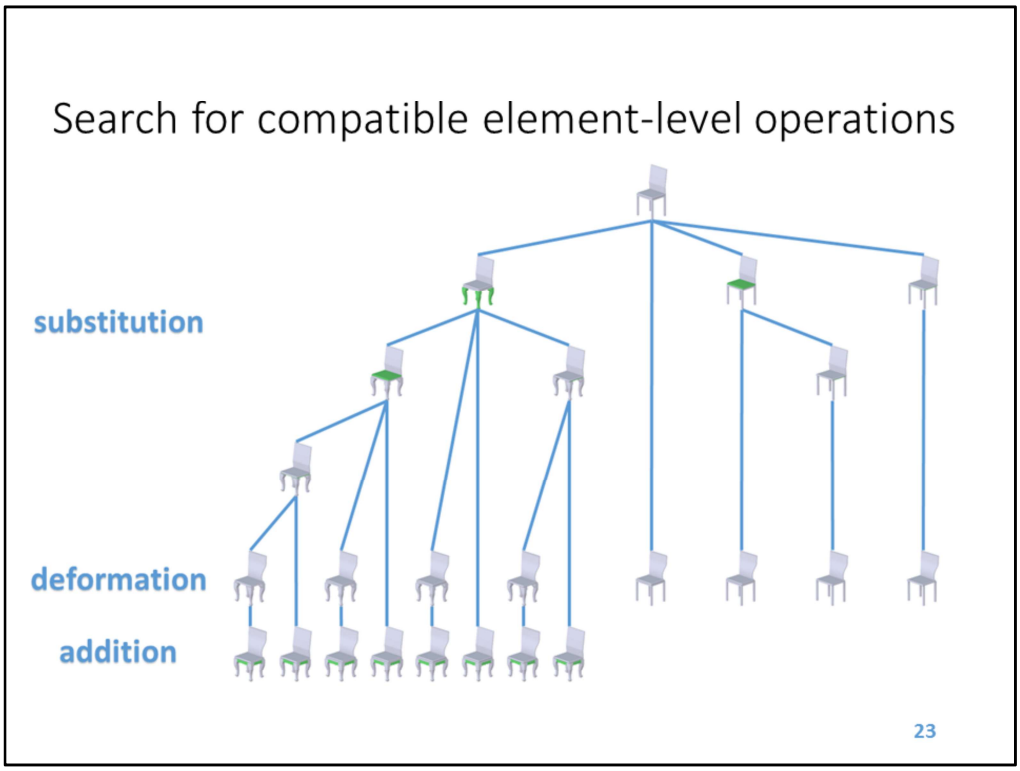


21

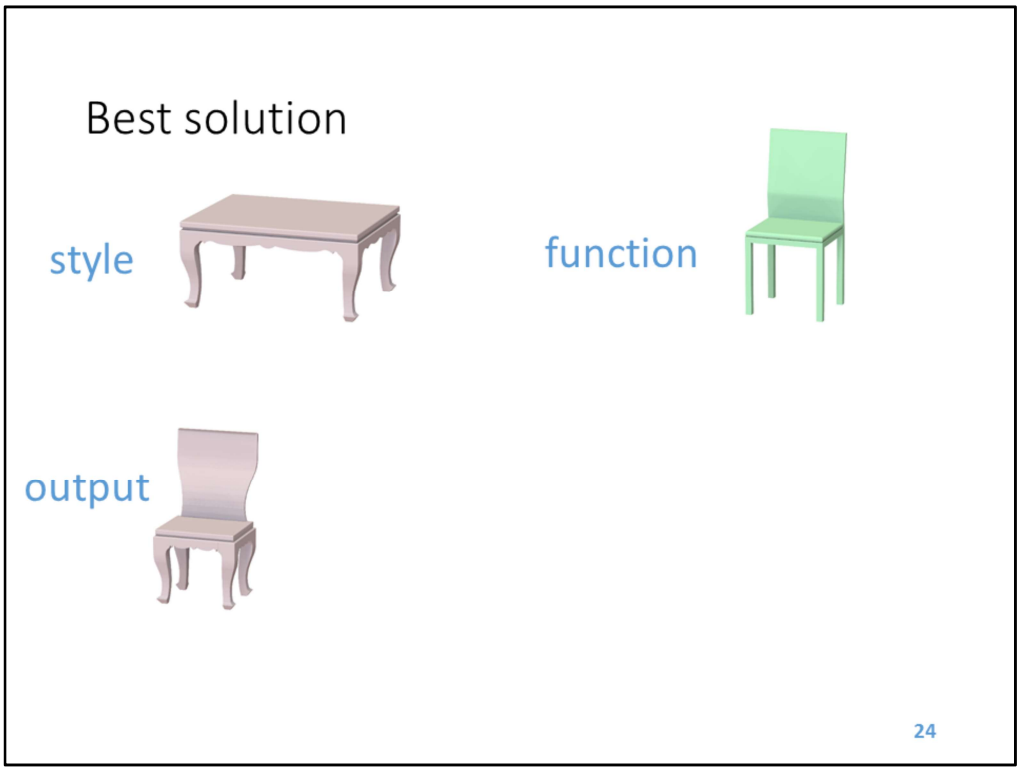
Here we visualize the search tree where each branch corresponds to one operation and each node represent one intermediate result.



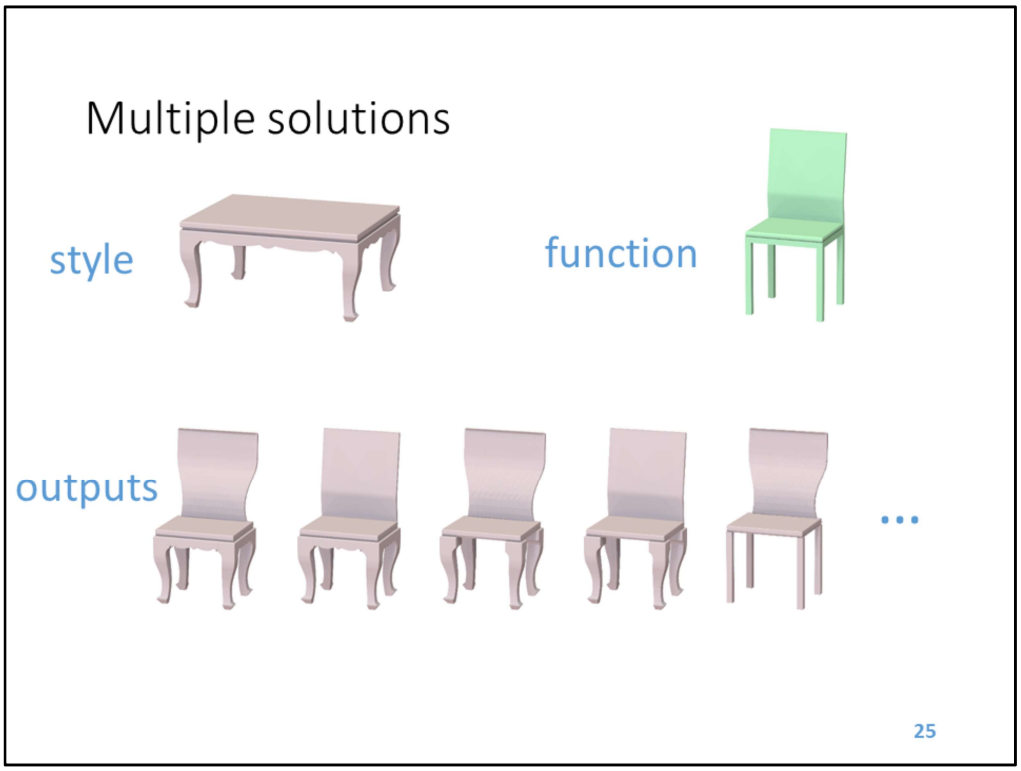
At each iteration, we use one of the element-level operations to bring the candidate shape closer to the exemplar in term of shape style similarity, while prohibiting operations which lead to functionally implausible shapes.



The iterations continue until no more style improvements can be performed on any shapes without breaking their functionality.



Our algorithm then provides the user with both the best result



as well as a ranked list of alternatives ranked according to style similarity to the exemplar.

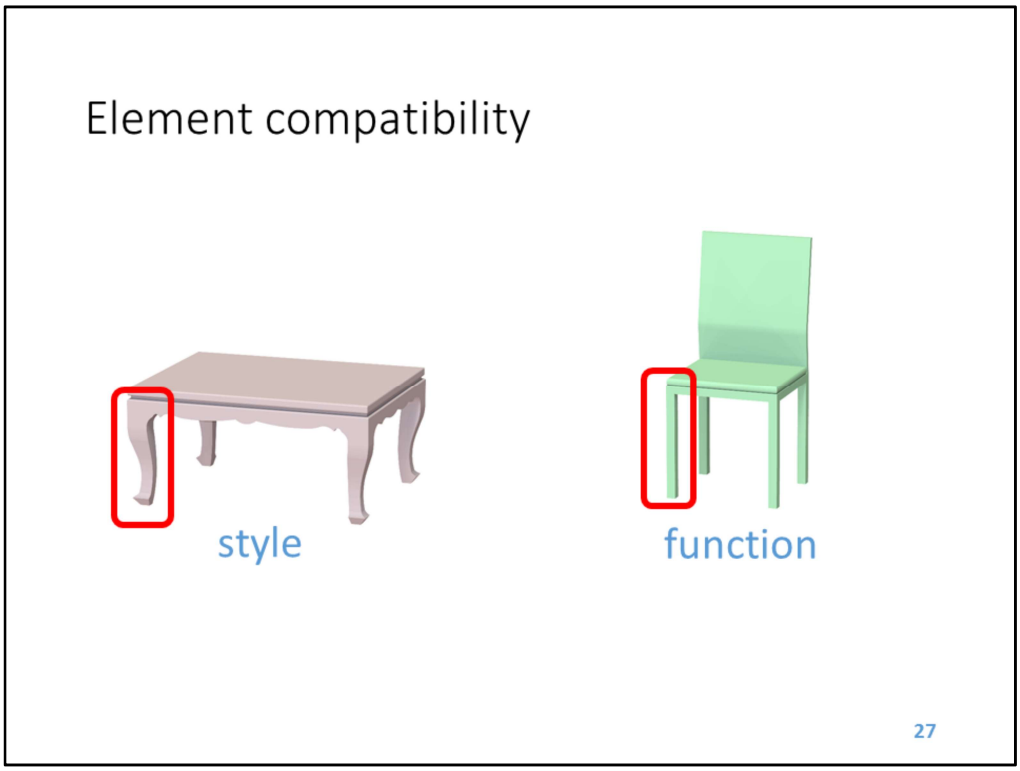
Measures

- **Functional compatibility measure**

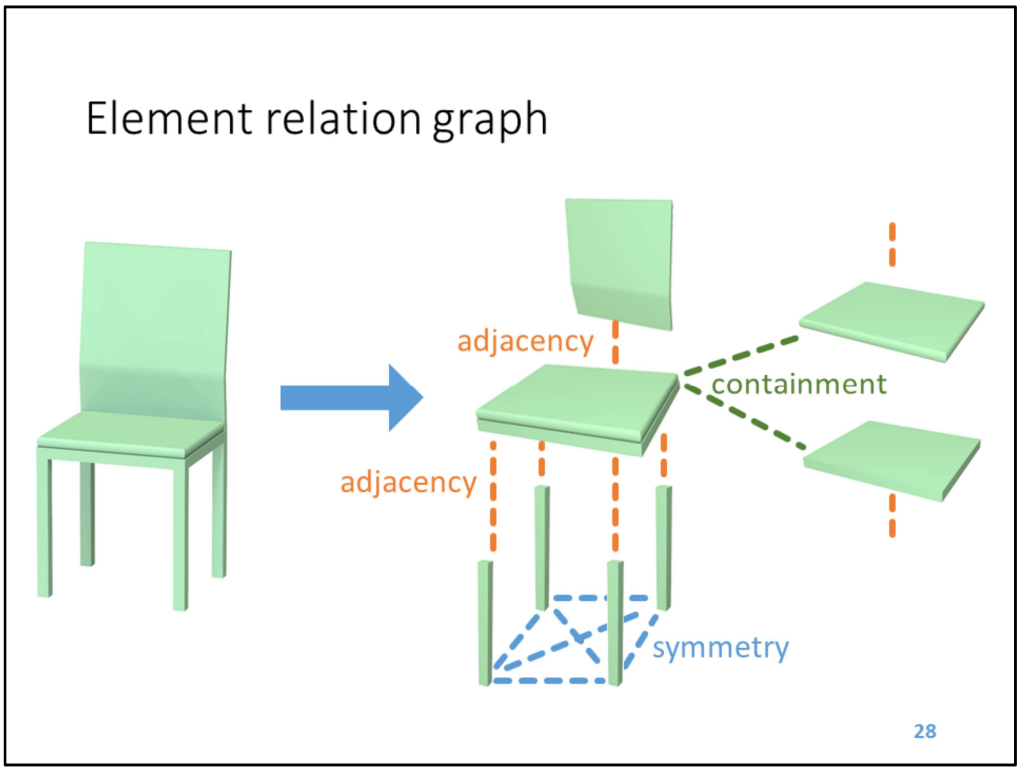
- Style similarity measure [Lun et al. 2015]

26

To execute the optimization loop, we need two measures: a measure that evaluates functional compatibility and a measure that evaluates style similarity. Style similarity is measured based on our previous work. **(CLICK)** In this paper, we introduce a new measure that estimates functional compatibility between shapes. Now we describe how this compatibility measure is formulated and learned.

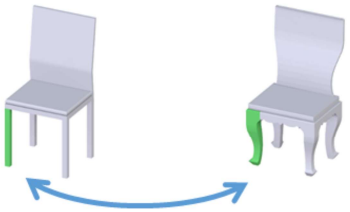


We first define the compatibility measure between elements on structurally different shapes. We use this metric to find the most compatible parts to do part substitution, such as the legs on the table and chair. We also use this element compatibility measure to define overall shape compatibility, as I will discuss later.



The compatibility measure between elements relies on their context and gross shape. **(CLICK)** We encode each element's functional and contextual properties using a graph. The nodes of the graph are different elements and the edges encode adjacency, containment, and symmetry relationships between elements.

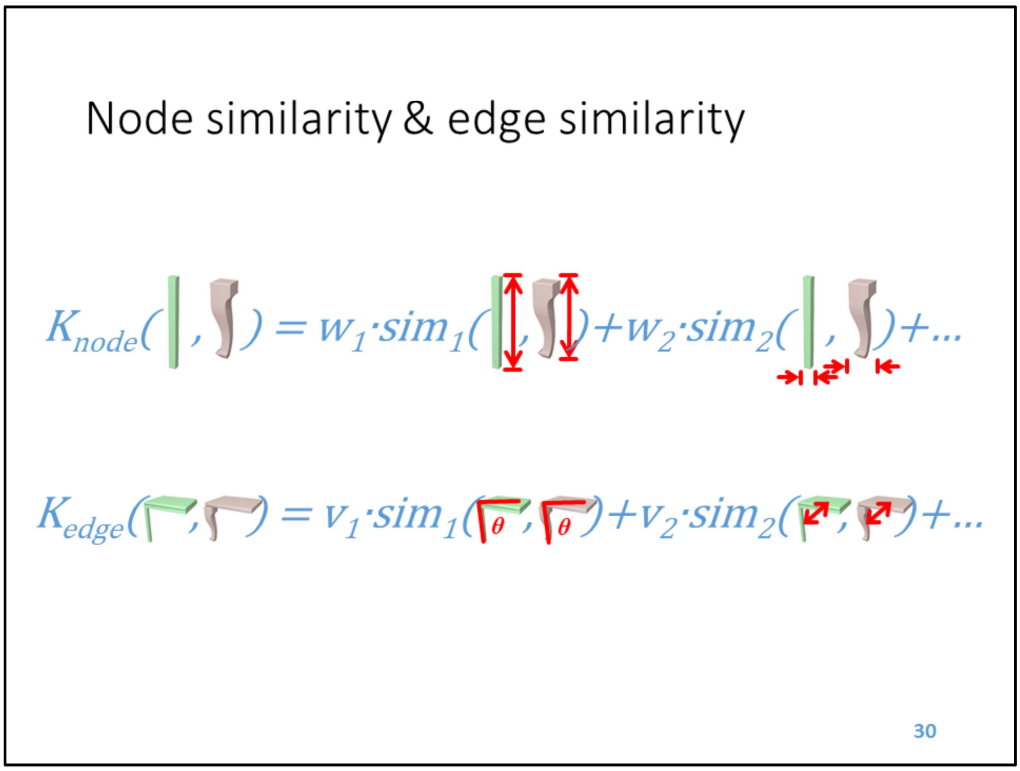
Recursive definition for node compatibility



$$K^n(\,,\,) = K_{node}(\,,\,) \cdot \sum_{edge} K_{edge}(\,,\,) \cdot K^{n-1}(\,,\,)$$

29

The element compatibility can be recursively evaluated using graph walks. The graph walks evaluate node and edge similarity.



(pointer) The node similarity is a kernel function comparing node descriptors capturing gross shape such as the height and width of the element. **(CLICK)** Similarly, the edge similarity captures context in the form of edge descriptors such as angles between their dominant orientations, the centroid distance between elements and so on. The impact of all the descriptors is weighted by the parameters w1, w2, v1, v2 and so on, as you see here

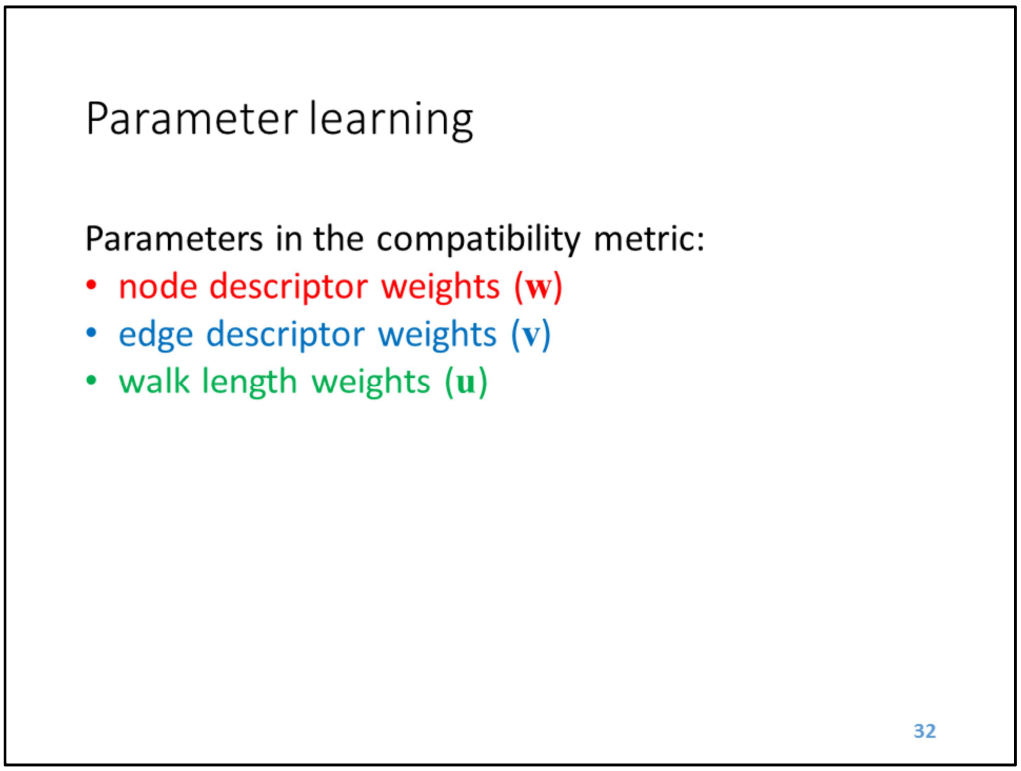
Element compatibility

$$K_{func}(\,,\,) = u_0 \cdot K^0(\,,\,) + u_1 \cdot K^1(\,,\,) + \ldots$$

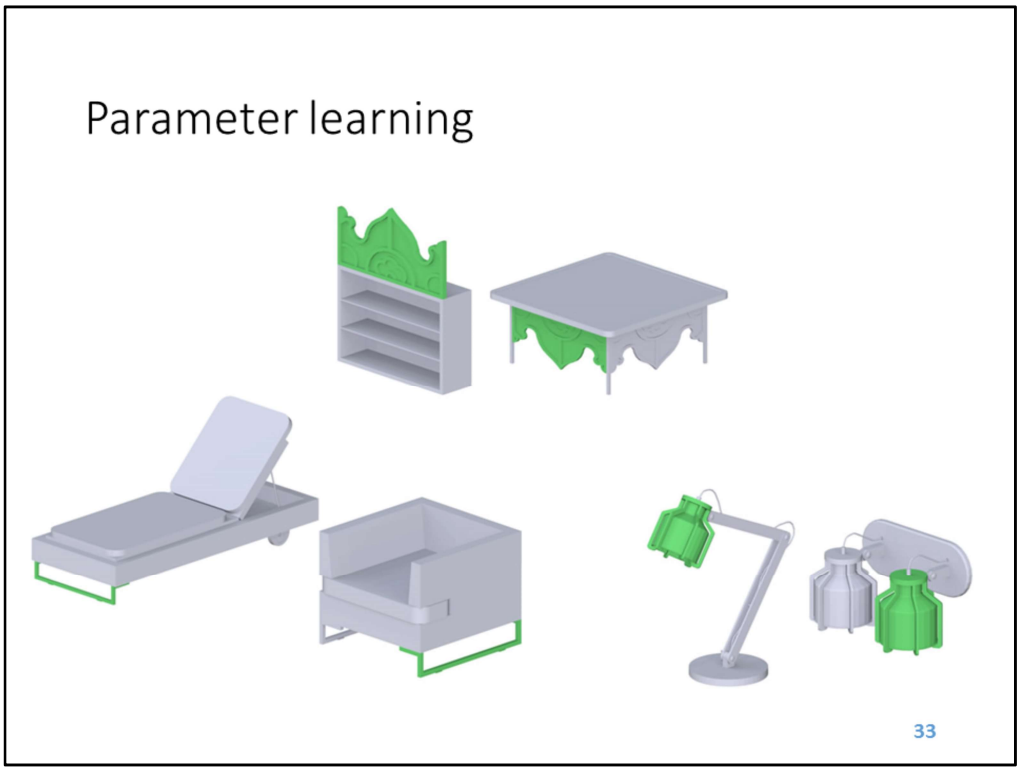
$$D_{func}(\,,\,) = \sqrt{K_{func}(\,,\,) + K_{func}(\,,\,) - 2K_{func}(\,,\,)}$$

31

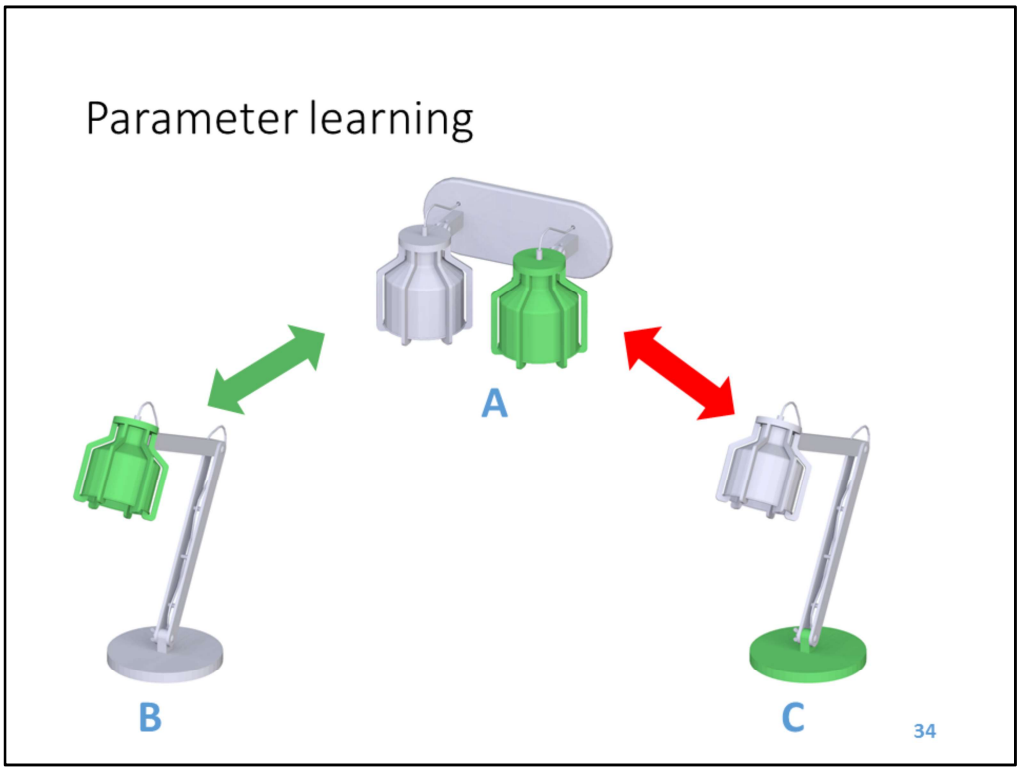
To leverage the impact of different graph walk lengths on the compatibility metric, the compatibility between elements is defined as a weighted combination of kernel similarities defined with different length of graph walks.
(CLICK) The resulting positive definite kernel similarity can then be converted to a distance measure by normalizing according to self similarity.



All the parameters of the element compatibility measure are automatically learned from training data.



The training data are automatically generated from scenes containing coordinated, same style shapes downloaded from online shape databases. From those coordinated scenes, we extract elements approximately identical up to an affine transformation. By construction these elements are compatible since they can be clearly substituted without affecting shape functionality.

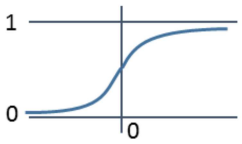


Given these training pairs, the goal of parameter learning is to compute the set of parameters with which our compatibility function will, on average, deem the pair A and B, more compatible than the pair A and C where C is a randomly selected element.

Relative comparisons

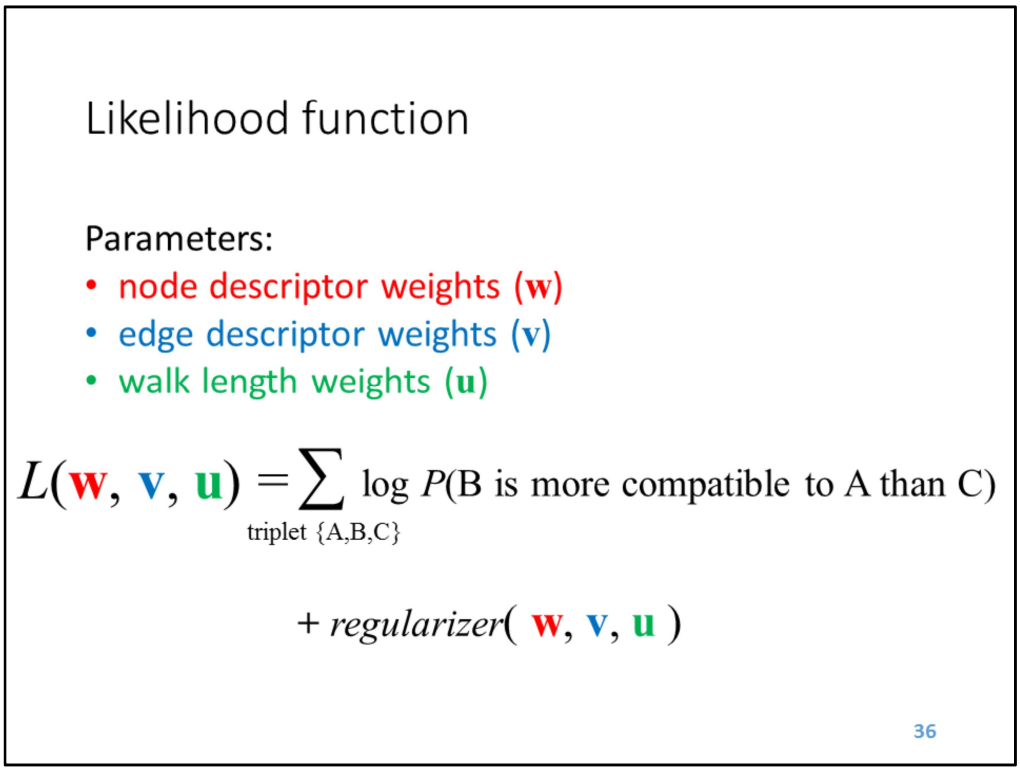
$$P(\text{B is more compatible to A than C}) = \sigma\left(D_{func}(\text{C, A}) - D_{func}(\text{B, A}) \right)$$

σ: sigmoid function

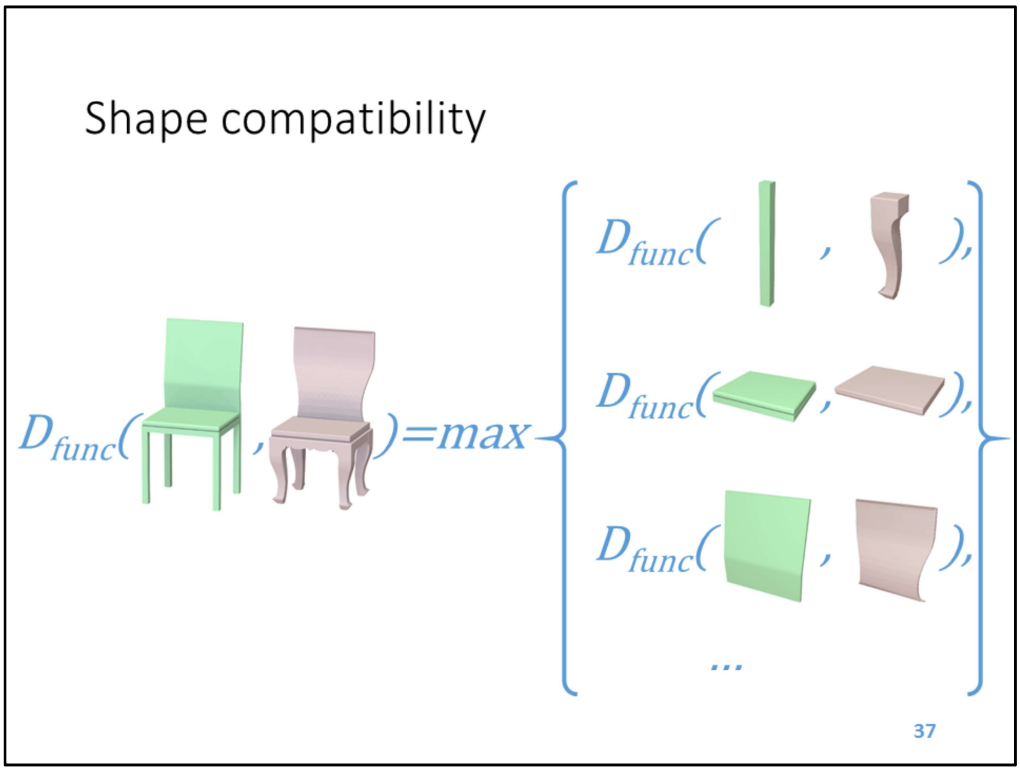


35

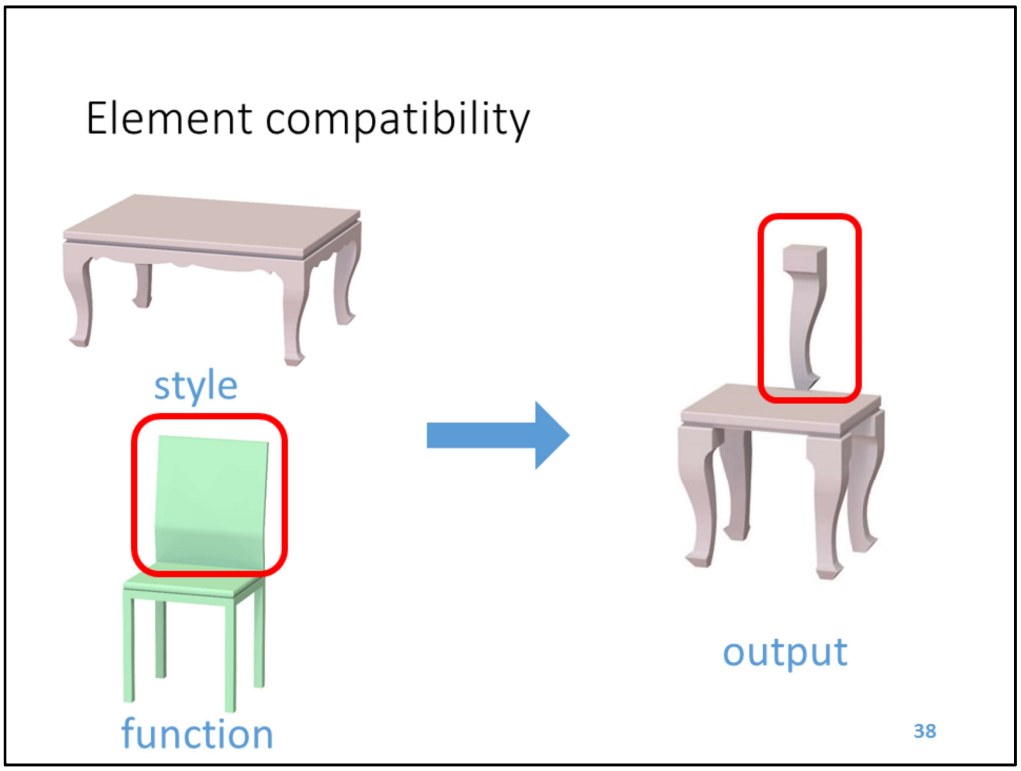
To handle such relative comparisons, we use a probabilistic framework. The compatibility difference is converted into a probability using the sigmoid function.



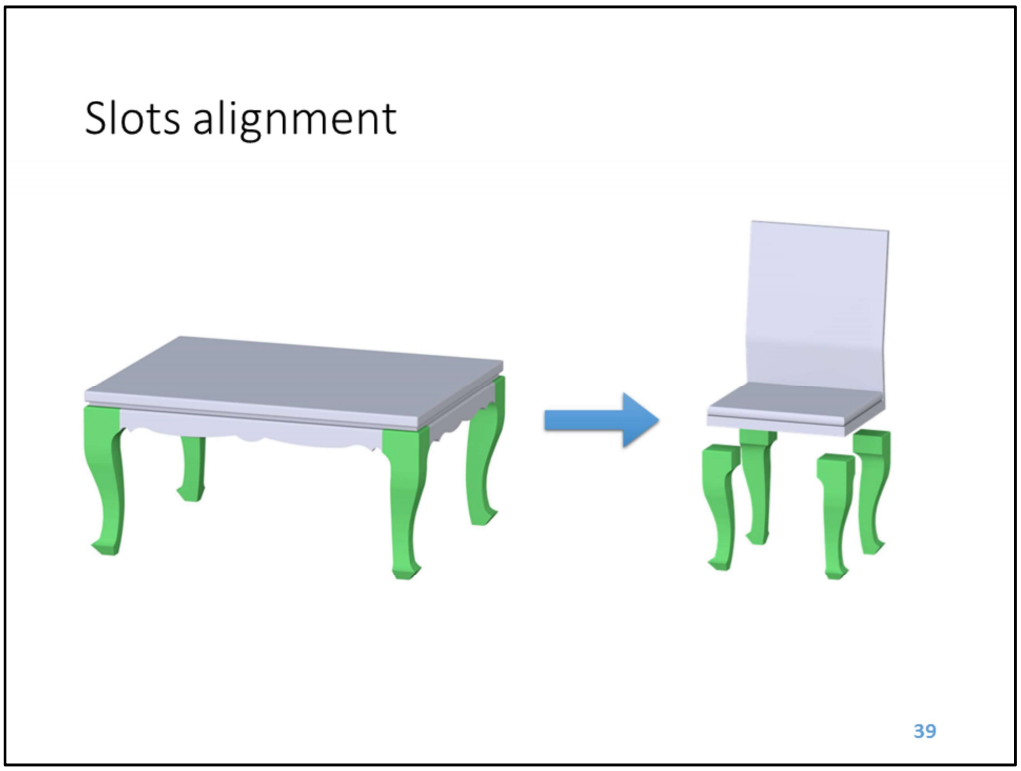
The goal of the learning procedure is to maximize a likelihood function describing the probability of training data with respect to the unknown parameters. **(CLICK)** We also include an L-1 regularization term to promote sparsity in the learned weights.



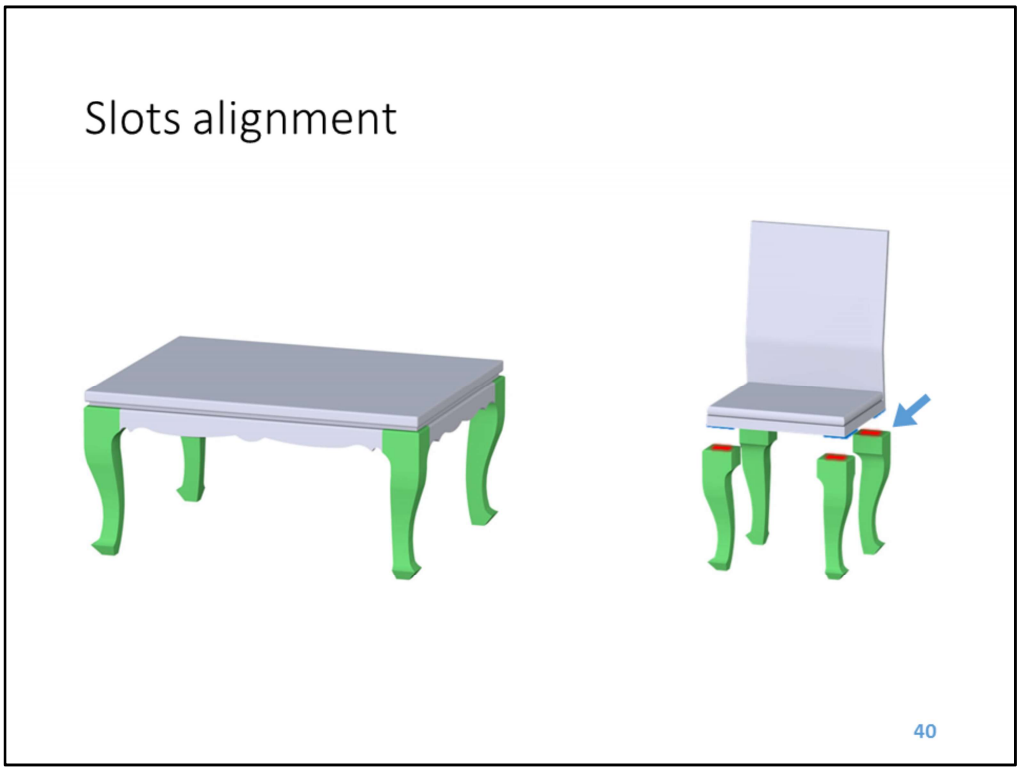
So far we have defined the compatibility measure between elements. The shape compatibility measure can be defined as the maximal compatibility distance between corresponding elements on the two shapes, representing the worst-case influence of an element operation on shape compatibility:



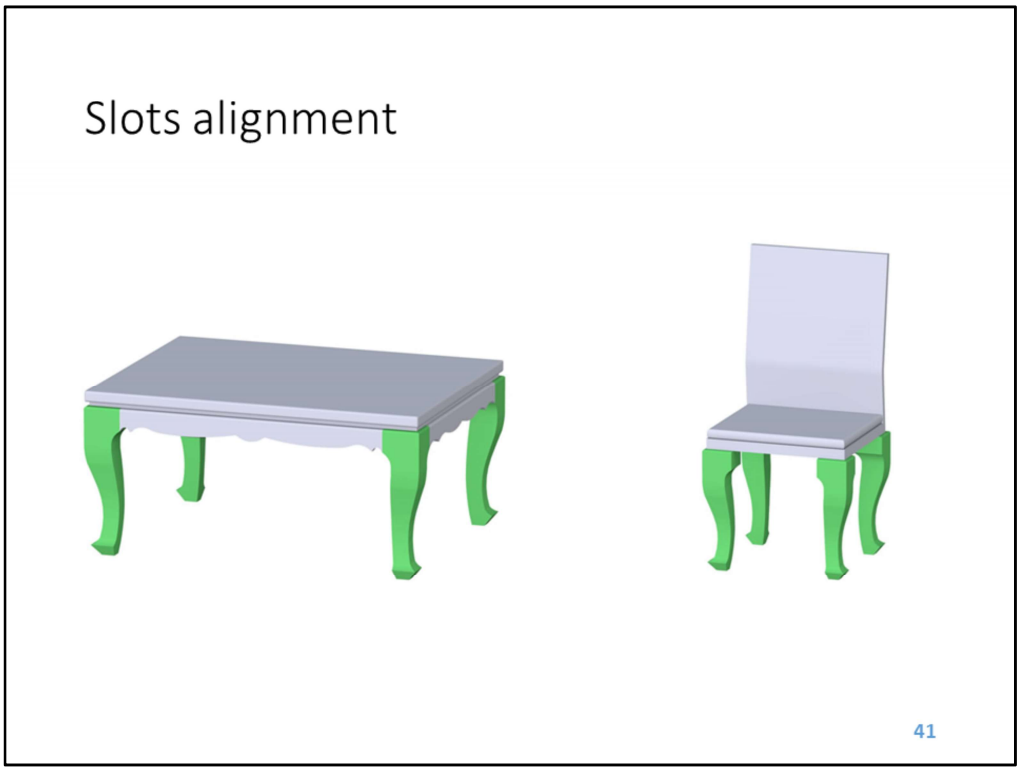
This shape compatibility measure prevents us from performing operations that largely violate the functionality of the target shape.



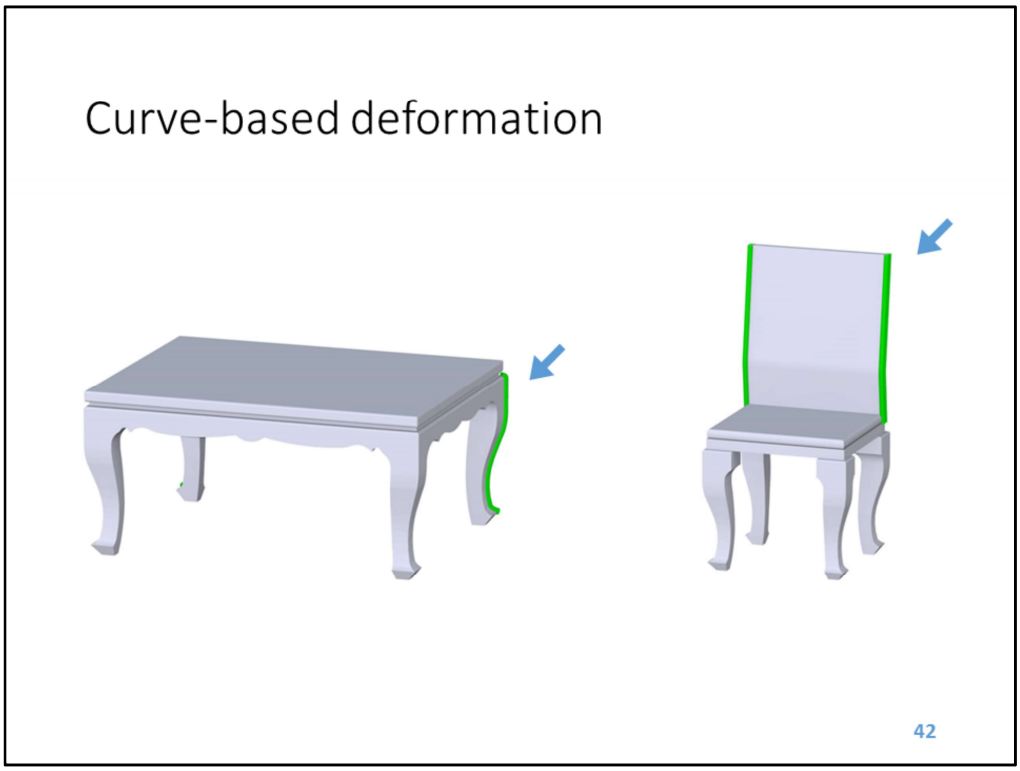
All the operations executed during our optimization loop are designed to preserve shape connectivity.



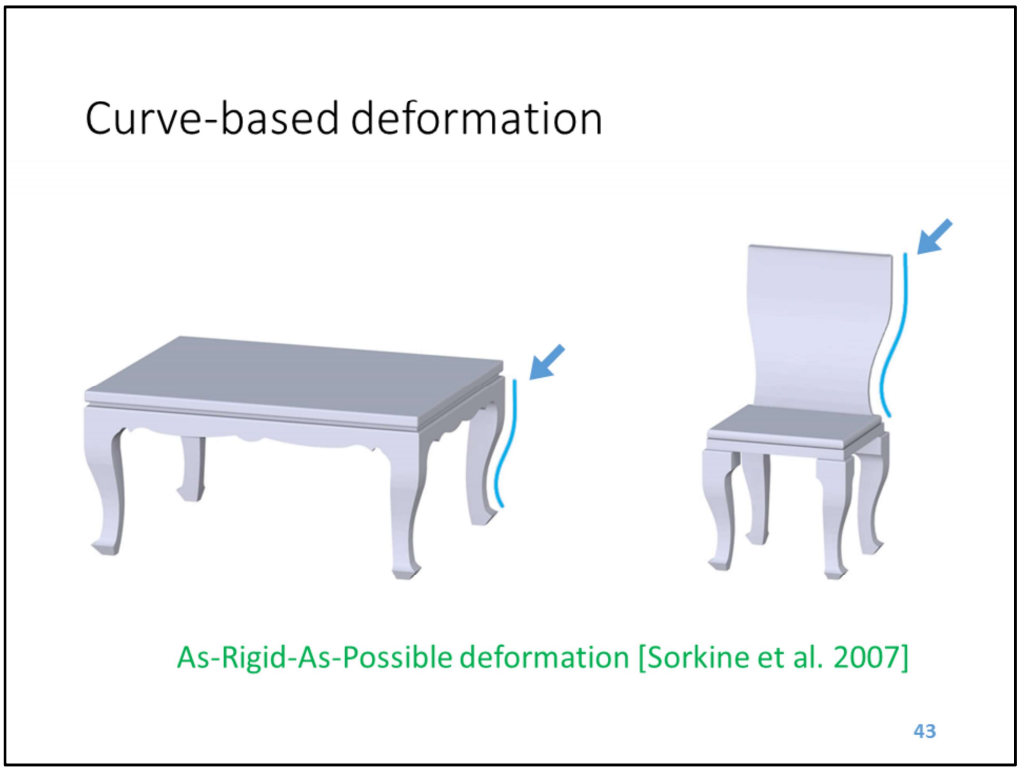
Specifically, substitution and addition operations require alignment of slots, or attachment areas between the newly added-in and pre-existing elements.



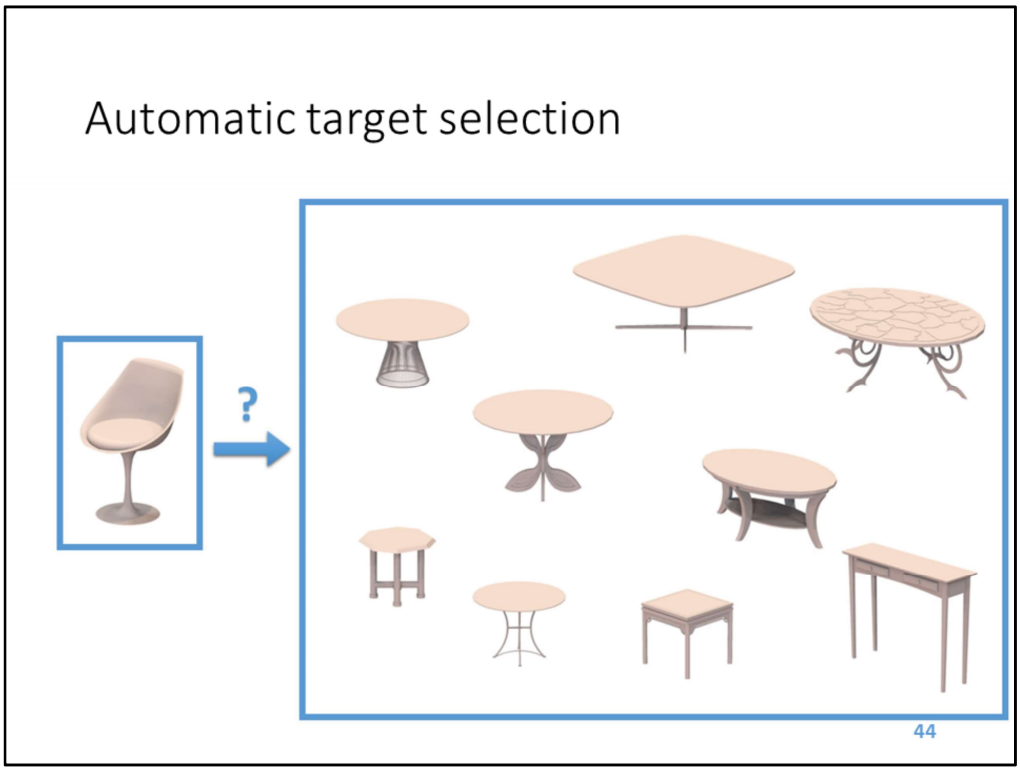
The alignment step seeks to both preserve output functionality and minimally change element shape



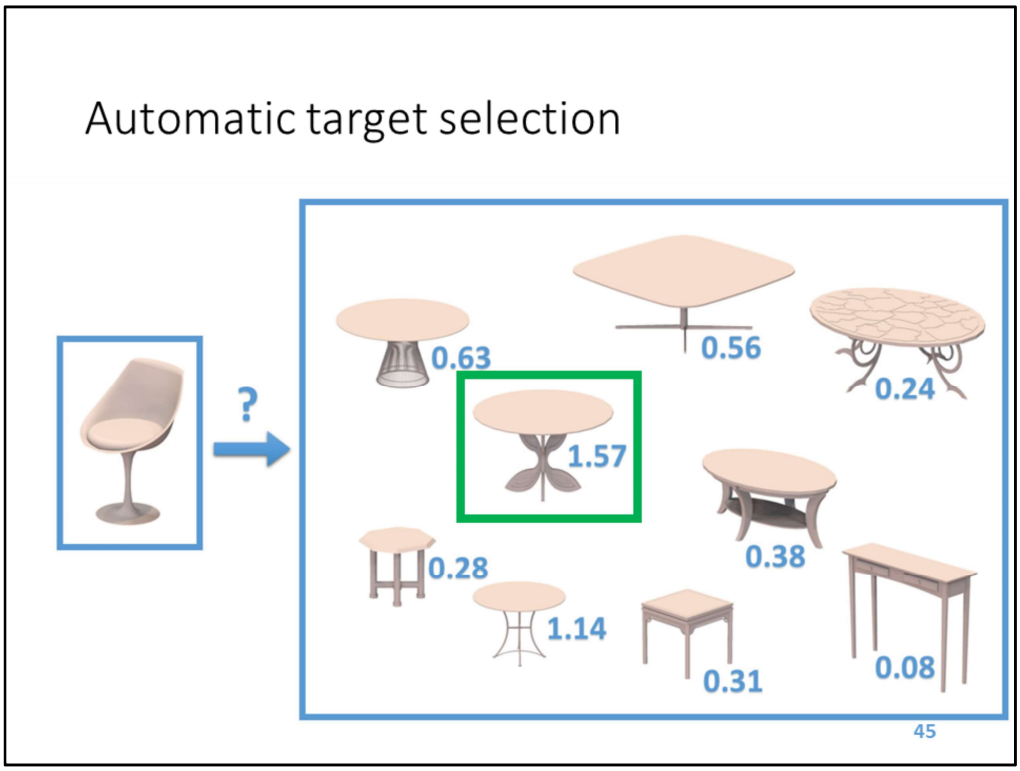
Curve-based element deformation operations modify the feature and contour curves on the output model to be more similar to their exemplar counterparts.



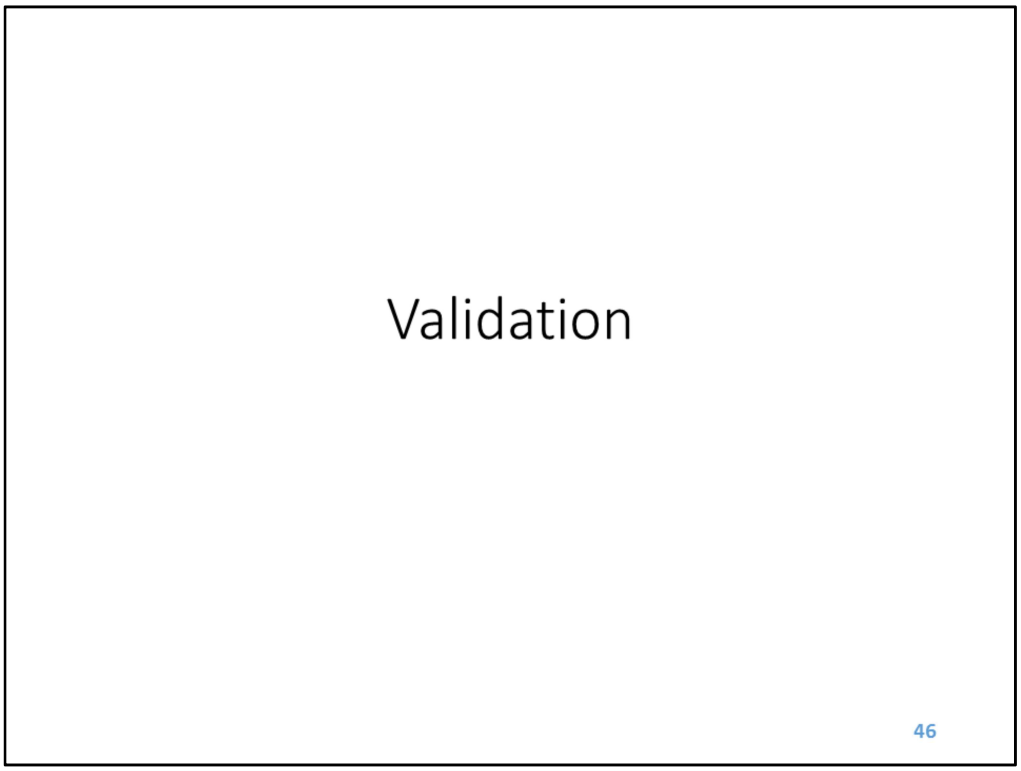
We perform As-Rigid-As-Possible volumetric deformation to transfer curves.



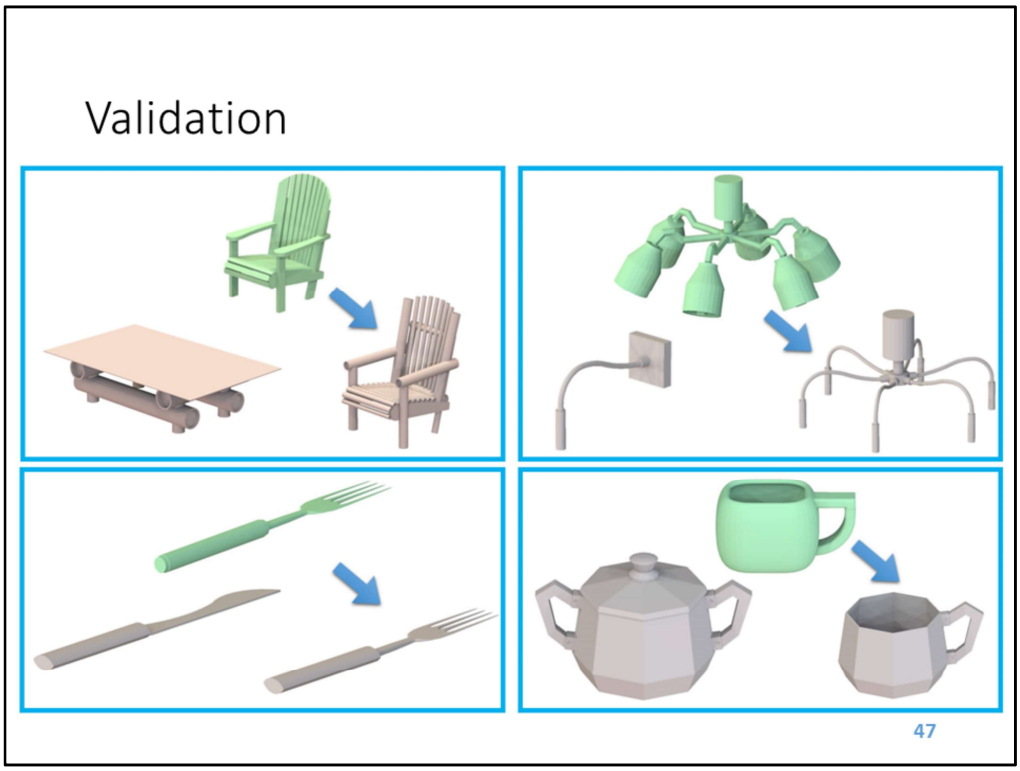
The degree of style similarity between the output and the exemplar is affected by the structure of the original target model. Users may provide a specific target shape for style transfer



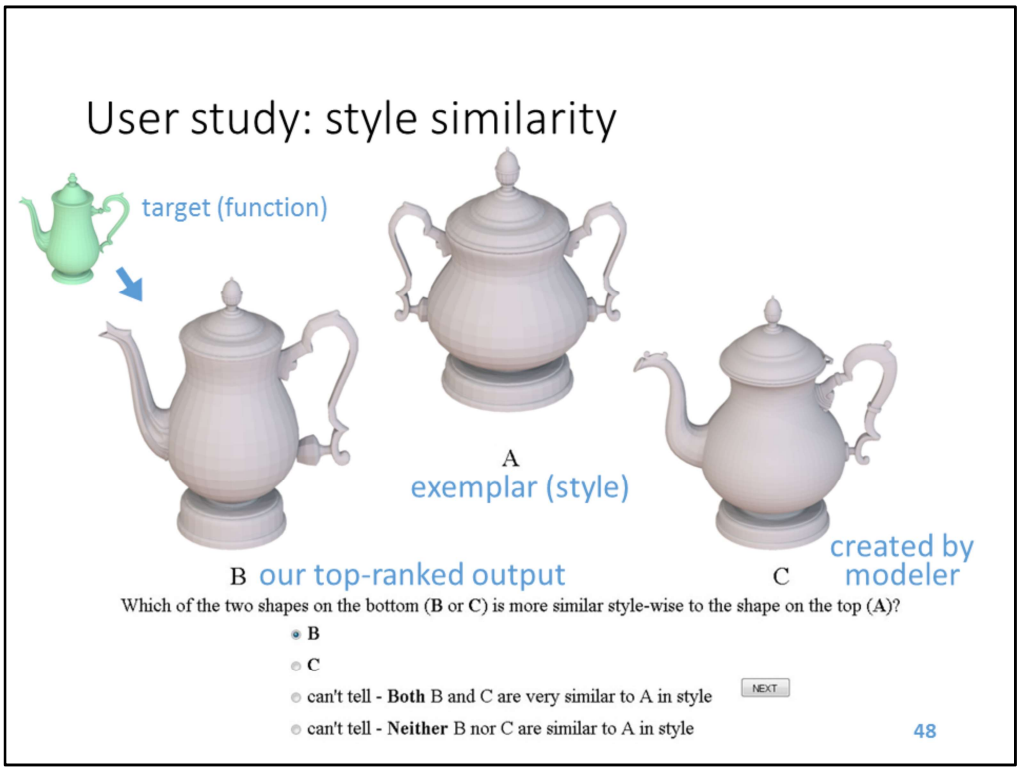
or have our learned compatibility metric to automatically return the most compatible target shape given a database of shapes in a specified class.



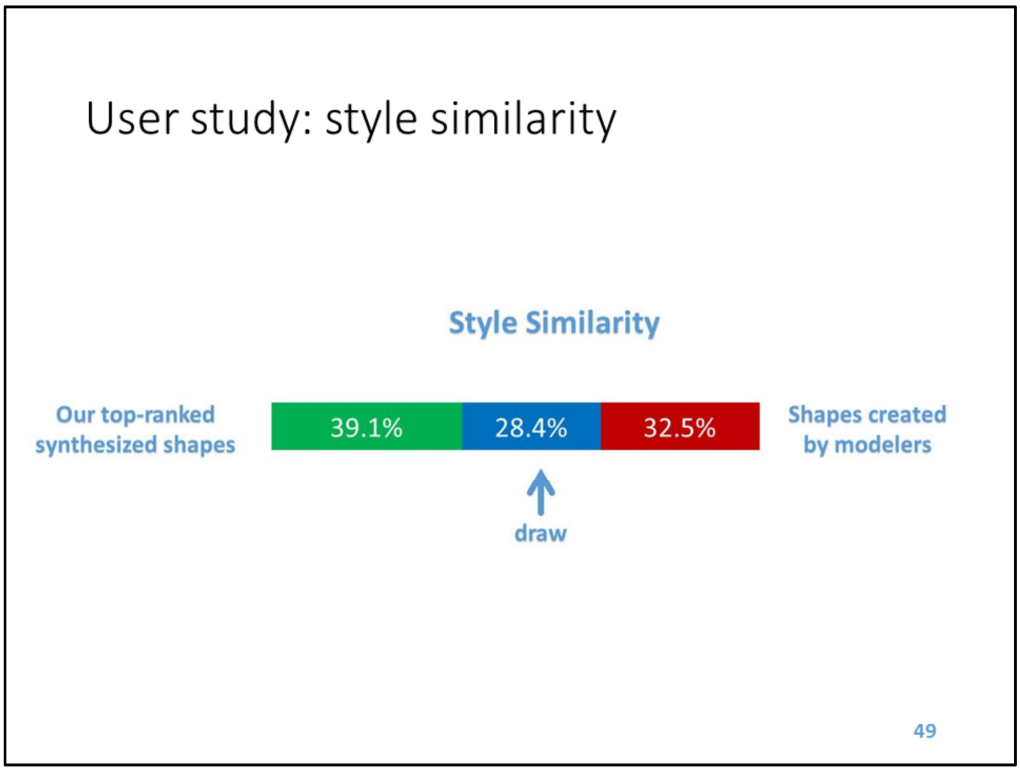
Let’s now discuss the validation of our method.



We evaluate our method by synthesizing more than a hundred new shapes covering four broad categories of everyday objects: furniture, lamps, cutlery, coffee and tea sets. We validate the key properties of our method via several user studies.



The first user study evaluates style similarity between our output shapes and the exemplars. We asked participants to compare style similarity between the exemplar model **(CLICK)** and our top-ranked output **(CLICK)**, against other alternatives, **(CLICK)** such as the shapes found in the same style-coordinated scenes, which are manually created by modelers and should be perceived as stylistically similar to the exemplar shape. Note that our output is randomly placed as option B or as option C so that participants are not biased.



Participants perceived our synthesized shapes as at least as similar style-wise to the exemplars as these manually created shapes.

User study: functionality

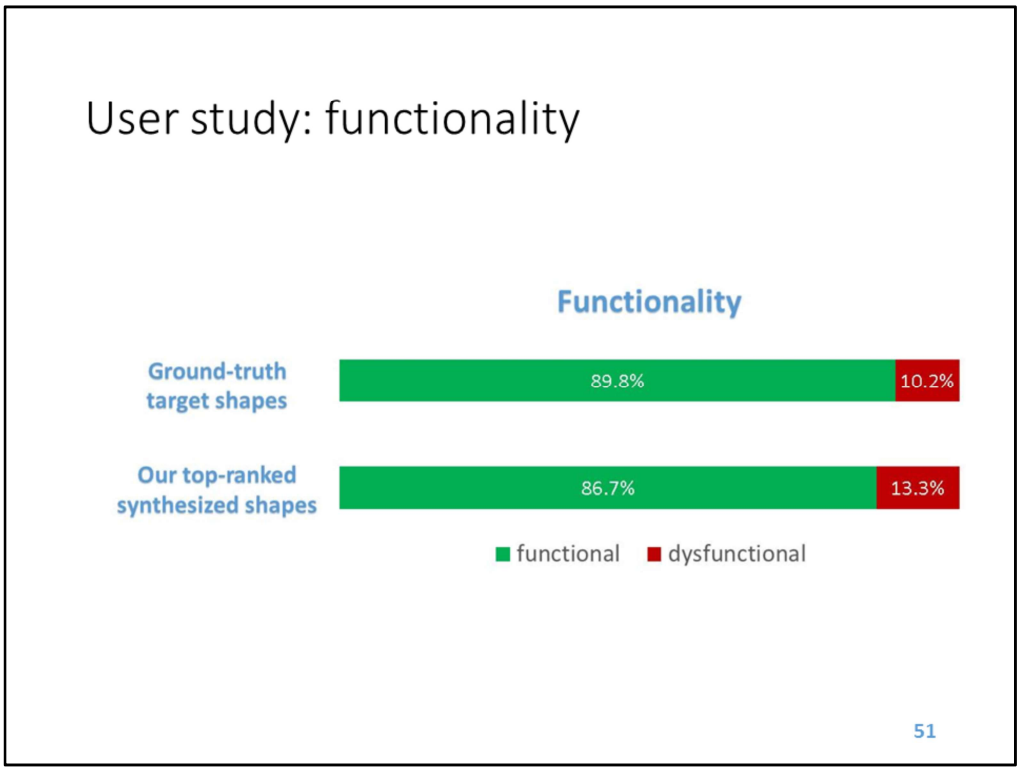
Is this a **functional table lamp** ?

- (•) **Yes**
- ( ) **No**

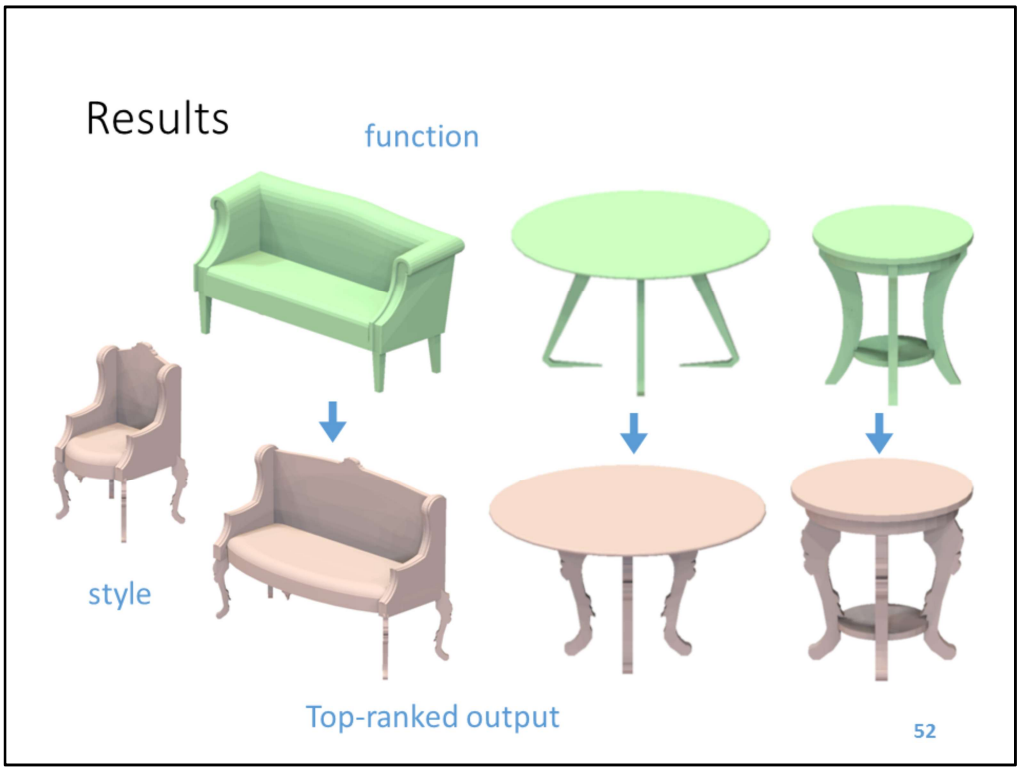
NEXT

50

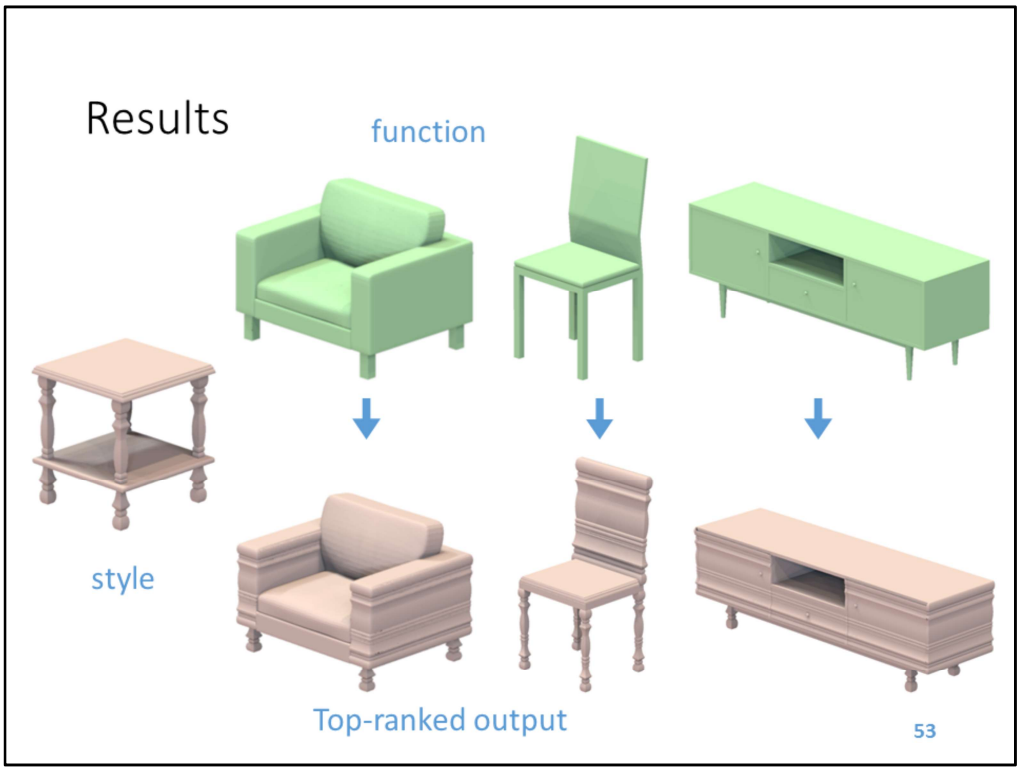
The second study evaluated the functionality of the output shapes, against other alternatives, such as the original ground-truth target shapes



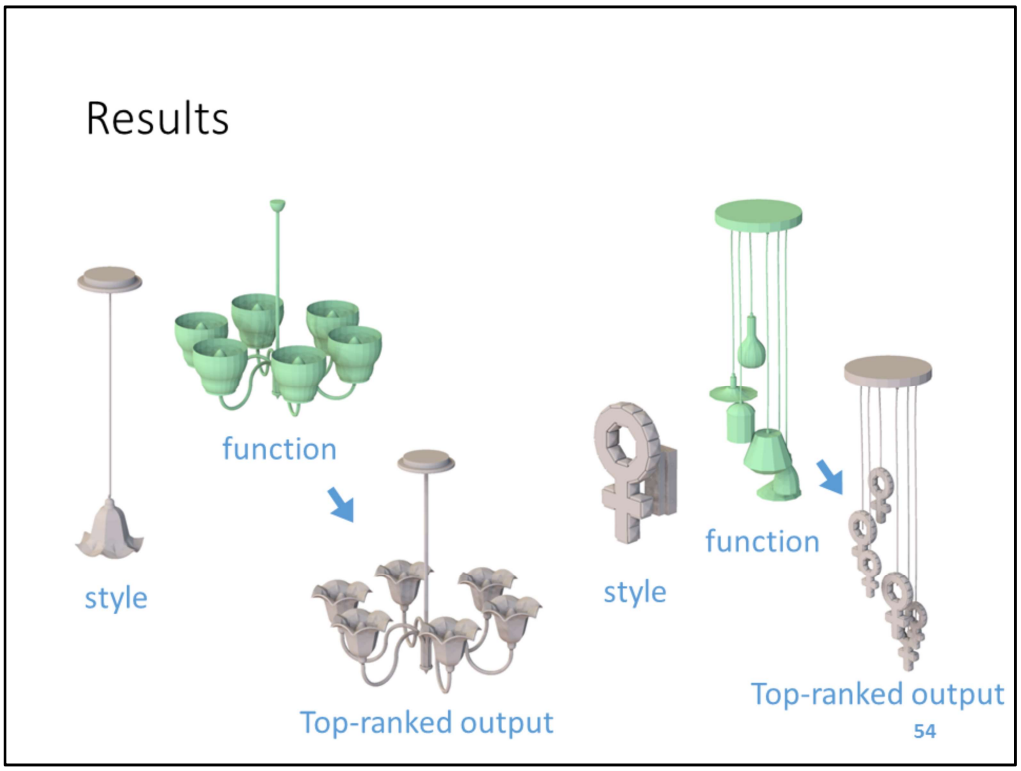
Our outputs are deemed functional at nearly the same rate as the ground truth shapes.



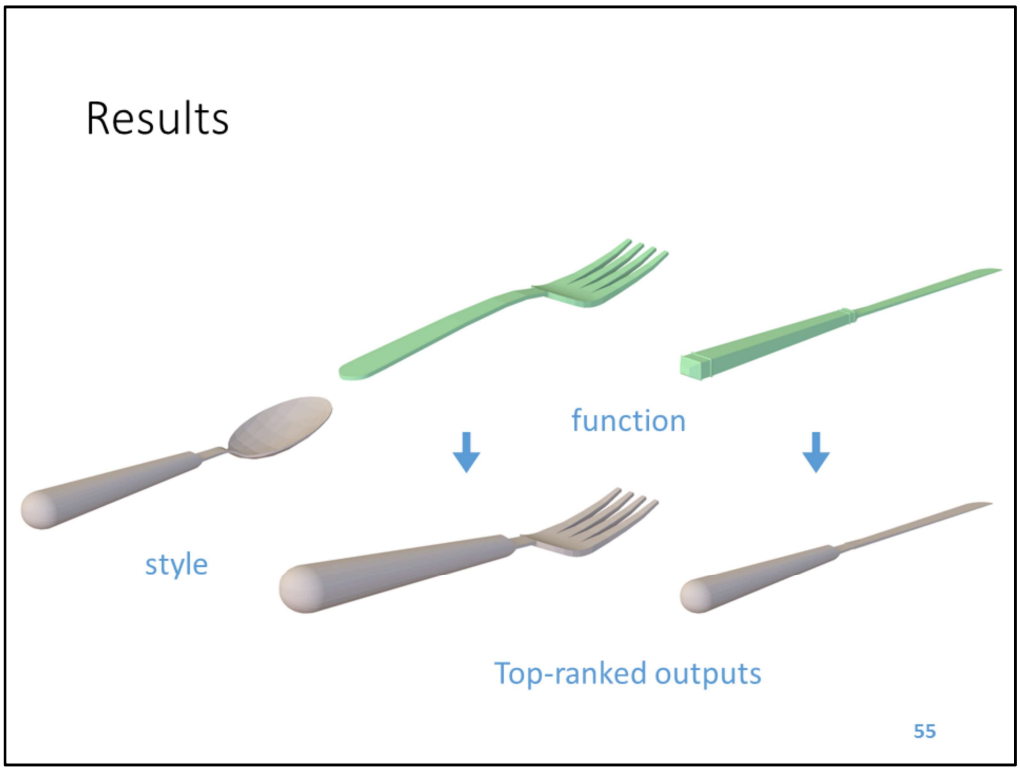
We now show more style transfer results based on our method for furniture. Here the stylistic legs, arm rests and seat back of the chair are transferred to the wide sofa. **(CLICK)** Here our algorithm successfully identified legs as functional compatible elements for style transfer.



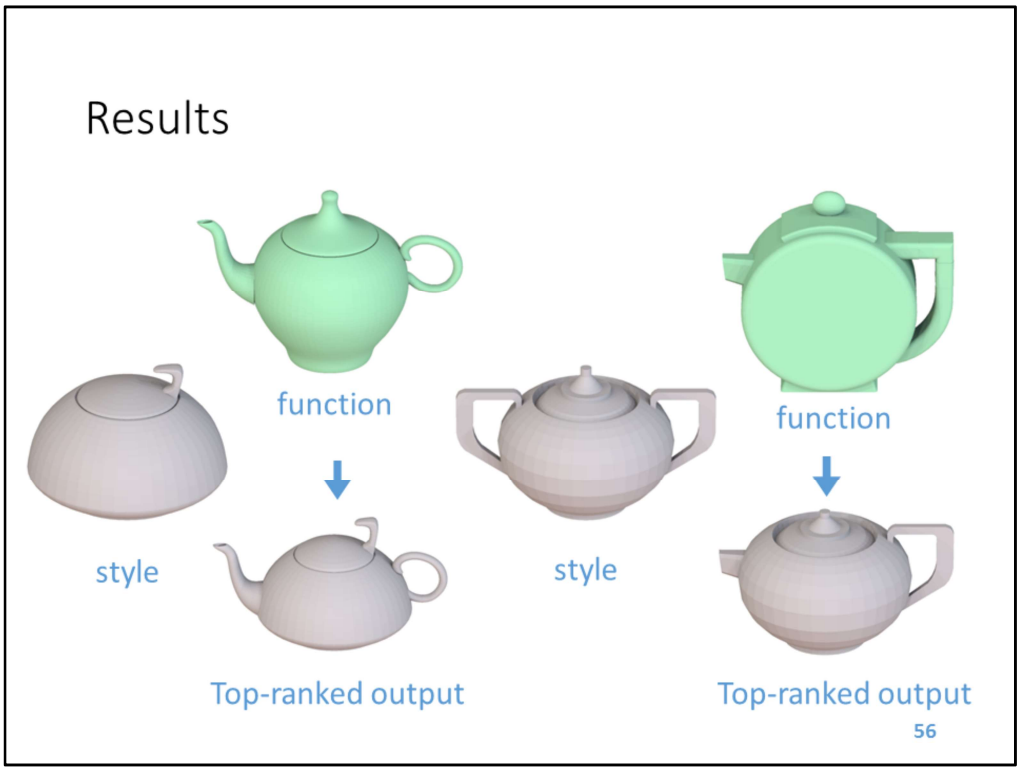
Here are the results involving part substitution and curve-based deformation. Here, legs are substituted, while curve deformation is used to carve the sofa armrests, chair back, and cabinet body according to exemplar feature curves.



Here we show results for lamps. Element substitutions are used to transfer the exemplar style.



Here are results for cutlery, again done through element substitutions.



and results for coffee sets – please see our paper, supplementary material and video for more results.

Summary

- First algorithm for synthesizing shapes by transferring style between objects with different structure and functionality
- Key component: learned measure for assessing element compatibility
- Demonstrated to generate functional, plausible, similar-style models

57

To summarize, we have described the first algorithm for synthesizing shapes by transferring style between man-made objects with different structure and functionality. The key component of our algorithm is a learned metric designed to assess element compatibility across shapes with different structure and function. Our method is demonstrated to successfully generate functional, plausible, similar-style models in a wide range of shape classes.

Thank you!

Acknowledgements: *Nicholas Vining, Mikhail Bessmeltsev*

Our project web page:

http://people.cs.umass.edu/~zlun/papers/StyleTransfer

58

Thank you for your attention! We show here our project page that includes source code and results.