Shape Analogies via Group Shape Difference Analysis

Li Yi
Stanford University
ericyi@stanford.edu

Abstract

In this project, we proposed a novel framework to solve the shape analogies problem between 3D shape collections. Instead of using geometric features based on individuals to describe each shape, we leveraged the idea that fine grained characteristic of a shape can be defined by other similar shapes, and used relationship within collections to depict shapes. To capture the abstract relationship, a novel shape descriptor, average shape difference, was proposed. Based on the descriptor, we formulated the shape analogies problem as a graph matching task and solved it via a coarse to fine algorithm. We tested our algorithm on several datasets and the result shows our framework effectively solves the shape analogies problem without knowing the cross-collection point correspondence information, which is a key factor for traditional shape matching techniques but usually not easy to get.

1 Introduction

Analogy is a cognitive process of transferring information or properties from one object to another. This is a central part in human cognition process. While looking at two objects, human can employ analogy easily and usually unconciously to transfer abstract information, e.g., style, structure, between the objects. These objects themselves can be some relationship. We often refer to such kind of analogy as A is to B what C is to D. In other words, we want to find an analogous object A for B so that how A relates to B is similar to how C relates to D.

Image analogies have gained attention for providing a very natural means of specifying relationship between images [2], which motivates many applications, e,g., super-resolution, texture transfer, image colorization and artistic filters. Shape analogies, however, haven't been well studied yet. Shape analogies, like image analogies, have the potential to specify abstract relationship among shapes, which will contribute to many other applications like shape classification, shape searching, semantic knowledge transfer, shape style transfer. The idea that relationship can be defined through analogies motivates us to conduct shape analogies in the shape analysis context.

In many settings, we may desire relationship among a shape collection instead of pairwise relationship. This becomes more and more common as large public repositories of 3D shapes continue to grow. Thus it is required to consider analogies in shape collections. For example, we may want to find the analogy that how shape A relates to a shape collection is similar to how shape B relates to another shape collection. By conducting analogies in a collection setting, we are actually defining a shape by its neighbors. This is reasonable since the fine grained characteristic of an object can be obtained by comparing it with the other similar objects. Shape analogy is worth studying since it provides a way to specify the variation in a collection and can thus be used in many areas like shape retrieval, shape manipulation, shape classification.

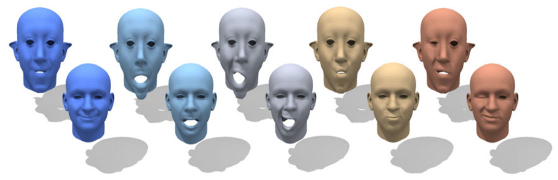

Figure 1: Shape analogies between two facial expression collections

We aim to make analogies between different shape collections. This is a very challenging task since usually the things we need to capture via shape analogies are hard to describe by computers. For example, we may want to make analogies between two facial expression collections of different people, as is shown in Figure 1. For each facial expression in the first collection, we want to find the corresponding expression in the second collection. How to properly capture the "expression" here is not an easy problem. We may use some geometric features. However, the lack of correspondences between different shape collections, which is usually the case, makes it hard to compare or transfer geometric features across collections.

In our framework, instead of comparing geometric features across different collections, we choose to describe the shape relationship in each collection and compare these relationships across different collections. By doing so, we are able to "align" different shape collections at once, which means for all the shapes from one collection we are able to simultaneously find the analogizing shapes in the other collection. We formulate the shape analogies task as a graph matching problem. Our work focuses on how to describe the relationship within a collection and use a weighted graph to represent such information. After getting the graph for each collection, we match graphs using spectral matching. Our framework makes full use of the structure of each collection and avoids comparing geometric features directly across different collections, powerful for the usual condition when it is hard to compare or transfer geometric features across collections.

2 Related Work

Shape analogy is performed between different objects. These objects can be shapes or the relationship between shapes. Previous shape analogy work mainly focuses on analogy between shapes, which includes a lot of topics like global registration, shape matching, complete or partial correspondence calculation. Our focus lies in analogy between the relationship of shapes. This is similar to shape matching in a way that we are trying to find similar shapes. But our work differs from traditional shape matching in how to define the similarity metric and the descriptor.

Traditional shape matching owns the prototype of solving nearest search problem in a shape descriptor space under certain similarity measure. Different shape descriptors have been proposed to capture the geometric or topologic information of 3D shapes and can be categorized according to the representation of shape descriptors as feature based, graph based, histogram or distribution based and 3D object recognition based. We will refer the reader to [7] for a detailed review of different shape descriptors.

A similarity measure (or a dissimilarity measure) can be formalized by a function defined on pairs of descriptors indicating the degree of their resemblance. From our point view, most of existing metrics suffer from certain drawbacks. First of all, the notion of shape difference is not made explicit at best only a shape distance is defined. With that it is impossible to understand precisely where the variation happens on a shape, as each shape is treated as atom typically, a point in a fixed-dimensional Euclidean space. Furthermore, it is hard to compare differences between shapes to express differences among the shape differences, for the same reason. Second, large amounts of information about the shapes is ignored, and this can affect the results. For example, the connectivity of the landmarks can be just as important as their absolute positions. Third, most of existing metrics are used to support discriminating shapes under large-scale variations (e.g., cars from humans), not capable of handling fine variability, which has been popular in the computer vision community in the last few years (see, e.g., [1] and the references in the papers therein).

[5] develops a novel formulation for the notion of shape differences, which provides an easy way to help us compare two shapes intrinsicly. Their proposed difference operator, derived from a shape map, makes shape difference itself becomes an object which could be compared and manipulated. Also their approach is based on a linear algebraic framework and makes it possible to use many common linear algebra tools for studying a matrix representation of the operator. In their work, they tried to conduct shape analogies in a collection setting without knowing cross shape correspondences between two collections. However, their approach fails rapidly when the collection size increases. The increase of collection size can actually provides more information about the characteristic of a shape, since more comparision can be conducted within the collection and thus defines shape properties in a clearer way. We hope to leverage this property to solve shape analogies with large shape collections.

Another similar work described in [6], which makes use of the idea of using analogy to specify the deformation of objects, successfully achieves deformation transfer for triangle meshes. In their work, analogies are made to synthesize new shapes. Basically the input is one source shape collection and a target shape model. The output of their algorithm is the target shape collection "aligned" with the source shape collection. The "align" here means finding the analogies of the source shapes for target shapes. However the work in [6] requires cross-collection point correspondence information to transfer geometric properties so that the analogy can be made, which is indeed a limitation of the work.

In our work, we are doing analogy in a more challenging condition without the cross-collection point correspondence information. Our goal is a bit different from [6]. We are not considering synthesizing a target shape collection, but given both source and target shape collections, we would like to "align" the two collections. The lack of correspondence information makes it difficult for the approach in [6] to succeed. Even if it is possible to estimate the cross-collection point correspondence, limited by the estimation accuracy, the approach is still not able to obtain satisfactory results. We follow the work in [5] and propose to explore the relationship among shapes to achieve our goal. We expand the shape difference descriptor designed in [5] to better capture the relationship information within a collection. Different experiments show the effectiveness of our proposed descriptor. Making use of the descriptor, we are able to simultaneously "align" one collection to another by a coarse to fine iterateable algorithm. The impressive results we obtained prove the power of our machinery. Our algorithm works great without the cross-collection correspondence, since our approach doesn't directly compare geometric features between collections but the abstract within-collection relationship. We do analogy to capture the common structures shared by different collections. Our approach provides a novel view to conduct cross collection shape analysis by using shape relationship as an alternative to geometric correspondence information, which is usually not available.

3 Technical Part

Shape analogies between different collections without cross correspondences are very challenging. In the most general case, it is even difficult for human being to successfuly make such analogies. Thus, we choose to regularize our problem setting. We will consider two similar collections as the target to conduct shape analogies. "Similar" here has two meanings. One is that the two collections are basically the same category, since there is no reason to transfer chair shape difference to human facial expression shapes. The other is that the two collections contain the same structure. Take the facial expression shape collections as an example, two collections from different people should contain the same set of expressions. Another condition is that we know the pointwise correspondences among shapes within each collection. This makes it easy for us to compare shapes within each collection. Therefore, in our setting, each shape in the first collection has one ground truth corresponding shape in the second collection, and vice versa. Our goal is to find out all the shape correspondences between two collections.

The challenges in this problem lie in following aspects. Firstly, it is difficult to compare geometric features between collections due to lack of cross collection pointwise correspondence, so we need to find some alternative approach to link the two collections. Secondly, it is unknown how to effectively define a shapes intrinsic characteristic in a collection setting. A collection is more than the sum of individuals so we need to define a shape not only based on itself, but also based on other shapes in the collection.

We managed to deal with these challenges in our proposed framework. We first expand the shape difference descriptor and propose to use a combination of both pairwise shape difference and average shape difference in a collection setting. The average shape difference is designed to capture the intrinsic characteristics of a shape in a collection. We use two experiments to show the effectiveness of this new descriptor. Then secondly, we depict intrinsic relationship in each collection with a weighted graph and formulate the shape analogies task as a graph matching problem. The graph vertexes denote different shapes in the collection and the edge weights are generated using the shape descriptors described in the first step. Finally a coarse to fine iterateabe algorithm is proposed to simultaneously analogize all the shapes from one collection to another based on spectral matching. Our main technical focus lies on three parts: describe shapes and the relationship of shapes within a collection, form weighted graphs for a collecion to synthesize all descriptors, iteratively refine the results by alternating between matching two graphs and getting better weighted graph for each collection. We will introduce these in the following part of this report.

3.1 Expanded Shape Difference Descriptor

According to [5], shape difference can be defined as is shown in Figure 2. Given two shapes M, N, endowed with inner products h^M and h^N respectively, and a functional map $F : L^2(M) \to L^2(N)$, there exists a unique linear operator $D_{h^M,h^N} : L^2(M) \to L^2(M)$ satisfying:

$$h^M(f, D_{h^M,h^N}(g)) = h^N(F(f), F(g)), \forall f, g.$$

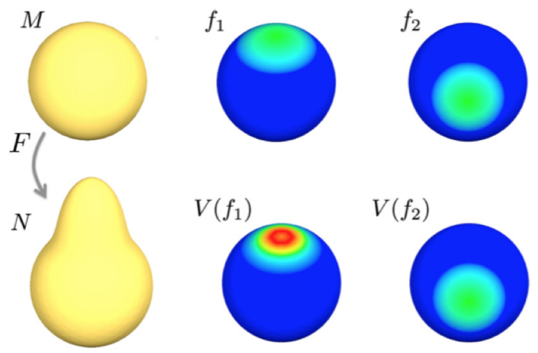

Figure 2: An explanation about the concept of shape difference. Here V denotes an linear operator which is just the special case of D when the inner products defined on M, N are all area based inner products.

Here we use $L^2(\cdot)$ to denote a set of square integrable real-valued functions on a surface. And we will refer to the operator D_{h^M,h^N} as the difference between h^M and h^N. It can be seen that the linear operator D modifies g so as to exactly compensate for the distortions introduced by the map F. It is explained in [5] that D is a universal compensator a single such operator works simultaneously for all functions f and g. Stated differently, D depends only on the given inner products on M and N, and the functional map F . It is also important to note that D is a linear self-map of the space of functions over M. And in the discrete setting, D can be denoted as a matrix.

Different kinds of inner product can actually induce different D according to the definition above. Several kinds of inner products are proposed in [5], including area based inner product h_a and conformal inner product h_c. For simplicity, we will not discriminate different kinds of inner product and simply use an operator $D_{M,N}$ to denote the shape difference between shape M and N. All our discussions apply to shape difference based on any kind of inner product.

Based on the above shape difference concept, we firstly expanded the pairwise shape difference and proposed a new shape descriptor called average shape difference, to synthesize pairwise shape difference in a shape collection. The average shape difference, combined with pairwise shape difference, can capture what makes a shape different from all the other shapes in the collection. We verified the effectiveness of the average shape difference by visualizing the synthesized average shape difference and seeing whether it agrees with human observation. Also based on the average shape difference, we implemented an interesting application, exaggerating the shapes. The success of this application also shows the average shape difference captures the intrinsic characteristics of a shape as we expected.

In our following discussion, we use $\{A_l\}, l = 1, 2, ..., n$ to denote a shape collection containing n shapes. We use D_{A_i,A_j} to denote the shape difference between A_i and A_j, where $i, j \in \{1, 2, ..., n\}$.

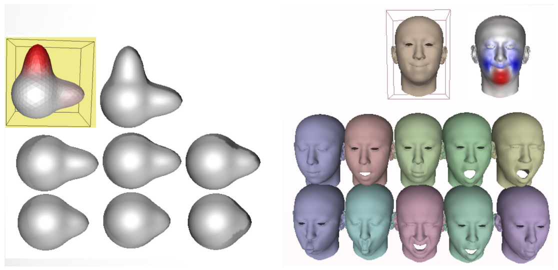

Figure 3: Visualization of average shape difference. In the bumpy sphere collection, average shape difference for the sphere with two bumpies shows large area increase in the top bump area and slight area increase in the side bump area. In the human face collection, target shape has relatively smaller area on the cheek and larger area on the chin.

Given pairwise shape difference D_{A_i,A_j}, we define average shape difference D_{A_i} as

$$D_{A_i} = \frac{1}{n} \sum_{j=1}^{n} D_{A_i,A_j}$$

Average shape difference D_{A_i} reveals the difference from A_i to the mean shape defined in the collection, thus could to some degree reveal what makes shape A_i special. To visualize D_{A_i}, we choose to visualize their effect on different functions. For every vertex v on the shape, we can define a locally supported function f_v, then we use $\frac{||Df_v||_2}{||f_v||_2}$ to denote the effect of D_{A_i} on this vertex. We choose area based shape difference in our visualization, so $\frac{||Df_v||_2}{||f_v||_2}$ reveals how severe the area around v changes after functional mapping. And we use red color to denote area increase, blue color to denote area decrease. Then we can visualize the average shape difference as is shown in Figure 3. The visualization result is reasonable and agrees with human observation.

Then we can make use of the average shape difference to do shape exaggeration in a relatively simple setting. Exaggeration here means making the characteristics of shapes more prominent. Shape difference is a linear operator and it is manipulatable and comparable. This property makes it possible for us to manipulate shape difference and generate shape difference for some new shapes. Exaggerated shape is just one kind.

In our setting, we firstly manipulate shape difference D_{A_i} to get the shape difference $\tilde{D}_{A_i}$ for exaggerated shape and then do nearest search in the whole shape collection to get a shape A_t, whose average shape difference D_{A_t} is the closest to $\tilde{D}_{A_i}$. There are two problems here, how to manipulate the shape difference and how to compare different shape difference. To figure out the first problem, we do eigen decomposition for D_{A_i}. When we use area based inner product, D_{A_i} is symmetric, so we get $D_{A_i} = V^T \Sigma V$. The eigenvalues and eigenvectors of D_{A_i} reveals the property of shape A_i. Each eigenvector represents a certain pattern and the corresponding eigenvalue denotes how severely A_i differs from the other shapes in A_l on this certain pattern. We can exaggerate a shape A_i by adjusting the eigenvalues of D_{A_i}. We use a stretching factor δ to adjust the eigenvalues of D_{A_i} and generate $\tilde{D}_{A_i}$ as below

$$\tilde{D}_{A_i} = V^T(I + \delta(\Sigma - I))V$$

As for the second problem, we simply use $||D_{A_i} - D_{A_t}||_F$ as a metric to compare how similar D_{A_i} and D_{A_t} is. The described procedure is shown in Figure 4. Following the procedure, we can achieve shape exaggeration as is shown in Figure 5.

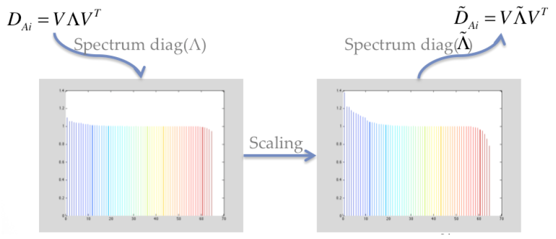
$D_{Ai} = V\Lambda V^T$
Spectrum diag(Λ)
Scaling
Spectrum diag($\tilde{\Lambda}$)
$\tilde{D}_{Ai} = V\tilde{\Lambda}V^T$

Figure 4: The procedure of how shape exaggeration is conducted.

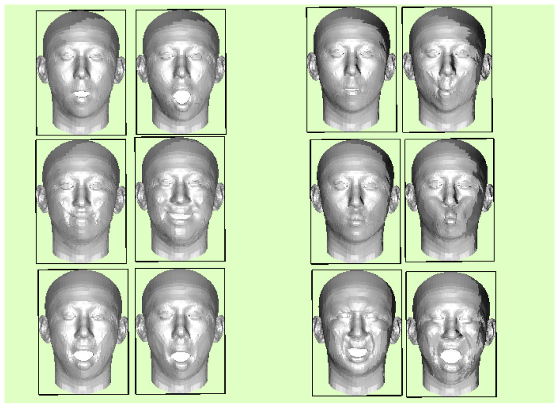

Figure 5: An example of shape exaggeration. In this experiment we have 6 pair of shapes. In each pair of shapes, the left one is the initial shape. We manipulate the average shape difference of the initial shape and do nearest search to get an exaggerated shape, which is the right shape in each pair.

By far, we demonstrate the effectiveness of average shape difference from two different view. One is to visualize the average shape difference directly. The other is to evaluate it in an application. Both shows that average shape difference can capture the characteristics of shapes in a collection as we expected. Then we will make use of both pairwise shape difference and average shape difference to tackle the shape analogy problem.

3.2 Graph Matching Formulation

In this part, we will show how we formulate the shape analogies task as a graph matching problem. Using weighted graph to denote a collection is a very naturaly idea under the assumption that all pairwise relationship can to some degree encode the overall property of a collection. We use graph vertexes to denote different shapes and use the edges between vertexes to denote the relationship between shapes. This representation approach is shown in Figure 6. A complete graph will be used and we need to associate each edge with some proper edge weight so that the structure of the collection can be captured. Then for collections sharing similar structures, the edge weights of corresponding edges will also be similar. Thus the main task should be assigning proper edge weights to weighted graphs so that the corresponding edges in two collections A and B own similar edge weight.The edge weights here can be of multiple forms like scalar, vector, matrix, or even higher order tensors. We will use a combination of scalar and vector in our formulation.

Firstly, we will introduce the idea of sample functions and pattern functions. As is stated in [4], each shape can be associated with a functional space. Sample functions is just some sampling points in the functional space. In our problem setting, shapes in a collection only contains isometric deformation among each other, therefore each collection can be associated with a functional space. In collection A, every sample function p_A can induce a set of edge weights g_{A_i,A_j} defined by

$$g_{A_i,A_j} = \frac{||D_{A_i,A_j} p_A||_2}{||p_A||_2}$$

The same for collection B. This is shown in Figure 6 as wll. This implies if we could find some relationship between a set of p_A's and p_B's, it will be possible to conduct analogy between two collections. Two requirements for these sample functions p_A's and p_B's are they should be supported in areas with large variation and their relationship should be accessible. The small set of sample functions satisfying these two requirements are called pattern functions. We will explain why we set these two requirements. Firstly, if a sample function is supported in areas with little variation, the introduced edge weight will have little variation in a weighted graph thus contain little information. In this case, the weighted graph doesn't capture the structure of the collection and cannot be compared. Secondly, if the relationship between the pattern functions from different collection is not available, we cannot ensure the edge weights of corresponding edges are similar and the graph matching cannot be conducted. Given these two requirements, the problems fall on how to find out a set of pattern functions.

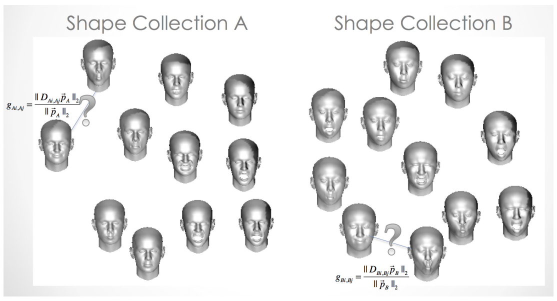
Shape Collection A
Shape Collection B
$g_{Ai,Aj} = \frac{\| D_{Ai,Aj}\vec{p}_A \|_2}{\| \vec{p}_A \|_2}$
?
$g_{Bi,Bj} = \frac{\| D_{Bi,Bj}\vec{p}_B \|_2}{\| \vec{p}_B \|_2}$
?

Figure 6: Show shape analogy from a graph matching view.

A set of pattern functions are found out in the following way. We use $e_{D_{A_i}}$'s to denote the eigenfunctions of D_{A_i}. All the eigenfunctions of D_{A_i} for $i = 1, ..., n$ are gathered together and form an eigenfunction pool. Then we pick up big cluster centers as the pattern functions' candidates. By putting a threshold on the variation of edge weights introduced by each candidate, we can filter out the candidates supported in areas with little variation and the remaining candidates will serve as pattern functions.

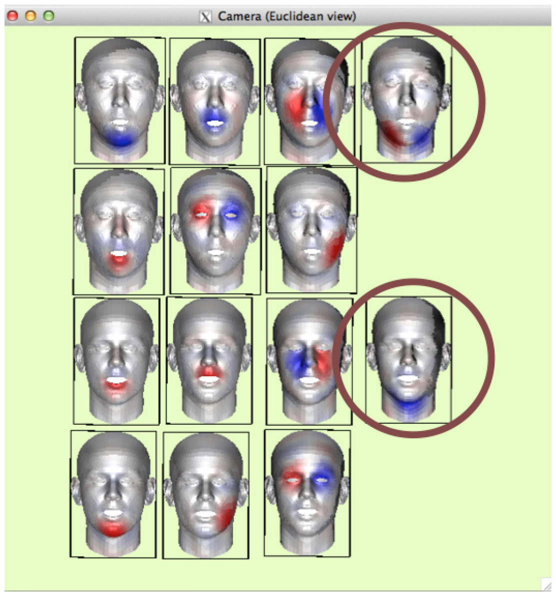

Figure 7: The pattern functions got for two facial expression collections. The first two rows correspond to the first collection and the last two rows corresponds to the second collection. It can be seen that a simple permutation-like linear relationship exists between two collections although there are some outliers as are shown in the brown circles.

For different shape collections, we can use this approach to find their own pattern functions set. Although we get pattern functions for different collections purely based on each individual collection, we observed in our experiments that for collections sharing similar structures, the pattern functions obtained in this way own simple permutation-like linear relationship between different collections. See Figure 7 for an explanation. The linear relationship are easier to get access to than an arbitrary relationship. We will show how we make use of this observation to alternatively refine this linear relationship and refine the graph matching results based on such linear relationship.

3.3 Graph Matching Algorithm

For each shape collection, we will use a set of m pattern functions. We use ${p_A}^{(k)}$, ${p_B}^{(k)}$ (k=1, ..., m) to denote these pattern functions. Each pattern function will introduce a set of edge weights. We use ${g_{A_i,A_j}}^{(k)}$ to denote the edge weights introduced by ${p_A}^{(k)}$, namely

$${g_{A_i,A_j}}^{(k)} = \frac{||D_{A_i,A_j}\,{p_A}^{(k)}||_2}{||{p_A}^{(k)}||_2}$$

If there exists a perfect correspondence between ${p_A}^{(k)}$ and ${p_B}^{(k)}$, namely ${p_A}^{(k)}$ corresponds to ${p_B}^{(k)}$ for k=1, ..., m, then the vector edge weight $({g_{A_i,A_j}}^{(1)}, {g_{A_i,A_j}}^{(2)}, ..., {g_{A_i,A_j}}^{(m)})$ will corresponds to $({g_{B_i,B_j}}^{(1)}, {g_{B_i,B_j}}^{(2)}, ..., {g_{B_i,B_j}}^{(m)})$ and we can directly use these vector edge weights to conduct graph matching based on spectral graph matching technique described in [3]. Unfortunately, this is not the case. We do not know the relationship between pattern functions of different collections. Thus we need to calculate the relationship. We notice that conducting graph matching and calculating pattern functions' relationship are mutually reinforcing processes. Once we have a rough graph matching result, we can calculate the relationship between pattern functions. With the pattern functions' relationship, we will get better graph matching accuracy. These two processes can be executed alternatively in a iterative way. Our coarse to fine iterateable algorithm is just based on this idea.

In our approach, we first synthesize all the edge weights ${g_{A_i,A_j}}^{(k)}$ to a single scalar edge weight

$$\sum_{k=1}^{m} abs({g_{A_i,A_j}}^{(k)} - 1)$$

Then based on this scalar edge weight, we will conduct a rough shape matching step simply using spectral graph matching technique described in [3]. This procedure is shown in Figure 8.

After the rough matching step, we will use the result to calculate the linear relationship between pattern functions of different collections. Each pattern function will introduce a set of scalar edge weight. Therefore, each pattern function corresponds to a weighted graph. These weighted graphs can be compared between different collections and can serve as feature for different pattern functions. This is shown in Figure 9. Based on these graph features, we can calculate a similarity score for each pair of pattern functions. Then we are able to match each pattern function from collecion A to several pattern functions from B by transforming these pairwise similarity scores into some probabilistic matching confidence. After this, we will know the edge weight ${g_{A_i,A_j}}^{(k)}$ has its correspondence in collecion B being $\sum_{l=1}^{m} p_{kl}\,{g_{B_i,B_j}}^{(l)}$. Here p_{kl} denotes the probabilistic matching confidence and $\sum_{l=1}^{m} p_{kl} = 1$. Knowing the relationship between pattern functions of different collections enables us to use vector edge weights to conduct graph matching as is shown in Figure 10.

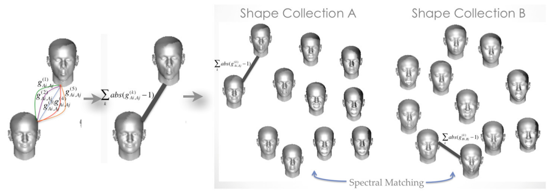

Figure 8: The process of rough shape matching

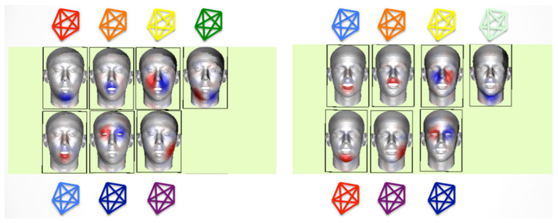

Figure 9: Each pattern function is associated with a weighted graph which can serve as a kind of feature. We can evaluate similarity scores between pattern functions using these graph features.

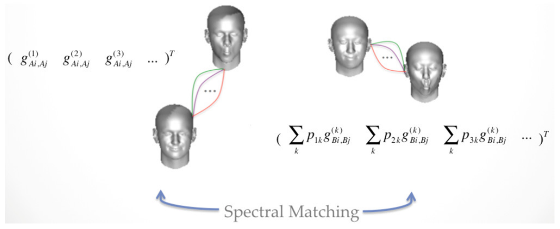
$(g^{(1)}_{Ai,Aj} \;\; g^{(2)}_{Ai,Aj} \;\; g^{(3)}_{Ai,Aj} \;\; ...)^T$
$(\sum_k p_{1k} g^{(k)}_{Bi,Bj} \;\; \sum_k p_{2k} g^{(k)}_{Bi,Bj} \;\; \sum_k p_{3k} g^{(k)}_{Bi,Bj} \;\; ...)^T$
Spectral Matching

Figure 10: Using vector edge weights for graph matching.

The above two steps, probabilistic pattern function matching and graph matching based on vector edge weights, can be run iteratively. We will call each pass one refining iteration. We can run the algorithm until the analogies result converges.

4 Experiments

To evaluate our proposed alorithm, we test it on a facial expression dataset used in [5], which contains 2*40=80 face models from 2 people. Each person corresponds to a set of 40 face models which contains the same set of expressions. This means the ground truth of shape analogy is known. Pointwise correspondence within model collection of each person is known and cross collection correspondence is unknown. Figure 11 shows how the shape analogies accuracy changes with the algorithm goes on. It can be seen that rough matching only achieves a 20% accuracy. But with the refining iteration step, the algorithm will achieves 100% shape analogies accuracy after 3 iterations.

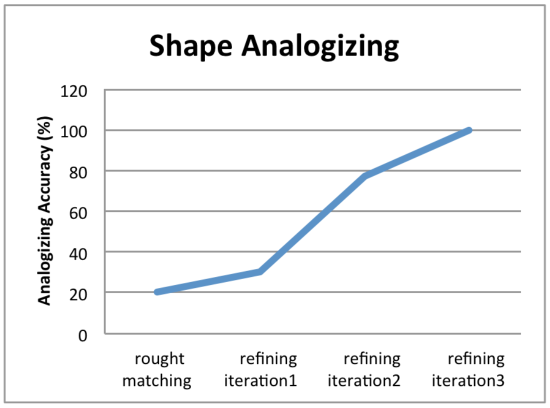

Figure 11: The overall shape analogies accuracy.

We pick up a few shape analogies results and show them in Figure 12. It can be seen that our shape analogies perfectly align two collections according to the facial expressions, which is quite abstract and not easy to capture. This shows the power of specifying shape variation via shape analogies.

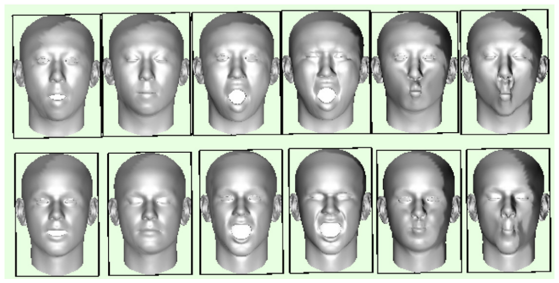

Figure 12: Shape analogies results.

5 Conclusions

In this report, we proposed a novel framework to tackle the shape analogies problem between two collecions in a graph matching way. We use novel descriptors to capture the intrinsic characteristics of shapes and based on these descriptors we are able to formulate the shape analogies task as a graph matching problem. We use a coarse to fine iterateable algorithm to solve the graph matching problem and thus solve the initial shape analogies problem. The experiment results show the effectiveness of our framework.

The most important meaning of this report lies in following two aspects. Firstly, we propose to use analogy to specify the variation between 3D shapes, which has intense application in many areas like shape retrieval, shape manipulation, shape synthesization. Secondly, instead of using geometric features purely based on individuals to describe shapes, we suggest using shape relationship as an alternative. This means we try to define a shape by its neighbor shapes. Using relationship to describe shapes will be much more effective and discriminative compared with purely geometric features as the shape collecion grows larger and larger. Also the relationship can be compared between different collections much more freely.

Some future directions are handling shape analogies between collections only sharing partial common structures, expanding this framework to handle much more heterogeneous shape collections like man-made object collection. Of course, this framework should be tested in larger dataset to further verify its effectiveness.

References

[1] Second workshop on fine-grained visual categorization (fgvc) at cvpr 2013. `http://www.fgvc.`

org.

[2] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[3] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1482–1489. IEEE, 2005.

[4] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):30, 2012.

[5] Raif M Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4):72, 2013.

[6] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM, 2004.

[7] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3):441–471, 2008.