A Study on View-based 3D Model Retrieval from a Single Input Image

A Study on View-based 3D Model Retrieval from a Single Input Image

Vicky Sintunata
B4ID2501
Graduate School of Information Science
Tohoku University
Miyagi, Japan

July, 2017

Acknowledgement

First and foremost, I would like to thank my supervisor Prof. Terumasa Aoki for his kindness and patience during my Ph.D study and research. Without his guidance and insight, this dissertation will not be as it is. I would also like to thank all the committee members Prof. Yoshifumi Kitamura, Prof. Ayumi Shinohara, and Prof. Takuo Suganuma for their comments and their guidance to improve this dissertation. I would also like to address my gratitude to Mitsubishi Corporation International Student Scholarship who supported me during my Ph.D study.

I would also like to give my gratitude to all the Aoki Laboratory members (present and past) who had not only supported me but also shared their knowledge to this point. A huge thanks to my family and friends here in Japan: the Indonesian Students Association in Japan, Miyagi Chapter (PPIS), Persekutuan Doa Sendai (PDS) and Keluarga Katolik Indonesia Sendai (KKIS), Sendai English Fellowship (SEF), and Mafu-mafu family who always supported and brought joy to me. Special mention to Eddwi Hesky Hasdeo, Evelyn Pratami Sinaga, Stevanus Kristianto Nugroho, and Ellen Tanudjaja who always encouraged and supported me during my research, disseration writing and defense. I'm really glad to meet and spending time with all of you! Last but not least, I would like to thank my parents and my sisters who always supported me all up to this point in my life.

Contents

Abstract

With the increasing number of models made available freely, the number of repository will also increase. This leads to the need of a fast and accurate 3D model retrieval system. In this study, two types of (view-based) 3D model retrieval systems, namely the rigid 3D model retrieval and non-rigid (articulated 3D model) retrieval system is studied. Since a skeleton information is needed to overcome the non-rigid retrieval system, a novel skeleton extraction algorithm with sparse points condition is also proposed. The experiment on the rigid 3D model retrieval shows that the proposed method can enhance the rerieval speed 14 times faster (in matching time) than the conventional view-based retrieval system. Furthermore, the proposed method can also approximate the orientation of the query input.The skeletonization proposed achieves a better visual result and achieve 1.7 times higher F-measure compared to an existing SSM method. Finally, our proposal on the articulated 3D model retrieval by combining the shape context and skeleton information achieves a promising result of 93% (on average) correct retrieval rate.

List of Figures

List of Tables

Chapter 1

Introduction

Creating a 3D model from scratch is not a trivial task. It is very time consuming and needs a lot of effort. Not to mention that usually only a skillful person can create a good quality of 3D models. Even for a skillful artist, creating a 3D model still takes a lot of time and therefore, instead of creating everything from zero, it is much preferable to have a repository of 3D models (Fig.1.1). With the increasing amount of 3D models in the repository (database), it is necessary to have a system that can retrieve the desired 3D model fast and correctly.

3D model retrieval mainly can be classified into two approaches: model-based 3D model retrieval and view-based 3D model retrieval. While the first one takes a 3D model as an input, the second approach takes a 2D input such as images or sketches as the input. There are many approaches that has been proposed in the model-based 3D model retrieval, but there are less research for the view-based 3D model retrieval. Furthermore it is much often that we do not have the intended 3D models at hand, instead we only have an image of the models or even only a sketch. Therefore, the view-based 3D model retrieval is more applicable compared to its counterpart the model-based 3D model retrieval. Thus, view-based 3D model

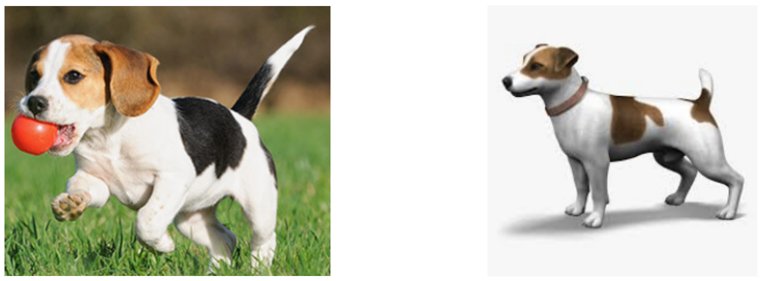

(a) input (b) output

Figure 1.1: Input an image and output the desired 3D model

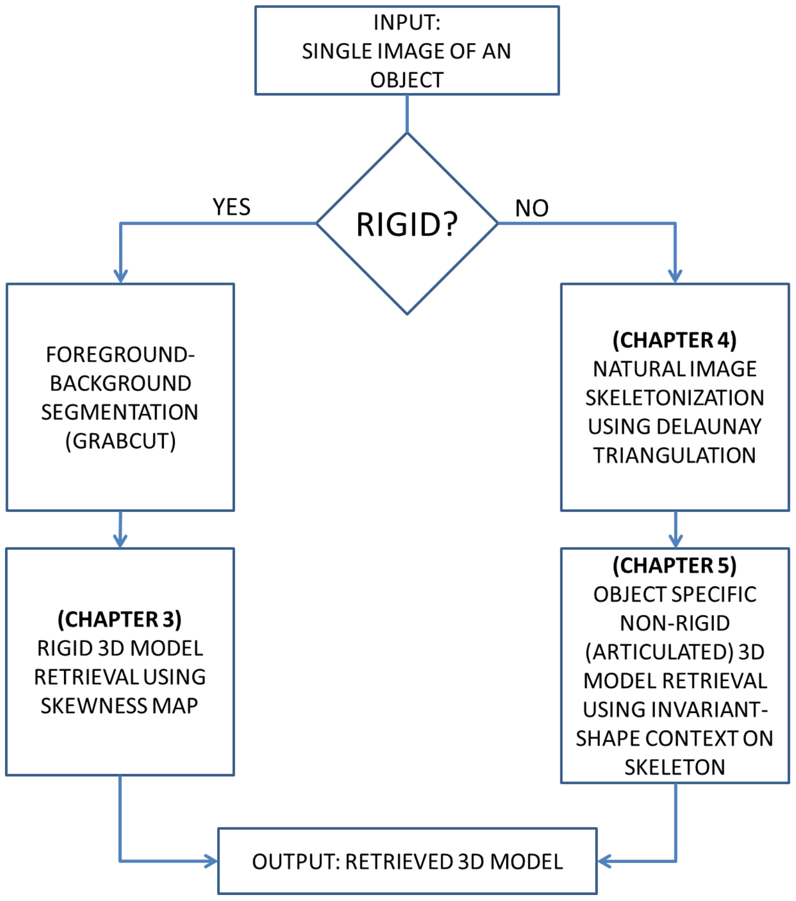

INPUT: SINGLE IMAGE OF AN OBJECT
RIGID?
YES
NO
FOREGROUND-BACKGROUND SEGMENTATION (GRABCUT)
(CHAPTER 4) NATURAL IMAGE SKELETONIZATION USING DELAUNAY TRIANGULATION
(CHAPTER 3) RIGID 3D MODEL RETRIEVAL USING SKEWNESS MAP
(CHAPTER 5) OBJECT SPECIFIC NON-RIGID (ARTICULATED) 3D MODEL RETRIEVAL USING INVARIANT-SHAPE CONTEXT ON SKELETON
OUTPUT: RETRIEVED 3D MODEL

Figure 1.2: System Overview

retrieval will be the main topic in this research.

3D model itself can be classified into two types: rigid 3D models and non-rigid (articulated) 3D models. Notice that throughout the entire content of this dissertation, the term non-rigid is regarded also as the articulated 3D models and therefore will be used interchangably. In rigid 3D models the pose of the object does not change or remain in rigid form. For example cup will remain a cup irrespective of its "pose". The second type, non-rigid 3D models, the object might be one, but the pose of the object might be different. For example: a human standing and a human running are two models with the same object (human) but different pose (standing and running). Based on these types of 3D models, the challenge to retrieve the model from the database will also differ and of course the non-rigid types are much more challenging than the rigid 3D model type. More of the related work can be found on chapter 2.

The overall system can be seen in Fig.1.2. Notice that in the input to the

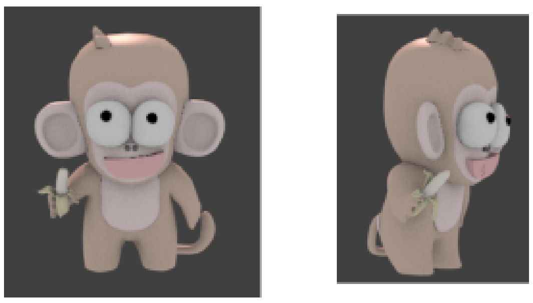


Figure 1.3: Shape might be different based on different viewing direction

system is a single natural image of a real 3D object, although for simplicity in the research done here we were using a single rendered image of a 3D CG model, with the exception of the input for the skeleton extraction in chapter 4.

In chapter 3 we will discuss more about the first type of retrieval, i.e. the rigid 3D model retrieval. Since we are dealing with the view-based 3D modeling, the simplest way to retrieve the object is to create a database of the rendered image of the 3D models. Unfortunately, depending on the viewing angle, the shape or the feature of the object might be different (Fig.1.3). A brute force way to handle is to render the 3D model from every possible viewing angle, so an object will be rendered at most ($360 \times 360 =$) 129,600 times. If there are 1000 objects in our repository, that means there will be 129,600,000 in total. Of course matching one input image of the model (or object) we want to create with 129,600,000 images require a lot of time and effort. Therefore, we propose a speed-up process to be used for this large database matching using the skewness map method. From the experimental results, it is shown that the matching process is 14 times much faster than conventional method. Furthermore by using the skewness map approach, the orientation of the object can also be approximated.

One way to solve the non-rigid 3D model retrieval is by using the information the object's skeleton provided. Although posing differently, the general classification of the object can be retrieved by the skeleton information. For example the skeleton generated from human model is definitely different from the skeleton generated from elephant. Therefore, skeleton is one of the key to solve the non-rigid 3D model retrieval.Before moving on to the non-rigid 3D model retrieval, we will discuss deeper regarding the skeletonization or skeleton extraction algorithm in chapter 4.

Skeletonization comes not without a limitation (Fig.1.4). Conventional skeleton method (left image in Fig.1.4) relies heavily from the binary image input. While it can be achieved by doing the foreground-background segmentation, segmentation itself is still an open problem in this area. To alleviate this limitation, the grey-scale skeletonization come into development (middle image of Fig.1.4). In this

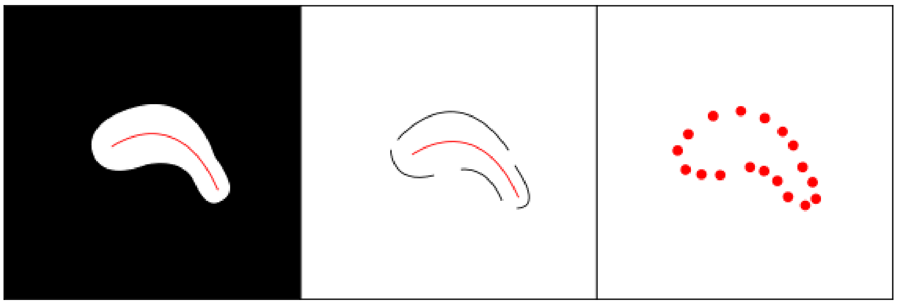


Figure 1.4: Skeletonization Limitation

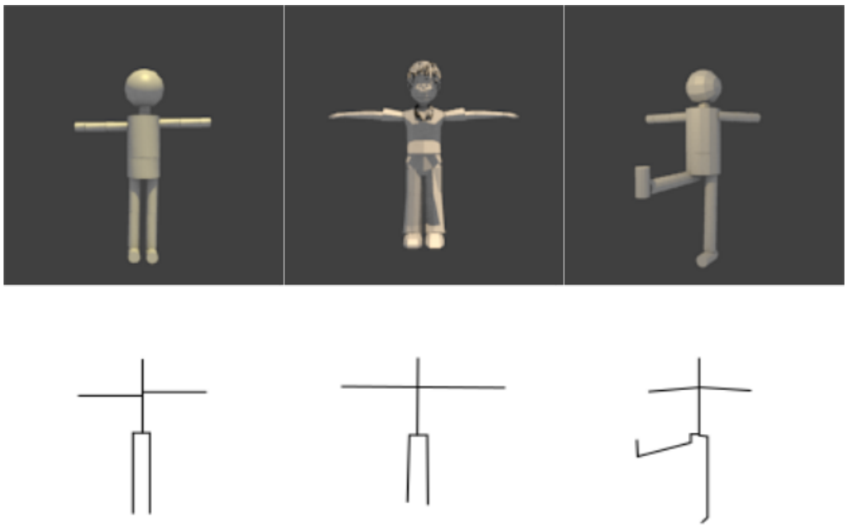
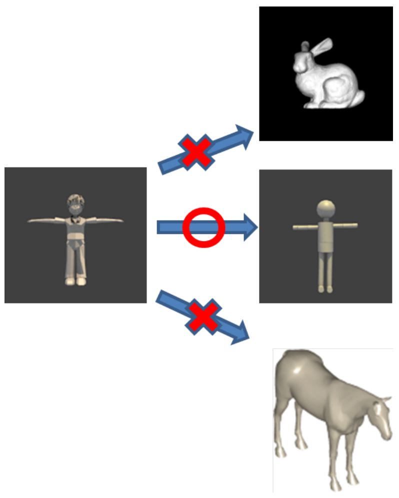


Figure 1.5: Different object might have similar skeleton in non-rigid 3D model retrieval

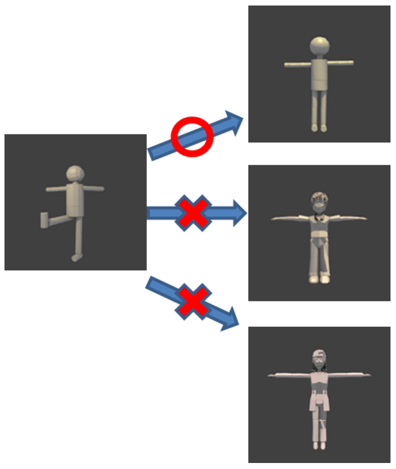
approach, a closed-curved boundary constraint on the binary image skeletonization is relaxed into the semi-closed curved boundary constraint. This means that the edge or boundary of the object should be connected one and another in some extend. More often when the background and the object in the image share a very high similarity (such as texture or color) with the background, the connectivity (and therefore the semi-closed curve boundary) constraint cannot be fulfilled (right image in Fig.1.4). In chapter 4, we are proposing a method to solve this problem.

In the fifth chapter a non-rigid 3D model retrieval is using the skeleton information is proposed. As far as we know, there are no existing literature regarding the fully view-based non-rigid 3D model retrieval system using only one input image. As mentioned beforehand, using only skeleton in retrieving the 3D model will only result in similar type of model being retrieved (Fig.1.5). Two objects with the same pose will have the same skeleton information or very similar information. When we want to retrieve the exact object from the database, it becomes very challenging if we only relies on the skeleton information (Fig.1.6). In order to solve this problem, we are incorporating the shape context descriptor, so that the retrieval system can retrieve the same object correctly regardless of its pose. Early experiment shows a promising result of this approach. Unfortunately, there are still some challenges that need to be addressed, namely the rotation and scale robustness of the shape context. A new method based on the cross correlation method is proposed in this chapter to minimize these challenges.

Finally the sixth chapter will conclude this research. In this chapter a summary of what have been achieved and the remaining challenges for future direction in this research will be presented.



(a) Class specific retrieval



(b) Pose invariant object retrieval

Figure 1.6: Relying only on skeleton information is insufficient to retrieve an object specific object (b)

Chapter 2

Related Work

In this chapter we are going to review several of the past works that had been done to solve the rigid and non-rigid retrieval. As mentioned on the first chapter, we will also review several algorithm of skeletonization. Finally, at the end of this chapter, several methods that will be used as a part in our proposed method will be described.

2.1 Rigid 3D Object Retrieval

Several authors [1, 2, 3] classified the problems of 3D object retrieval into model-based and view-based 3D object retrieval. Model-based systems are dealing with 3D model as input (or query), while view-based systems are dealing with 2D input(s) such as images, sketches, etc. The term query and input will be used interchangeably unless mentioned otherwise. Although the focus in our research is mainly dealing with the 2D input, for completeness sake we will also review some of the methods used to solve the model-based 3D object retrieval.

One of the core problem in a retrieval system is how to match the query and the content of the database. One such example is a method proposed by Shah et al. [4] which utilizes a local surface descriptor based on the divergence for feature matching between two 3D models. Another approach to solve model-based problem using 3D curve-matching had also been proposed by Feinen et al.[5]. In [6] the authors proposed a method which project the 3D model into a panoramic view (cylindrical projection) which is used as a descriptor for the retrieval system. Based on the same descriptor, the authors in [7] proposed a method which converts the qualitative descriptions into attribute signature and reference set signature which can be considered as an annotation problem for 3D model retrieval system.

The problem is much more pronounce in the 3D-2D matching (view-based systems) since, depending on the viewing angle, the shape of an object might

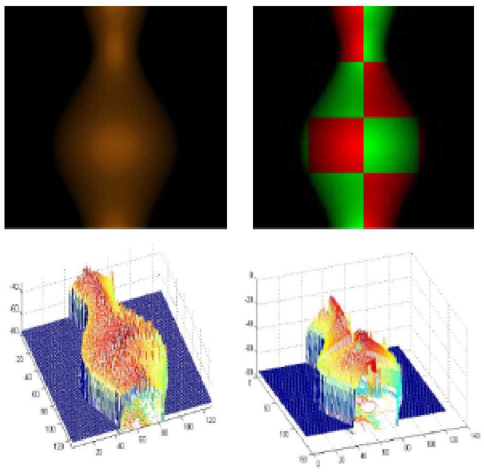


Figure 2.1: Shape from shading drawbacks [12]

differ. One might elevate this problem using methods such as shape from shading [8, 9, 10, 11] to get the semi-3D information of an object and then matching the available information with the models in the database using the aforementioned methods. Unfortunately shape from shading has its own problem, such as the specular reflection and color effect [12, 13] and therefore, the generated semi-3D model does not represent the object well (Fig.2.1).

Gao et al.[1] proposed a hypergraph analysis to solve the view-based retrieval system (Fig. 2.2. A hypergraph is a graph with edges that can connect to several nodes. The nodes in the hypergraph are considered objects in the database and the edges are connected to a cluster of views. Multiple hypergraphs are constructed in their proposed method to model the objects and their connections with each other. Retrieval and recognition were all done based on these hypergraphs. One important point regarding these hypergraphs is the weight on each edges. While for retrieval the weight were set simply using equivalent weight, the recognition weights were learned from the hypergraph fusion. Note that when a new object is inserted to the database, a new hypergraph (including all the weights) needs to be established again. Contrary to our proposed method where each object can be considered to be independent with the others in the database.

Similar with [1], Ke et al.[2] proposed a retrieval system based on a graph method. First, bipartite graphs which match images (object viewed from several angles) feature (49 Zernike Moments) of one object with another were created. From these bipartite graphs, a global graph with objects as nodes and weighted edges connecting these nodes were generated. To determine the weights, a semi-supervised learning method was applied.

Wang et al.[3], in order to solve the view-based retrieval system, proposed a

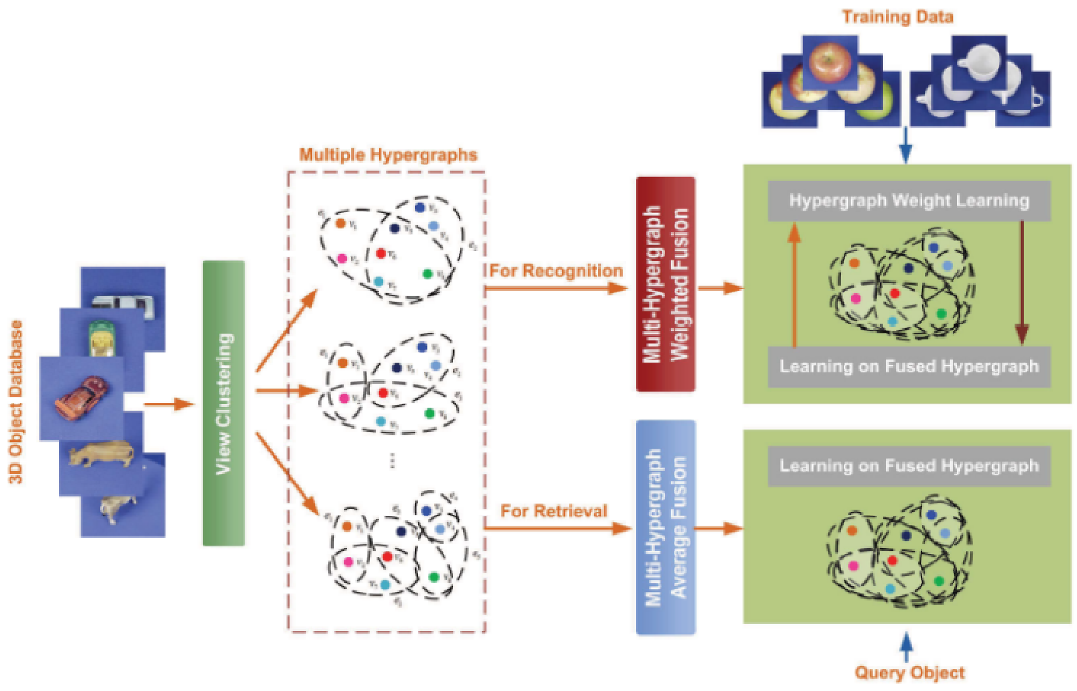

Training Data
3D Object Database
View Clustering
Multiple Hypergraphs
For Recognition
For Retrieval
Multi-Hypergraph Weighted Fusion
Multi-Hypergraph Average Fusion
Hypergraph Weight Learning
Learning on Fused Hypergraph
Learning on Fused Hypergraph
Query Object

Figure 2.2: Hypergraph method proposed in [1]

learning method based on the Gaussian Mixture Models (GMM) of the object. First, several view images taken from different viewing angles oof each object will be taken and then 49 Zernike Moments as features of each view will be calculated. A general GMM based on these features will then be calculated. Based on the general GMM, an adapted version (object specific) GMM parameters will then be calculated. In order to retrieve the corresponding object with the query, a distance measurement based on the Kullback-Leibler (KL) divergence is then employed. Note that the methods mentioned above [1, 2, 3] requires a learning step in one of their steps, contrary to our proposal which does not require any learning at all.

Mahmoudi and Daoudi[14] proposed a method by utilizing a characteristic views of 3D object. First a 3D object is represented by 7 characteristic views. The first three characteristic view (principal views) are taken from the Principal Component Analysis (PCA) of the 3D object. The rest of the characteristic views are deduced from the principal views. In the retrieval step, two curvature scale space (CSS) distance between the query and the models in the database will then be calculated. M-tree is utilized to reduce the calculation time when calculating the distance between the CSSs.

Latecki and Lakämper[15] proposed a method which utilizes the discrete curve evolution. The shape of an object will be simplified to reduce the influence of

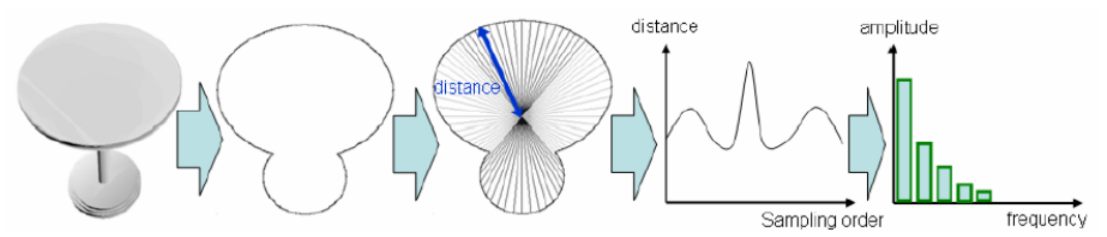


Figure 2.3: Magic Canvas proposed in [17]

noise and remove "irrelevant" shape features. A minimum value of similarity measurement, which calculates the arc similarity between the curve of the objects, will determine several objects in the database to be retrieved.

Yang et al.[16] proposed a method which combines the skeleton and contour information of an object. First skeleton information are extracted from the object. From the extracted skeleton, contour segments belonging to those skeleton end-points can be determined. Twelve contour features can be extracted from each contour segment. Finally for the retrieval step, a similarity measurement that combines the skeleton similarity value and contour segments similarity value are calculated.

Recently Liu et al.[17] proposed a method that utilizes multi-modal clique-graph matching. While this method's strength lies in the multi-modal utilization, a single modal (or feature) utilization method can also be used. The most important thing in this method is the clique-graph generation which kind of similar with the hypergraph generation in [1]. The difference is that in the clique-graph, not only hyper-edges but also hyper-nodes (cliques) are used. To create the hyper-nodes correspondences a hierarchical agglomeratice clustering (HAC) is applied. Similar with [1, 2] since a graph based is used, when new objects are inserted, all the process (correspondences between objects) should be established again.

Magic Canvas[18] is a system where user can draw a sketch and a selection of 3D object based on the sketch will be retrieved (provided) to the user to be selected (Fig.2.3). Their 3D model retrieval method is based on the Fourier transform of the distance measurement of the sketch's contour. First, the contour of the query is retrieved and then the distance from the center mass of the object to each contour point is recorded. This distance data will then go through a Fourier transform to get the frequency information. In further step, these frequency feature will then be used as the global feature. In order to differentiate object with the same frequency, a local feature based on the inverse Fourier transform aof the filtered frequency is then calculated. The linear combination of these features will be the score of the object (similarity score).

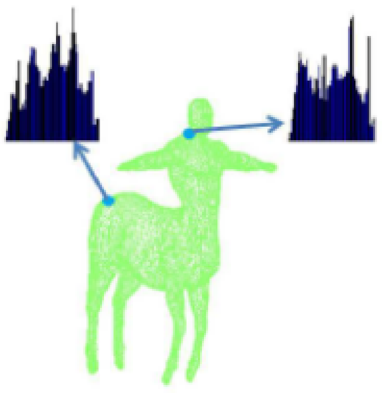


Figure 2.4: Bag of Geodesic Histogram (BOGH) method proposed in [20]

2.2 Non-rigid Retrieval

A good survey regarding the non-rigid retrieval method can be found in [19, 20, 21]. Some of the papers will be discussed here.

In[19, 20] Nguyen and Porikli proposed a method by using the Bag of Geodesic Histogram, a combination of bag of feature and normalized geodesic distance approach (Fig.2.4). A geodesic distance is defined as the shortest path from one points to another point on the model. Then, a set of points on the model will have each a feature descriptor of these geodesic distance between the points on the model which is encapsulated in a histogram. Finally, when matching between two objects. the Hungarian algorithm is used with the Chi-square histogram distance is utilized to calculate the difference between the histograms.

Lian et al. [22] proposed a method using a CM-BOF (Clock Matching - Bag of Feature) method (Fig.2.5). This method first normalize the input object into its canonical form. These canonical forms are then rendered into multi-view images and SIFT descriptors are extracted from the rendered images as feature vector. Then, using the Bag-of-Feature method, a cluster of thse feature vectors will then be used as a codebook that will be used to represent the object. Finally, the matching is done using the clock-matching method which basically compare the multi-view of the object with the codebooks (database) generated from the previous step. Contrary to this method, our proposal uses only one input image.

Localized statistical feature (LSF)[19, 20] and its improvement SV-LSF[21] has also been proposed. The method starts by taking sample points from the surface of the 3D model and generate a sphere of interest (SOI) around the points and checking the point on the surface of the models. The feature of each point is extracted by computing 4D joint histogram consisting of angles and distance information among all the pairs of oriented points inside the SOI. In [19, 20] the

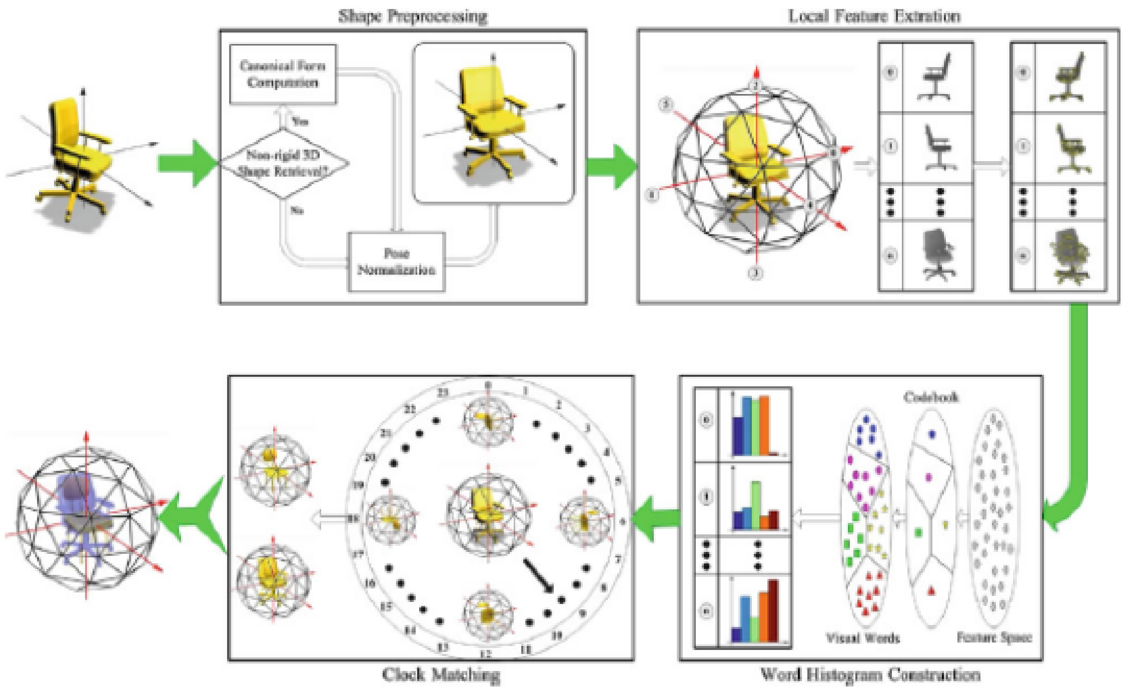


Figure 2.5: CM-BOF proposed in [22]

features are then combined using the bag-of-feature method and in [21] a super vector based method is used instead.

In [23] Li et al. proposed a method which utilize the Spatial Structure Circular Descriptor (SSCD). Fig. 2.6 shows this method. The SSCD image is created from the projected points of the 3D model into the minimal bounding sphere. In further step, this bounding sphere is projected into the image plane. These points along its spatial feature are then sent into a modified bag-of-feature method such that it is translation-invariant. The difference between the model is then calculated using the earth-moving distance. Notice that in order to get the SSCD images, a 3D model of the input is needed.

Bronstein et al.[24] proposed a method based on the (diagonal) heat kernel (or heat kernel signature) of the object. Like any other previously mentioned approaches, they use the bag-of-feature approach when discriminating the input object and the objects in the database. Instead of using the normal bag-of-feature approach, they proposed a modified bag-of-feature which instead of using independent feature between one point with another, they use the spatial relationships of nearby points also.

Although not specifically designed for 3D model retrieval, a skeleton-based method has been proposed by Bai et al.[25] to do the object detection in an image.

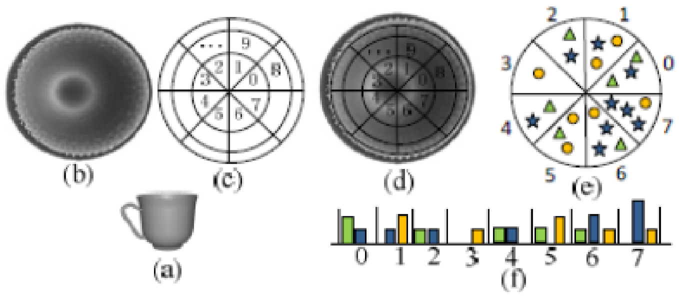


Figure 2.6: An SSCD-based method proposed in [23]

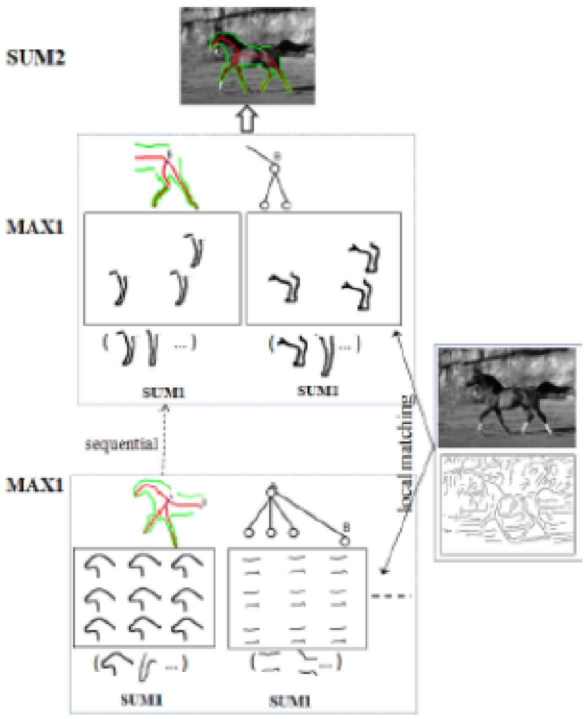


Figure 2.7: An object recognition method proposed in [25]

The method takes a bayesian approach to detect the object in the image. Given a skeleton and its (semi- or full) contour information, the method will then retrieve (or detect) the object in the image. First the skeleton is classified into several types, i.e. the junction point skeleton and branch point skeleton. By classifying the skeleton type, a tree structure of the skeleton can then be generated. Since the method is based on a bayesian approach, a training step is required. The learning parameter is then applied to the tree structure of the training images along with the contour information. To localize the object in the image, a sum-max algorithm [26] is then applied, i.e. summing each part with the maximum similarity of the contour (shape) of the object and the object in the database (Fig.2.7).

2.3 Skeletonization

Based on the input, skeletonization can be classified into two types: 2D and 3D skeletonization. Since our focus deals only with the skeletonization of 2D images, only 2D image skeletonization will be covered here. Further classification can be applied to the 2D skeletonizaton, i.e. binary image skeletonization and grey-scale skeletonization.

Most of the approaches deal with the binary image skeletonization. Cornea et al.[27] classify skeletonization into 4 classes, i.e. thinning and boundary propagation method, distance field based method, geometric method and general field method. In thinning and boundary propagation, the skeleton is extracted by removing the boundary (object) pixel iteratively until the required thinness is obtained.

One of the most famous algorithm to extract the skeleton from binary image using the thinning type of approach is Zhang-Suen algorithm [28, 29] The algorithm works by checking the neighborhood of all one-valued pixel (white) and the transition of the binary pixel from zero to one. There are four predefined conditions that need to be checked for each pixel of the binary image, ie. the number of zeros (black) and ones (white) pixels on the eight neighborhoods, the number of transition as mentioned before and two of the conditions that checked the value of the neighbors' value of a 4 -neighborhood system (north, east, west south positions of a current pixel). If a pixel fulfills the predefined conditions, then the current pixel will change its current value from one to zero, i.e. from white to black. Fig.2.8 shows the visualization of this method.

In distance field based methods, skeleton is extracted by checking the position of an inner pixel to the boundary of the target object [30, 31, 32, 33, 34, 35]. Initial skeleton generated by a distance transform usually produce a lot of redundant skeleton branches. Therefore a pruning step is usually needed to get less redundant and much more informative skeleton. In [35] a skeleton pruning using the discrete

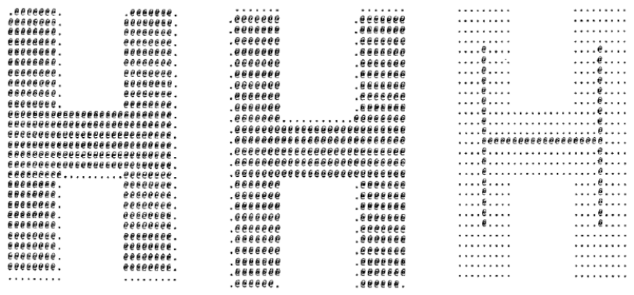


Figure 2.8: Thinning method proposed in [28]

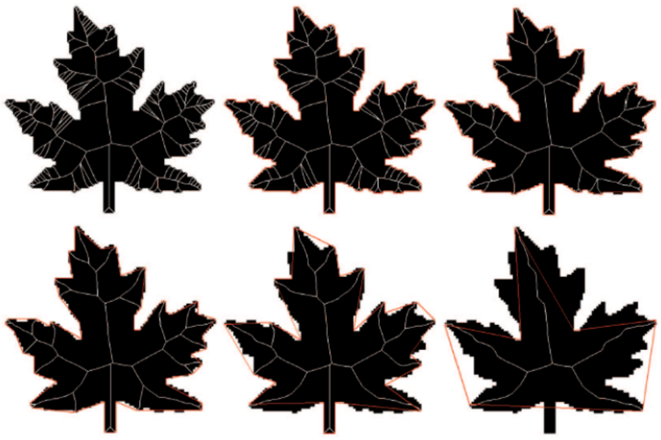


Figure 2.9: Distance field based method and its pruning step proposed in [35]

curve evolution is proposed. Basically the method will check every branch of the skeleton. If the branch does not belong to the simplified contour of the object (from the evolution of the curve), then it will be removed (Fig.2.9).

In geometric methods, polygonal meshes or scattered point sets are used. Reeb graph and Voronoi diagram are some of the examples belonging to this class. The potential field function and radial bases function are the examples of the methods under the general-field functions. The most crucial part of the approaches belonging to the binary image skeletonizaton is the availability of some closed-curved boundary, represented by the binary image. Without a closed-curved boundary object, these approaches cannot be used.

As some might notice that closed curved constraint might not always be achieved in an image due to some occlusion or high similarity of the object characteristics (such as color) with the background. With this restriction, the second type of 2D skeletonization, i.e. the grey-scale skeletonization, came into development. Skeleton is closely related to symmetry, therefore several authors proposed methods based on the symmetry information.

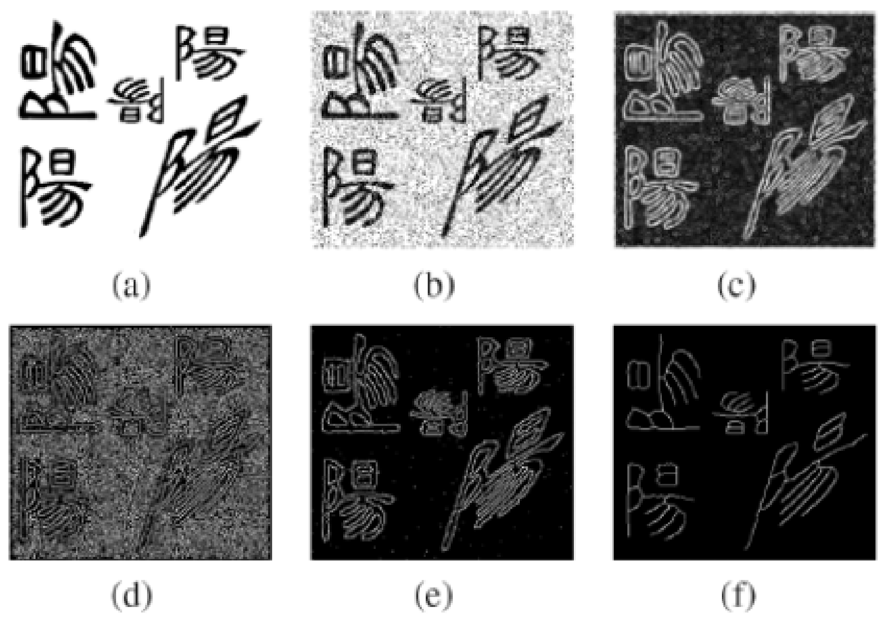


Fig. 12. (a) Original image composed of five samples of a Chinese character by affine transform; (b) the image with "salt and pepper" noises; (c) the modulus image of the WT with $s = 4$; (d) the image of the modulus maxima of the WT without the threshold processing; (e) The image of the modulus maxima of the WT after the threshold processing; (f) The final skeletons extracted by the amendment process.

Figure 2.10: Wavelet-based approach proposed as in Figure 12 in [36]

You and Tang [36] proposed a character's skeleton extraction method using wavelet (Fig.2.10). They proposed a new wavelet function to detect the symmetry of the character in the input image. Although the extraction perform well, the generated skeleton has a discontinuity in some of its part. Therefore, they also proposed a method to amend this drawback by connecting the loss skeleton. A character usually has a distinguish uniform color and contour and so the method perform really well. Unfortunately, in natural images, such condition is rarely or hard to be achieved.

Widynski et al. [37] proposed a method based on particle filter approach. Particle filtering approach requires a training or (a) known distribution(s) beforehand. Levinshtein et al. [38] proposed a method based on superpixel segmentation and learned affinity function. First a multiscale superpixels are generated from the input image. In the training step, an affinity matrix, i.e. how similar a superpixel with its surrounding and other superpixels are learned. Based on the affinity matrix, a connection between the center of each superpixel will be generated. Finally

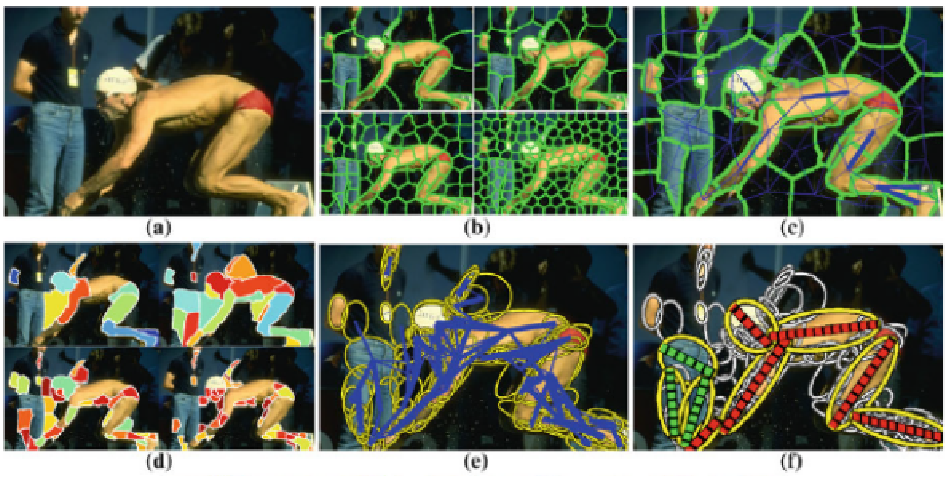


Figure 2.11: Multiscale symmetry detection proposed in [38]

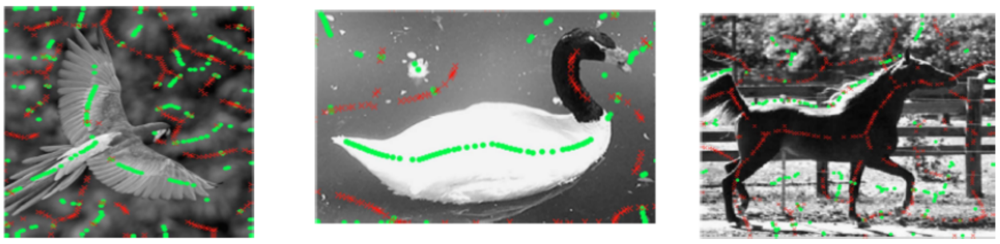


Figure 2.12: Grey-scale skeletonization using structure adaptive anisotropic filtering in [39]

a symmetry (or in this case, the skeleton) is extracted through a graph-based segmentation approac (Fig.2.11). Notice that in these last two methods, a supervised learning step is incorporated.

Du et al. [39] proposed a method based on structure-adaptive anisotropic filtering (Fig.2.12. The intuition behind the adaptive approach is that large scale (of the kernel) is preferable in edge and smooth area, while smaller scale are performed on the skeleton points. An iterative approach which updates every pixels' value and parameters is used in this approach and therefore requires a high computational cost.

Quannan et al. [40] proposed a grey-scale skeletonization based on the distance transform method [41] (Fig.2.13. Instead of using the distance transform directly, the inverted distance transform is used. The authors argued that the relative value between a skeleton point and its neighbors is significantly larger for the inverted version than using the distance transform and therefore easier to work with. In a later step of the method, the authors applied a foreground-background segmentation-like (taking the higher level information) using a method described

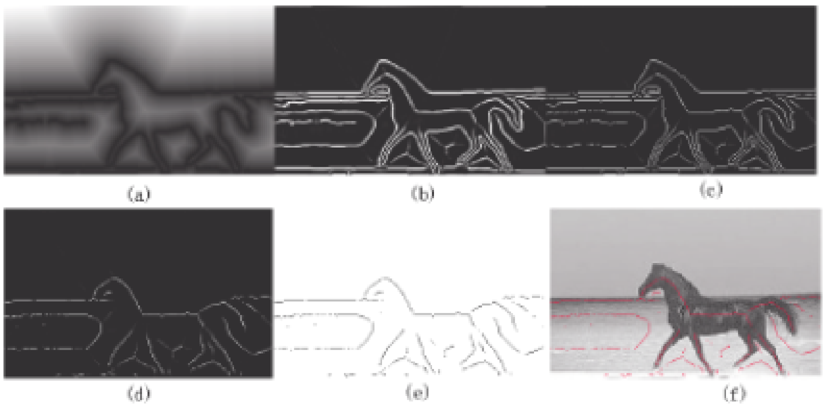


Figure 2.13: Grey-scale skeletonization using skeleton strength map in [40]

in [42]. By doing so, the skeletonization can then be converted into a binary image skeletonization problem.

Note that both of the works in [39] and [40] utilize the Skeleton Strength Map (SSM). The only difference is the calculation method. While in [39] the calculation is based on the divergence calculation, in [40] the diffused gradient vector field is utilized instead.

2.4 Background for the Proposed Methods

The first two subsections will describe image features that will be used (or be the background) in the proposed methods in the subsequent chapters. Afterwards, the grabcut algorithm which is very common to be used as the foreground-background segmentation will also be described. In the fourth subsection, interpolation methods will be discussed. A bicubic spline interpolation method which will be used in chapter 3 and bezier curve interpolation used in chapter 4 will be briefly explained. Finally, the delaunay triangulation which is one of the core method in chapter 4 will be described briefly.

2.4.1 SIFT and SURF

There are four steps in SIFT feature detection: scale-space extrema detection, keypoint localization, orientation assignment and keypoint descriptor. In the scale-space extrema detection, a difference of Gaussian is used to identify the candidates for interest points that are invariant to scale and orientation. The difference of gaussian is defined as the difference of two nearby scales separated by a constant multiplicative k [43]. In order to find the extrema location, 26 of its neighbors (8 pixel neighbors in current scale, 9 pixels in lower scale, and 9 pixels in higher

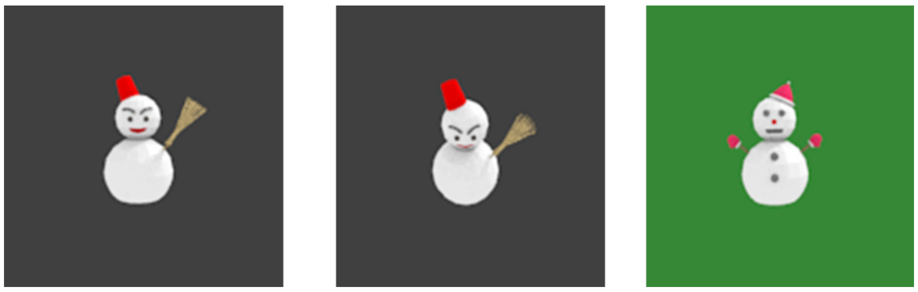


Figure 2.14: Example images of similar objects

scale) are compared. The point will be selected if the sample point is larger (or smaller) than all of these neighbors.

In the next step, the location of these extrema or minima points are located through the keypoint localization. Candidate which has a low contrast and a strong edge responses are eliminated because they are very sensitive to noise. In eliminating the edge responses, the element of the Hessian matrix are used to calculate the ratio of principal curvature. Point with ratio more than the pre-determined threshold will be rejected. At each keypoint, a gradient and (therefore) orientation at a region surrounding it, is computed at the orientation assignment step. Finally a keypoint descriptor is defined by calculating an orientation histogram, which will generate (4×4(subregions)$\times$ 8 (bins) =) 128 dimensional vector at each keypoints.

SURF is a faster version of SURF. The less computational costs is achieved by utilizing the integral image approach and using a box filter to approximate the (second order) gaussian filter. Compared to the SIFT method where the image needs to be subsampled and the gaussian filter needs to be applied iteratively, by using these approaches (integral image and box filter), SURF can reduce the computational costs of iteratively reducing the image size (during the scale-space analysis) and calculating the filtered image. For the matching between SIFT or SURF descriptors, Fast Approximate Nearest Neighbor (FLANN, [44]) which utilizes either randomized kd-tree algorithm[45] or hierarchical k-means algorithm, can be used. The smaller the distance is, the more similar the object to be retrieved.

Consider some images in Fig.2.14, where the left most image is the query, the middle and the right images are the images in the database. Applying SURF to the images and calculating its distances will give us a result of 8.324 for the query-middle pair and 7.343 for the query-right pair. This values mean that the right pair is much more similar according to the SURF feature, which is a false conclusion. This shows that in some cases, SURF is not really suitable for a texture-less object as in Fig.2.14 as also noted in [46, 47].

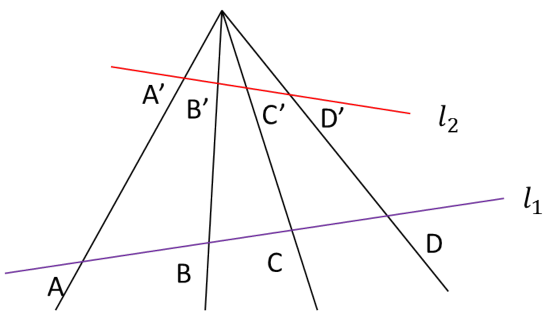


Figure 2.15: Cross ratio example

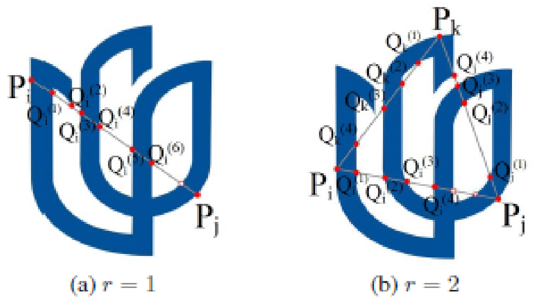


Figure 2.16: An example of Characteristic Number in [47]

2.4.2 Cross Ratio based Feature

Recently, a shape descriptor based on cross ratio calculation has been gaining popularity in pictogram matching area [46, 47]. This is due to the characteristic of cross ratio that maintains segments of line ratio under a perspective projection. Li and Tan [46] proposed a cross ratio spectrum descriptor. In their method, they used a set of cross-ratio from the combination of point lying on the outer contour of the pictogram. When calculating the cross ratio, only two points inside the pictogram were used. Luo et al.[47] argued that using only tow points inside the pictogram made the method not comprehensive enough. Therefore, they proposed a Characteristic Number (CN) approach (Fig.2.16) which also utilizing cross ratio, but instead of using only a pair of sampling points to calculate the cross ratio, they used three sampling points on the boundary of the object creating a triangle and calculating all the cross ratio along the edges of the triangle and therefore more information can be acquired. Although the proposed method performs well in terms of accuracy, CN requires a lot of computational cost. Furthermore, the calculation of cross ratio proposed in [47] is not the standard way to calculate the cross-ratio.

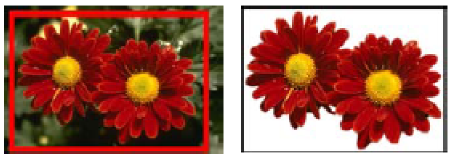


Figure 2.17: Grabcut example in [48]

2.4.3 GrabCut

Grabcut [48] is a method to segment an input image into a foreground and background parts (Fig.2.17). The method works by first creating Gaussian Mixture Model (GMM) of the color distribution of the object in the image specified by a user (through a bounding box surrounding the object). There are two GMMs generated, one for the background and another one for the foreground. Color information belonging to the part outside of the bounding box will be considered background. Finally, the segmentation process is done by minimizing the energy function (Gibbs Energy) through a graph-cut algorithm. Notice here that the hard segmentation part defined by the user is only the background part. At this point the unlabelled part is considered to be the ”foreground” and the energy minimization process will update the parameters of the GMMs, which eventually will produce the final foreground segmentation during convergence.

2.4.4 Interpolation Methods

Interpolation methods are used to predict the unknown value given several known values. Here we are going to discuss two interpolation methods: (generalized) bicubic interpolation method and bezier curve interpolation.

2.4.4.1 Generalized Bicubic Interpolation

In cubic spline interpolation [49], given a set of known points $P(x_i, y_i)$, we would like to get the parameters (coefficients) such that equation (2.1) holds.

$$S(x_i) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{2.1}$$

A spline interpolation has a characteristic that it has a smooth curve. In order to maintain its smooth curve characteristics, i.e. for each segment the first and second derivatives should be continuous, several conditions must be met:

- $S_i(x_i) = y_i$
- $S_i(x_{i+1}) = y_{i+1}$

- $S'_{i-1}(x_i) = S'_i(x_i)$
- $S''_{i-1}(x_i) = S''_i(x_i)$

The first and second condition ensure that the input points will be traversed by the spline. The third and fourth condition are the smoothness constraint of the spline. By applying these conditions to (2.1), we can get the following equations:

$$S_i(x_{i+1}) = y_{i+1} \tag{2.2}$$

$$S''_i(x_i) = 2b_i \tag{2.3}$$

$$S''_i(x_{i+1}) = 2b_{i+1} \tag{2.4}$$

Which, we can specify the following:

$$y_i = d_i \tag{2.5}$$

$$h_{i-1} = x_i - x_{i-1} \tag{2.6}$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - a_i h_i^2 - b_i h_i \tag{2.7}$$

$$a_i = \frac{b_{i+1} - b_i}{3h_i} \tag{2.8}$$

Combining these equations result in equation (2.9).

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{(b_{i+1} + 2b_i)h_i}{3} \tag{2.9}$$

Taking into account that $b_0 = b_n = 0$ (free ends, not clamped), where n = number of points -1, we can arrange the equation into matrix form to solve b as follows.

$$\begin{pmatrix} p_1 & h_1 & 0 & \cdots & 0 & 0 \\ h_1 & p_2 & h_2 & \cdots & 0 & 0 \\ 0 & h_2 & p_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & p_{n-2} & h_{n-2} \\ 0 & 0 & 0 & \cdots & h_{n-2} & p_{n-1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_{n-2} \\ q_{n-1} \end{pmatrix}$$

Having known the coefficients needed, we can then apply the cubic spline interpolation to a new x_i data.

Bicubic spline interpolation is a generalization of the cubic spline with two variables. A more generalize form is a generalized bicubic interpolation. In generalized bicubic interpolation, we need the following information:

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{i,j}x^i y^j \tag{2.10}$$

$$p_x(x,y) = \sum_{i=1}^{3}\sum_{j=0}^{3} a_{i,j}x^{i-1} y^j \tag{2.11}$$

$$p_y(x,y) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{i,j}x^i j y^{j-1} \tag{2.12}$$

$$p_{x,y}(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} ij x^{i-1} y^{j-1} \tag{2.13}$$

In order to solve (determining) the coefficient (a_{ij}), 16 equations created by inputing the corner value, i.e. $x, y \in [0..1]$ are needed.

$$f(0,0) = p(0,0) = a_{00} \tag{2.14}$$

$$f(1,0) = p(1,0) = a_{00} + a_{10} + a_{20} + a_{30} \tag{2.15}$$

$$f(0,1) = p(0,1) = a_{00} + a_{01} + a_{02} + a_{03} \tag{2.16}$$

$$f(1,1) = p(1,1) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} \tag{2.17}$$

$$f_x(0,0) = p_x(0,0) = a_{10} \tag{2.18}$$

$$f_x(1,0) = p_x(1,0) = a_{10} + 2a_{20} + 3a_{30} \tag{2.19}$$

$$f_x(0,1) = p_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13} \tag{2.20}$$

$$f_x(1,1) = p_x(1,1) = \sum_{i=1}^{3}\sum j = 0^3 a_{ij} i \tag{2.21}$$

$$f_y(0,0) = p_y(0,0) = a_{01} \tag{2.22}$$

$$f_y(1,0) = p_y(1,0) = a_{01} + a_{11} + a_{21} + a_{31} \tag{2.23}$$

$$f_y(0,1) = p_y(0,1) = a_{01} + 2a_{02} + 3a_{03} \tag{2.24}$$

$$f_y(1,1) = p_y(1,1) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} j \tag{2.25}$$

$$f_{xy}(0,0) = p_{xy}(0,0) = a_{11} \tag{2.26}$$

$$f_{xy}(1,0) = p_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31} \tag{2.27}$$

$$f_{xy}(0,1) = p_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13} \tag{2.28}$$

$$f_{xy}(1,1) = p_{xy}(1,1) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij}ij \tag{2.29}$$

Let α be the vector consists of the coefficients a_{ij} and F be the vector consists of the value of *f*. In matrix form, the 16 equations above can be written as (2.30), where M is the coefficient of a. Thus, by calculating the matrix multiplication of inverse matrix M^{-1}, we can get the values of a_{ij}.

$$M\alpha = F \tag{2.30}$$

2.4.4.2 Bezier Curve Interpolation

Similar with the cubic interpolation, given a set of points bezier curve interpolation tries to interpolate the curve among those points. While in typical interpolation where the curve generated will traversed the known points, in a bezier curve the curve generated will not intersect (traverse) those points with an exception of the first and the last point. These known points are called the control points and will determine the shape of the bezier curve generated. Given n+1 control points, a general bezier curve can be defined as in (2.31). b_i denotes the i-th control point and $B_{i,n}(t)$ is called the Bernstein polynomial defined in (2.32). An example of a quadratic bezier curve (3 control points) and cubic bezier curve (4 control points) can be seen in Fig.2.18.

$$B(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \tag{2.31}$$

$$B_{i,n}(t) = \begin{cases} \frac{n!}{(n-i)!i!}(1-t)^{n-i}t^i & \text{if } 0 \le i \le n \\ 0, & \text{otherwise} \end{cases} \tag{2.32}$$

2.4.5 Delaunay Triangulation

Given a set of (randomly placed) points, delaunay triangulation is a method to connect these points one with another to create a set of triangles, such that there are no other triangles lie inside the circumcircle of a triangle (Fig. 2.19). Notice in figure 2.19a, the triangle configuration is invalid since vertex B is inside the circumcircle of triangle ACD. In case where there is no possible solution, delaunay

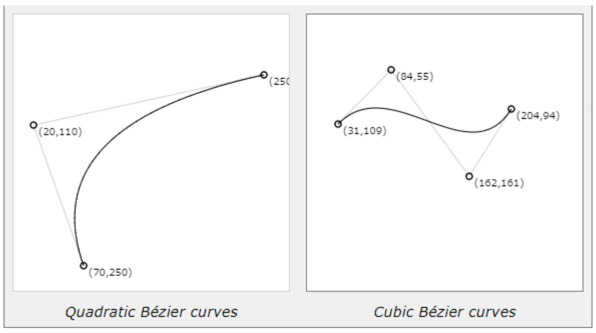
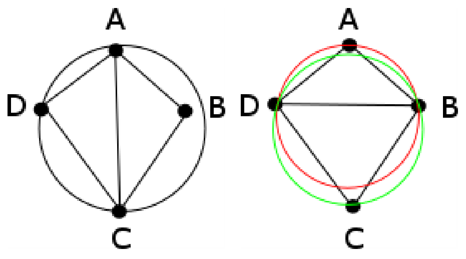


Figure 2.18: Bezier examples (image source: https://pomax.github.io/bezierinfo/)



(a) Invalid Delaunay

(b) Valid Delaunay

Figure 2.19: Delaunay Triangulation from a set of points

triangulation will minimize the total number of triangles lying in the circumcircle of another triangle. Notice that by avoiding creating a triangle inside a circumcircle of another triangle, a skinny triangle can be avoided. There are several algorithms that have been developed to solve the delaunay triangulation, such as divide and conquer, sweephull algorithm, etc.

In the divide and conquer algorithm, the basic idea is to divide the vertices into two sets recursively until a set consists only at most three vertices. From these sets of vertices, the algorithm then merged a set of vertices into another set until all set has been merged and finally the final triangulation can be acquired. During each merging process, the constraint that each triangle should avoid being generated inside another triangle circumcircle is kept. In the sweephull algorithm, the idea is to expand the seed (randomly picked triangle from the set of points) radially by adding the point to the previously generated triangle(s), such that the circumcircle constraint still be maintained. In the sweep-hull algorithm, a triangle flipping step is also incorporated to minimize the number of skinny triangle.

Chapter 3

Rigid 3D Model Retrieval using Skewness Map

In this chapter, we discuss the rigid 3D object retrieval case. The purpose of the system is to retrieve a 3D object which has the most similarity with the query from the database. One of the core problem in a retrieval system is how to match between the query and the database. In model-based systems, matching between the input and models in the database can be done relatively easy, since both of the models have the same dimensions, i.e. 3D features matched with 3D features.The problem is much more pronounce in the 3D-2D matching (view-based systems) since, depending on the viewing angle, the shape of an object might differ.

The simplest way to solve this is by having a full information regarding the object in all possible viewing angle and then do the matching step. This means that an object shall be represented by at most 360 $\times$ 360 images (129,600 images). If the database consists of 1000 objects, then the total images in the database will be 129,600,000 images. Matching a query with 129,600 images is of course a time consuming task. It is natural to ask then whether or not matching between the query and all the images in the database is required. As will be shown in the later section, the answer is no. Utilizing a skewness information from the object, we can reduce the search space of the matching step greatly. To further analyze the accuracy, the proposed method will then be applied to solve the orientation estimation (viewing angle estimation) of an object. Fig. 3.1 shows the pipeline of our proposal. To summarize, the purpose of this research can be considered to answer the following questions:

1. How fast is the proposed method in retrieving an image from the database compared to the conventional retrieval method (e.g. using SURF[50])?

2. Is there any other descriptor that can be used?

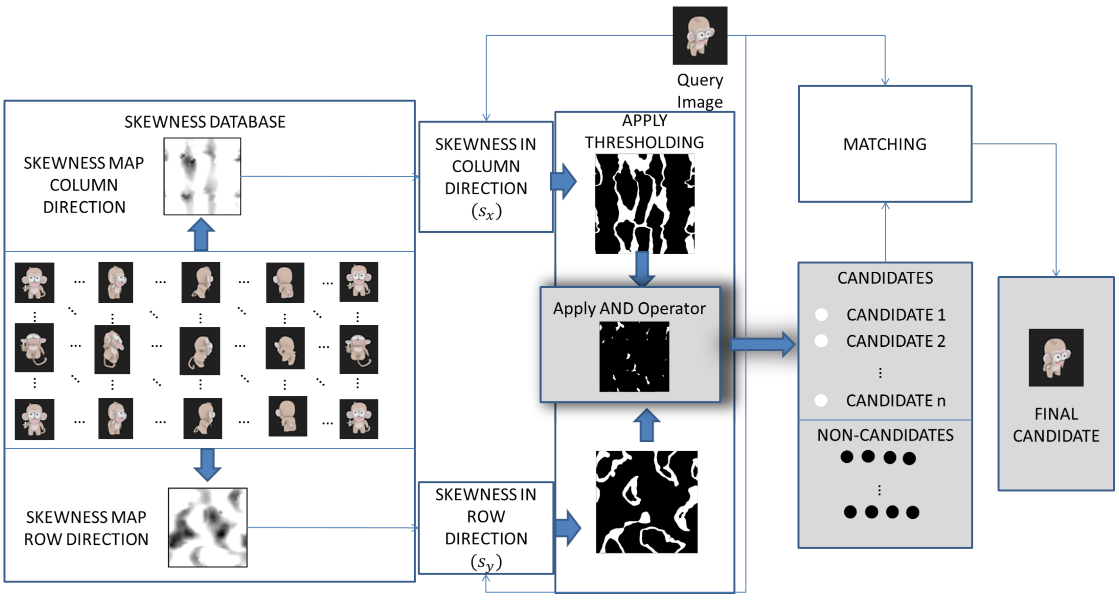


Figure 3.1: General overview of the proposed method

3. How accurate is the proposed method in estimating the orientation of the object?

The rest of this chapter is organized as follows: in the first section, a search space reduction utilizing the skewness map will be introduced. In the second section the matching step between the query image and the reference (database) will be presented. The details about the experiments done, the results and discussion will be presented on the third section of this chapter.

3.1 Search Space Reduction

In order to reduce the search space, we proposed a skewness map method. Before delving deeper into the skewness map, we will present the background information regarding skewness and how it relates to the objects.

3.1.1 Skewness

In probability theory, skewness is a method to measure the symmetricity of a distribution. A distribution with a zero valued skewness will be regarded as symmetrical, while a positive (negative) skewness value will be regarded as right-skewed (left-skewed). A genereal way to define skewness is by using the third order moment of the distribution. Consider a distribution with two variables x and y. Eq.(3.1) defines the central moment calculation, which can be used to calculate the skewness value.

$$m_{ji} = \iint (x - \bar{x})^j (y - \bar{y})^i f(x, y) dx dy \tag{3.1}$$

The subscripts ji denote the order of the moment in x and y spaces respectively. Note that Eq.3.1 needs the average values (center mass) of x and y, which are denoted by $\bar{x}$ and $\bar{y}$. Both of these values can be calculated using the raw moment equations defined in Eqs.(3.2)-(3.3). M_{00} denotes the area of the object.

$$\bar{x} = \frac{M_{10}}{M_{00}} \tag{3.2}$$

$$\bar{y} = \frac{M_{01}}{M_{00}} \tag{3.3}$$

$$M_{pq} = \iint x^p y^q f(x, y) dx dy \tag{3.4}$$

Since a calculation which can be used to calculate the skewness of a distribution has been defined, one might ask what the relation between skewness and an image of an object is. Given an image of an object shown in the left-most part of Fig. 3.2, we can binarize the image by first finding the outer contour of the object. After acquiring the outer contour information, each pixel will then be labelled depending on its position to the contour points. A pixel lying inside the contour will be given a label of one (visualize as white in Fig. 3.2) and zero otherwise (visualize as black pixel). Finally we can calculate the sum of the ones (white pixels) at each column or row respectively to generate a distribution as shown in the right-most part of Fig. 3.2. Note that in Fig. 3.2 we only show the column-wise summation of the white pixels.

Note that the equation (3.1) is not invariant to scaling, therefore, instead of using equation (3.1) as it is, we use the normalized moments instead (Eq.(3.5)). Calculating this distribution using the normalized moment will give us a unique value for its distribution. Since this distribution is symmetric, the skewness value will be zero (or very near to zero).

$$n_{ji} = \frac{mji}{m_{00}^{\frac{(i+j)}{2}+1}} \tag{3.5}$$

Assume that the horizontal axis is the x axis, the axis which protrudes from the paper is the z axis and the vertical axis is the y axis. Fig. 3.3 is the result of rotating the same object in Fig. 3.2 by 30 degrees with respect to the y axis. Doing the same binarization process mentioned beforehand, we can get another distribution as shown in the right-hand most part of Fig. 3.3. As can be seen here, the distribution does not maintain its symmetricity. From this conclusion, we

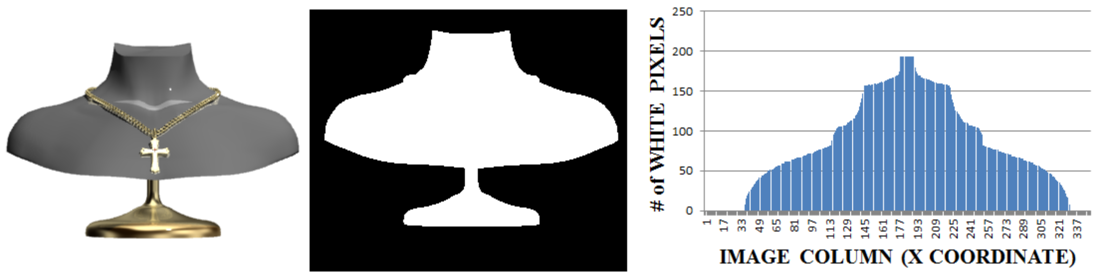


Figure 3.2: Binarization Process

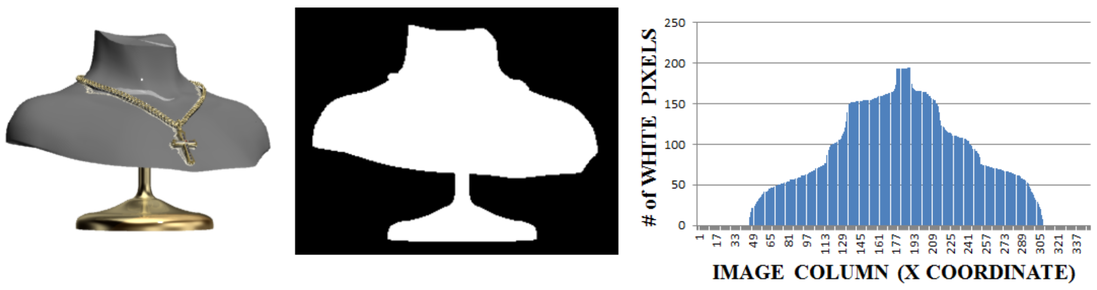


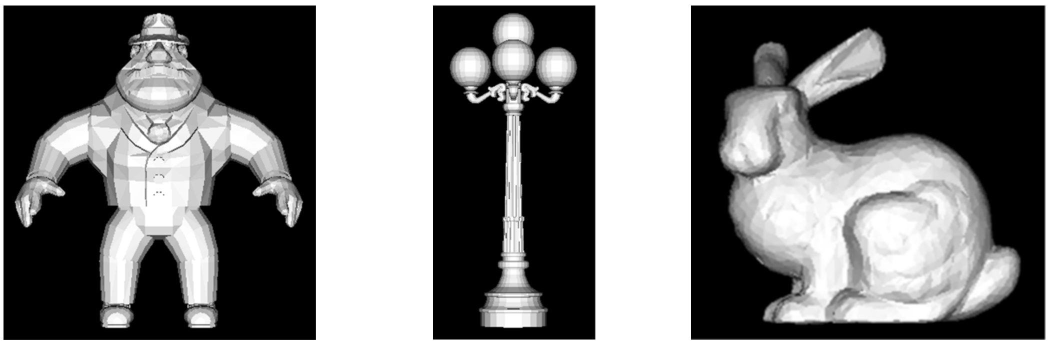
Figure 3.3: Applying rotation to object in Fig. 3.2

believe that by utilizing this skewness information of an object distribution, given a set of known symmetry values, the orientation of an object can be approximated. Not only can we approximate the orientation of an object, but also by utilizing this skewness information, a search space reduction method namely skewness map can be applied. The following subsection will describe the skewness map proposal in more details.

3.1.2 Skewness Map

A skewness map is a collection of skewness value of a given object. Each point (or pixel) in a skewness map represents the skewness value of an object. The position of the pixel represents the orientation of the object. For example, an object viewed from polar angle ((xz)y-plane) $\theta = 30°$ and azimuth angle $\phi = 45°$ will be represented by a pixel at (x,y) = (30,45) coordinates on the skewness map. Ideally, a skewness map should be generated by calculating the skewness value of every possible viewing angle of an object, but of course this will take a lot of time and resources. In order to relax this condition, an interpolation method is used.

There are several known interpolation methods that can be used. In this research the generalize bicubic interpolation is chosen. While bicubic spline is the generalization of cubic spline [49] in two dimensions, it is also a special case of a



(a) Al (b) Lamp (c) Stanford Rabbit

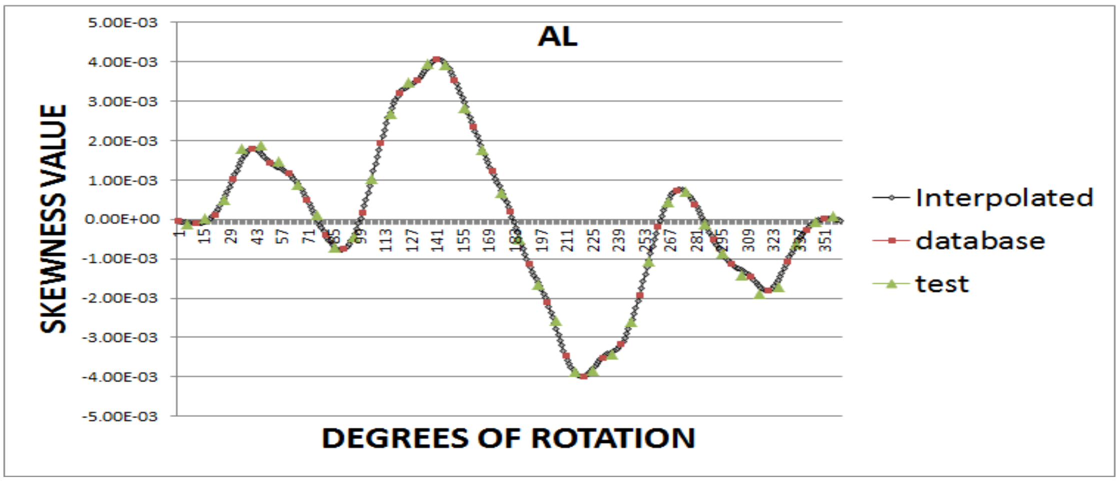
Figure 3.4: Test objects

generalized bicubic interpolation. Bicubic spline interpolation has two properties which are suited for skewness interpolation, i.e. it has a smooth continuous curve and maintains the original value at the sample points (node). Consider objects given in Fig. 3.4. If a (azimuthal) rotation operation is applied to each object and the skewness values are plotted, a graph created using the cubic spline interpolation as in Fig. 3.5 will be acquired. As can be seen from the Fig., rotational operation will make a smooth changes to the skewness value. Also note that in order to make an interpolation, a ground truth value of the object at the sampling points are taken. Therefore at the exact sampling point we know the skewness value of the respective object. Since the bicubic interpolation has a property of maintaining its value at the start and end points and has a more flexibility, it has a better interpolation result compared to other interpolation methods, such as linear interpolation.

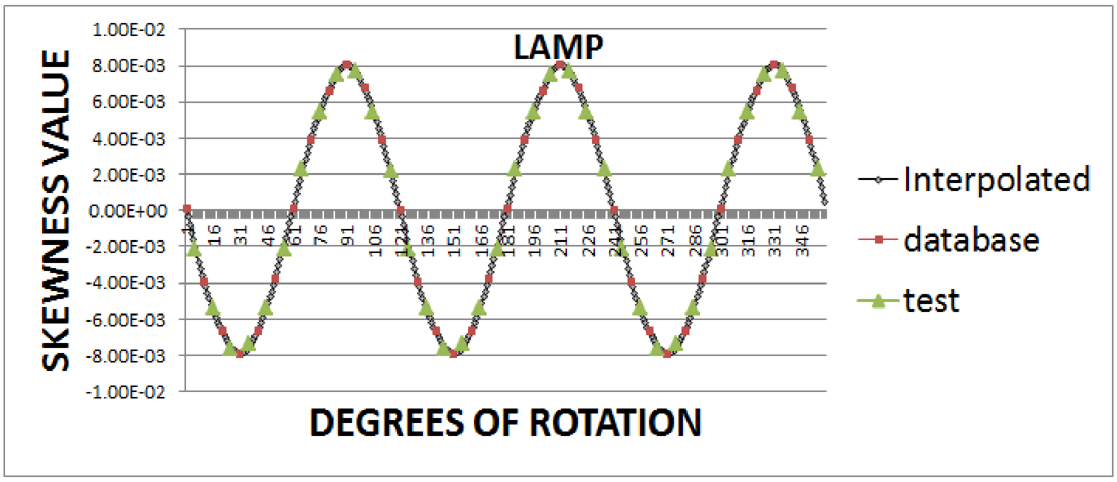
Recall that an image will have two types of distribution, i.e. column-wise and row-wise distributions, therefore there will also be two skewness maps to describe an object. Fig. 3.6 shows examples of the skewness maps. The next subsection will discuss how the skewness map will be used to reduce the search space during the matching step.

3.1.3 Skewness Maps as Search Space Reduction

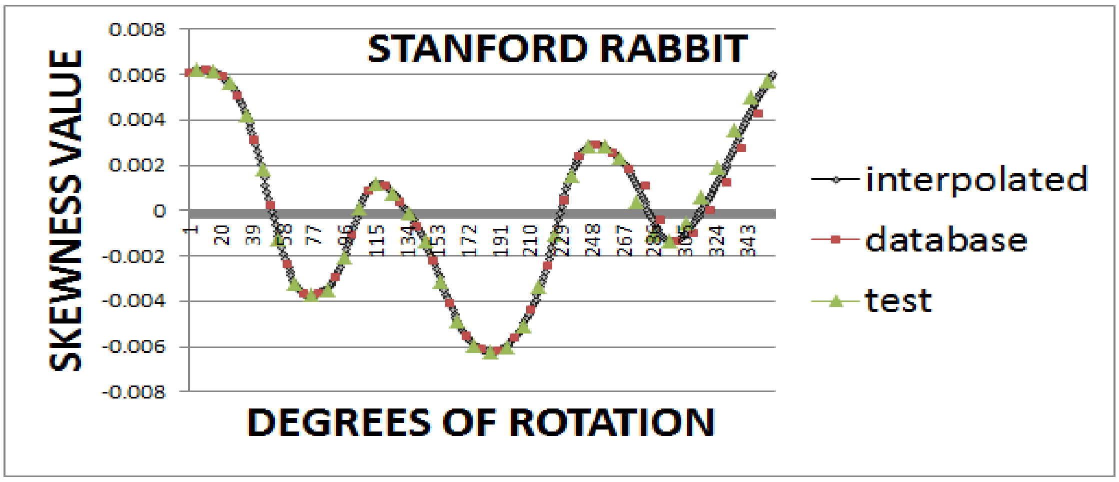
A visualization of this approach can be seen in Fig.3.8. Given a query object, using equation (3.5) with $i, j = \{0, 3|i \neq j\}$, both the column-wise and row-wise skewness values can be calculated. Let S_x denotes the column-wise skewness value and S_y denotes the row-wise skewness value. These values will be the threshold to reduce the search space. Given skewness maps of an object, all the pixels value will be compared with the calculated skewness value. If a pixel has a value beyond (less or higher than) the threshold value, then it will be neglected dyrubg the matching



(a) Al skewness plot

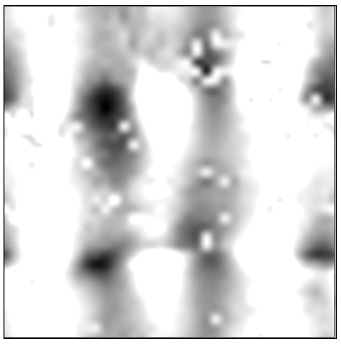


(b) Lamp skewness plot



(c) Stanford Rabbit skewness plot

Figure 3.5: Skewness plot results


(a) Column-wise skewness map

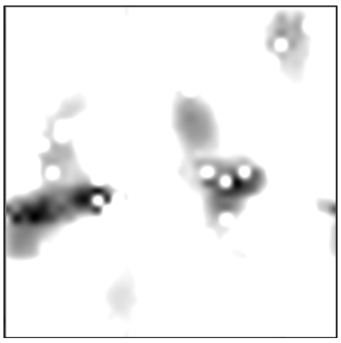
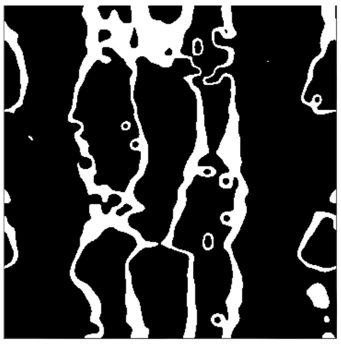

(b) Row-wise skewness map

Figure 3.6: Skewness maps example


(a) column-wise

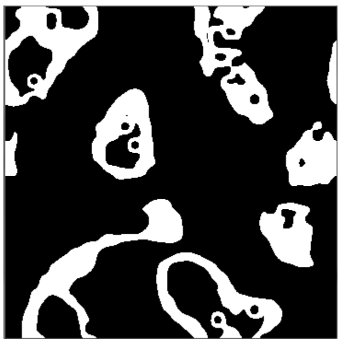

(b) row-wise

Figure 3.7: Thresholded maps

step. Otherwise, it will be kept and will be considered during the matching step. Recall that each pixel at the skewness map represents the object (image) viewed at certain angles.

Mathematically equation (3.6) defined the aforementioned selection (thresholding) step and Fig. 3.7 visualizes the thresholding operation. $I_x(x, y)$ denotes the new pixel value at pixel (x,y) of the column (subscript x) thresholded map. $I_y(x, y)$ denotes the new pixel value at pixel (x,y) of the row (subscript y) thresholded map. $f_x(x, y)$ and $f_y(x, y)$ denote the original pixel value at the skewness maps. ε denotes the user-defined constant error value.

$$\begin{aligned} I_x(x,y) &= \begin{cases} 1, & \text{if } (s_x - \varepsilon) \leq f_x(x,y) \leq (s_x + \varepsilon) \\ 0, & \text{otherwise} \end{cases} \\ I_y(x,y) &= \begin{cases} 1, & \text{if } (s_y - \varepsilon) \leq f_y(x,y) \leq (s_y + \varepsilon) \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.6}$$

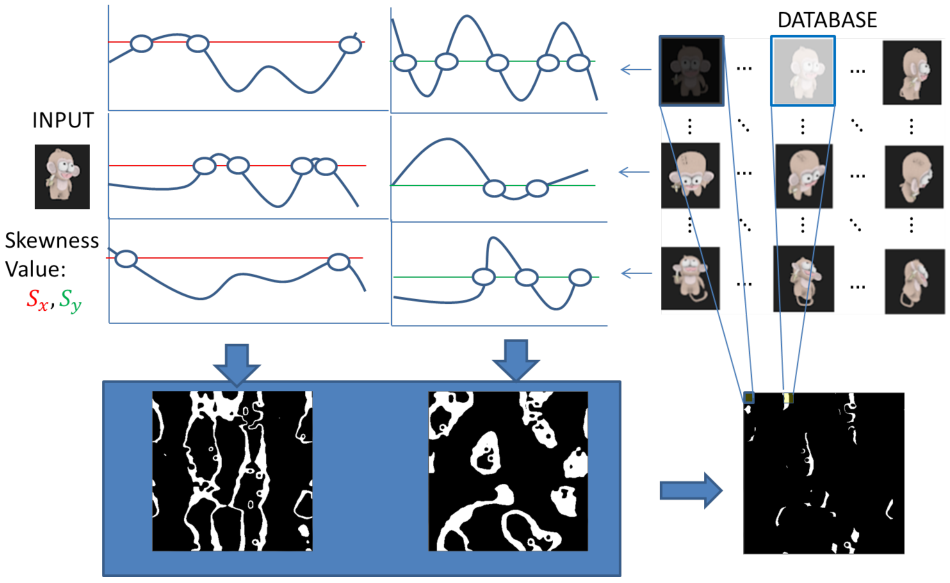


Figure 3.8: Visualization of search space reduction

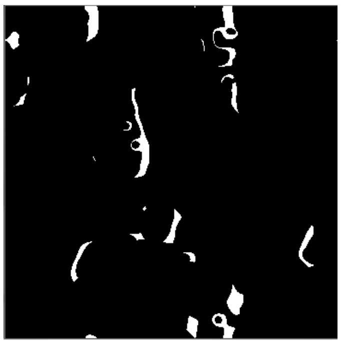


Figure 3.9: Final search space (”AND” Map)

Finally, given the two new thresholded maps, we can combine both of the maps using a simple AND operator as in equation (3.7). $I_{xy}(x, y)$ denotes the pixel value at the ”AND” map. Fig. 3.9 visualize the final search space for the matching step. Only the pixel which still remains after the logical AND process (visualize as white pixels) will go through a matching process with the query. As can be seen most of the images in the database can be neglected using the skewness maps.

$$I_{xy} = \begin{cases} 1, & \text{if } (I_x \wedge I_y) \\ 0, & \text{otherwise} \end{cases} s \tag{3.7}$$

3.2 Matching Step using Cross Ratio-like Number

Matching between the input image and the images on the database is one of the most important task. One way to calculate the similarity between the images is by using the image feature. Therefore, the feature extraction method is also a crucial part in this matching step[51]. In this section, we will discuss several image features (along with a matching step for each feature) that can be used to help match between the input and the reference images in the database.

As described in the beginning of chapter 2, SURF might have a drawback of retrieving an incorrect result given an object with less texture. Recently, a descriptor based on the cross ratio has been gaining popularity to solve this problem. In light of this, we propose a cross-ratio like number to be used as feature descriptor as follow. A cross ratio is usually defined as a ratio between 4 collinear points in a projective geometry. Given a collinear points denoted by A, B, C, and D (Fig.2.15), the cross ratio (CR) betwen those points can be defined as in (3.8). The overbar denotes the length between two points. Notice here that the order will determine the outcome of the equation, i.e. CR(A,B,C,D) is not the same as CR(A,C,B,D).

$$CR(A,B,C,D) = \frac{\overline{AC}}{\overline{BC}} \cdot \frac{\overline{BD}}{\overline{AD}} \tag{3.8}$$

In order to calculate the CRN feature, first of all an edge image of the object shall be created. This can be done simply by using Canny edge detector[52] or any other edge extraction algorithm. A convex hull from the edge image is then generated. Afterwards, a set of n points with equal spacing are generated on the convex hull. A set of line connecting these points withe one and another is generated and the cross ratio along these lines will be used as a feature.

Given n points on the convex hull, at most there will be $N =_n C_2$ point pairs. Among these lines, there will be lines which intersect the edges of the object as shown in Fig.3.10. On such lines, the cros ratio will then be calculated. If a line consists only three points including the hull points, then it will be regarded as a feature point with zero value (zero cross ratio). In [47] the authors proposed a multiple-line combination to generate the CN feature. Unfortunately multiple-line combination is computationally expensive and therefore we use only one line with d-number of cross ratios here. d can also be considered as the number of dimensional descriptor pre-defined by the user.

For an example, consider the line, two hull points (small circle on the convex hull) and the four intersection points in Fig.3.10. In this line example, we can generate a ($D =_4 C_2 =$) 6-dimensional descriptor to describe the cross ratios on

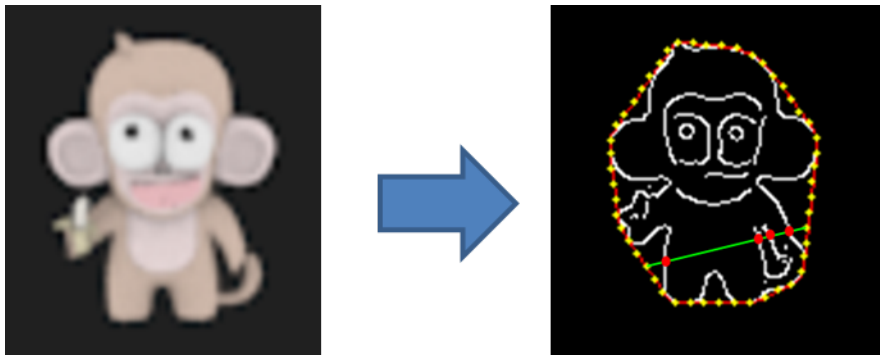


Figure 3.10: CRN feature visualization

this line. If $D \leq d$, the rest of it $(d - D)$ will be considered as zero-valued feature vector's element. If $D > d$, we will truncate the feature vector's element, i.e. only the first d cross ratios will be used. In total, assuming that the number of points $n = 25$ is used, we will have at most $(N =_{25} C_2 =)$ 300 descriptors, where each descriptor consists of d-cross ratios. Let O be the object, $(P_i)_d$ denotes the i-th line each with d number of cross ratios, m is the number of points $(m \leq N)$, the CRN is the set that encapsulates these features and defined in (3.9).

$$CRN(O) = \{(P_1)_d, (P_2)_d, \cdots, (P_m)_d\} \tag{3.9}$$

Cross matching is utilized here for the feature matching step. Let $\overline{A}$ and $\overline{B}$ each denote a feature vector of the query and the reference respectively. For every element in A an L1-norm distance between the point and all elements in B will be calculated (3.10).

$$dist(\overline{A}, \overline{B}) = \sum_{k=0}^{k=d-1} |a_k - b_k| \tag{3.10}$$

Let $(X,\ Y)$ be the feature point pair with the minimum distance of two objects 1 and 2 ($X \in CRN(O = 1)$ and $Y \in CRN(O = 2)$) from X point of view. The cross matching is then utilized by switching the feature point reference, i.e. Y is compared with all the feature points in *CRN(O = 1)*. Let *(Y, X')* be the minimum distance from Y point of view. If *X'* is the same point as *X*, then the point pair is regarded as matching point.

In order to assess whether the query is similar with an object in the database, the sum of these matching pairs (i.e. matching points) is calculated. Object with the highest total similarity number will be considered to be matching with each other (hence, the term Cross-Ratio-like Numbers). Notice here the difference of inference between the FLANN and cross-matching criteria.

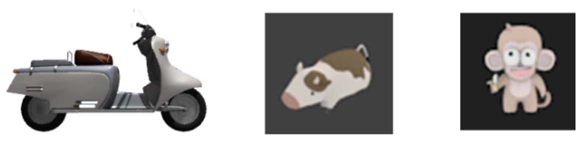


Figure 3.11: 3D objects used in accuracy test

3.3 Experimental Result and Discussion

Several experiments were done in order to assess the proposed method. All of the experiments were done using C++ language in an Intel i5, 4GB RAM, 3.20 GHz, Windows 7 OS computer using OpenCV 2.49 as the image processing library. GrabCut[48] algorithm was used in order to do the foreground-background extraction. Each image used in the experiment has 150 $\times$ 150 resolutions. One object database consists of 129,600 artificial object (rendered 3D model) images, taken at every possible orientation. The thresholding parameter ε was set to 10% of the s values.

3.3.1 Accuracy comparison

The first test is to assess the accuracy of the proposed method. There are two types of accuracy that were considered here. The first one is whether the method retrieve the same object correctly and the second one is whether the orientation of the object can be approximated correctly.Three test objects were used in this experiment as shown in Fig.3.11. Five orientations were selected randomly from the database for every object.

First the retrieval method based on Magic Canvas [18] was tested and the result is shown in Table 3.1. x denotes the rotation (in degrees) with respect to the x-axis and y denotes the rotation with respect to y-axis. NA means the incorrect object was retrieved as the result of the algorithm. There are several reason why we choose to compare our proposal with Magic Canvas. The first one is most of the methods mentioned in rigid retrieval in chapter 2 use two or more inputs to the system, while our method and Shin's method use only one input to the system. Second, both of them rely on the contour information. Third, most of the previous method used a learning step in one of their proposal, while our proposal and Shin's do not require any step at all.

Table 3.2 shows the result of the proposed method using the Skewness Map combined with SURF (SMSURF) and CRN (SMCRN) respectively. As can be seen from Table 3.2 the proposed method can retrieve the object correctly. Furthermore the second type of accuracy, i.e. approximating the orientation of the object,

Table 3.1: Comparison Result: Magic Canvas(MC) vs Ground Truth (GT)

Query Image	GT		MC			
			Result		Error	
	x	y	x	y	x	y
Bike1	176	349	NA		NA	
Bike2	289	301	NA		NA	
Bike3	318	319	180	351	138	32
Bike4	142	43	0	191	142	148
Bike5	72	255	70	141	2	114
Cow1	129	239	NA		NA	
Cow2	104	329	30	11	74	318
Cow3	224	263	NA		NA	
Cow4	89	53	30	61	59	8
Cow5	125	314	NA		NA	
Monkey1	220	159	NA		NA	
Monkey2	14	125	NA		NA	
Monkey3	348	126	NA		NA	
Monkey4	119	214	90	101	29	113
Monkey5	118	34	110	121	8	87

generated only a really small error (less than 2 degrees error in average).

Notice here that we do not show the result of the accuracy of SURF. This is because SURF will give us 100% accuracy since the query is taken from the database. Seeing the results, one might ask then whether the proposal actually decrease the accuracy of SURF in general. Statistically, the proposed method and conventional SURF method generate the same performance. The multivariate one-sample t-test (Hotelling's t-square test 3.11) is used to prove the previous statement. $\bar{X}$ denotes a vector (mean vector) which every element is the mean value of the variables, S^{-1} is the inverse covariance matrix. k denotes the number of variable and n denotes the number of samples. As mentioned before, SURF generates zero errors, therefore $\mu_0 = 0$. For small number of n, the value T^2 can be calculated by using the F distribution (Eq.3.12). We were using $\alpha = 0.05$. Table 3.3 shows the result of the statistical analysis. Note that since the F-values are smaller than the F-critical values, we can not reject the Null-hypotheses and therefore there is no significant difference between the result of conventional SURF which gave zero errors and our proposals.

$$T^2 = n\left(\bar{X} - \mu_0\right) S^{-1} \left(\bar{X} - \mu_0\right) \tag{3.11}$$

Table 3.2: Comparison Result: Proposed vs Ground Truth (GT)

Query Image	GT		SMSURF				SMCRN			
			Result		Error		Result		Error	
	x	y	x	y	x	y	x	y	x	y
Bike1	176	349	176	349	0	0	176	349	0	0
Bike2	289	301	289	301	0	0	289	301	0	0
Bike3	318	319	318	319	0	0	318	319	0	0
Bike4	142	43	142	43	0	0	142	43	0	0
Bike5	72	255	71	258	0	3	67	257	5	2
Cow1	129	239	129	239	0	0	129	239	0	0
Cow2	104	329	104	329	0	0	104	329	0	0
Cow3	224	263	224	263	0	0	224	263	0	0
Cow4	89	53	89	53	0	0	89	53	0	0
Cow5	125	314	125	314	0	0	125	314	0	0
Monkey1	220	159	220	159	0	0	220	159	0	0
Monkey2	14	125	15	125	1	0	21	126	7	1
Monkey3	348	126	355	124	7	2	357	123	9	3
Monkey4	119	214	115	206	4	8	115	204	4	10
Monkey5	118	34	118	34	0	0	118	34	0	0
	Average				0.867	0.867			1.67	1.067
	RMSE				4.467	5.133			11.4	7.6

$$F = \frac{n-k}{k(n-1)} T^2 \sim F(k, n-k) \tag{3.12}$$

3.3.2 Time Comparison

The second experiment is to compare the time comparison achieved by the proposed method and the conventional SURF. Table 3.4 shows the result of the experiment using the same data as the accuracy test. By average, the proposed method (SMSURF) is much faster, i.e. the total time is reduced 75% and for SMCRn it is reduced further around 91%. In order to assess the validity of these measurements, a statistical test using one sample t-test is performed. Again $\alpha = 0.05$ is used here. $\mu_{d,P}$ is defined as $\mu_{d,P} = \mu_{SURF} - \mu_P$, where the subscript P denotes the proposal methods respectively, i.e. SMSURF or SMCRN. Table 3.5 shows the result of this test.

Since the p-values are less than $\alpha = 0.05$, we can reject the null hypotheses, which mean conventional SURF significantly has higher computational cost

Table 3.3: Result comparison using Hotelling's t-square test

Hypothesis	n	k	T^2	F	*F-critical*
$H_0 : \mu_{SMSURF} = 0$ $H_1 : \mu_{SMSURF} \neq 0$	15	2	3.239552	1.504078	3.805565
$H_0 : \mu_{SMCRN} = 0$ $H_1 : \mu_{SMCRN} \neq 0$	15	2	4.825895	2.240594	3.805565

Table 3.4: Time Comparison of Conventional SURF, SMSURF and SMCRN (in seconds)

Query Image	Total Time		
	SMCRN	SMSURF	SURF
Bike 1	115.572	340.227	1632.57
Bike 2	119.178	416.308	2466.83
Bike 3	147.19	433.35	1759.32
Bike 4	171.239	516.879	2170.32
Bike 5	207.645	532.747	1271.2
Cow 1	130.661	376.933	1270.6
Cow 2	91.935	277.637	1538.65
Cow 3	177.126	466.511	1228.96
Cow 4	116.435	376.661	2045.69
Cow 5	124.088	349.275	1202.16
Monkey 1	82.181	258.934	1411.03
Monkey 2	135.442	336.955	934.219
Monkey 3	155.161	410.519	1165.35
Monkey 4	56.923	161.814	1553.25
Monkey 5	166.119	438.426	1136.19
Average	133.126	379.545	1519.089

compared to both of the proposed methods. By these results, we believe that our proposal using the skewness map can reduce the time needed to retrieve the object from the database faster than the conventional SURF method.

3.3.3 Larger Database

In this experiment, a larger database is used to compare the result of SMSURF and SMCRN. Fig.3.12 shows the objects used in this experiment. 30 models were used. Each model in the database were rendered from every possible viewing angles, i.e.

Table 3.5: t-Test for Time Comparison Result

Hypothesis	t-value	p-value
$H_0 : \mu_{d,SMSURF} = 0$ $H_1 : \mu_{d,SMSURF} > 0$	10.254	3.422×10^{-8}
$H_0 : \mu_{d,SMCRN} = 0$ $H_1 ; \mu_{d,SMCRN} > 0$	12.019	3.809×10^{-9}

Table 3.6: Summary of Large Database Experiment

Method	Error		RMSE		Time (in seconds)
	x	y	x	y	
SMSURF	4.133	0.211	36.173	0.367	4765.556
SMCRN	7.856	0.8	41.049	3.197	1402.428

$(360 \times 360 =)$129,600 images for one model. For each object, three samples were taken randomly. Therefore, in total there were 90 images tested and used as the input image. The full comparison data can be accessed in Table A.1 and Table A.2 in appendix section. Table 3.6 summarize the results. As can be seen, SMCRN has lower accuracy but performs faster than the SMSURF. The reason is becaue CRN uses less vector element for its descriptor. Of course this also depends on the parameter of the CRN, which leads us to the next part.

3.3.4 CRN Parameters

There are two parameters that need to be defined by the user, i.e. the number of points on the convex hull (n) and the size of descriptor (d). To determine these parameters, a simple test was performed. Several objects are taken from the database (Fig.3.12) One query and two test comparison objects will be used for each test. One of the two comparison objects is taken from the same object with the query but with different viewing angle. All of the objects are chosen manually, such that either the appearance of the object is similar or have a very minimum texture information.

Table (3.7) and Table (3.8) show the results of the experiment. The bold numbers highlight the highest similar number, i.e. the retrieved object. For exxample, with $n = 10$ and $d = 5$, with the query of monkey object, the retrieved object will be monkey also since it has higher similarity number compared to oni. We are trying to find the minimum requirement which has not only less number of descriptor but also a good accuracy. From the results shown, we then defined the

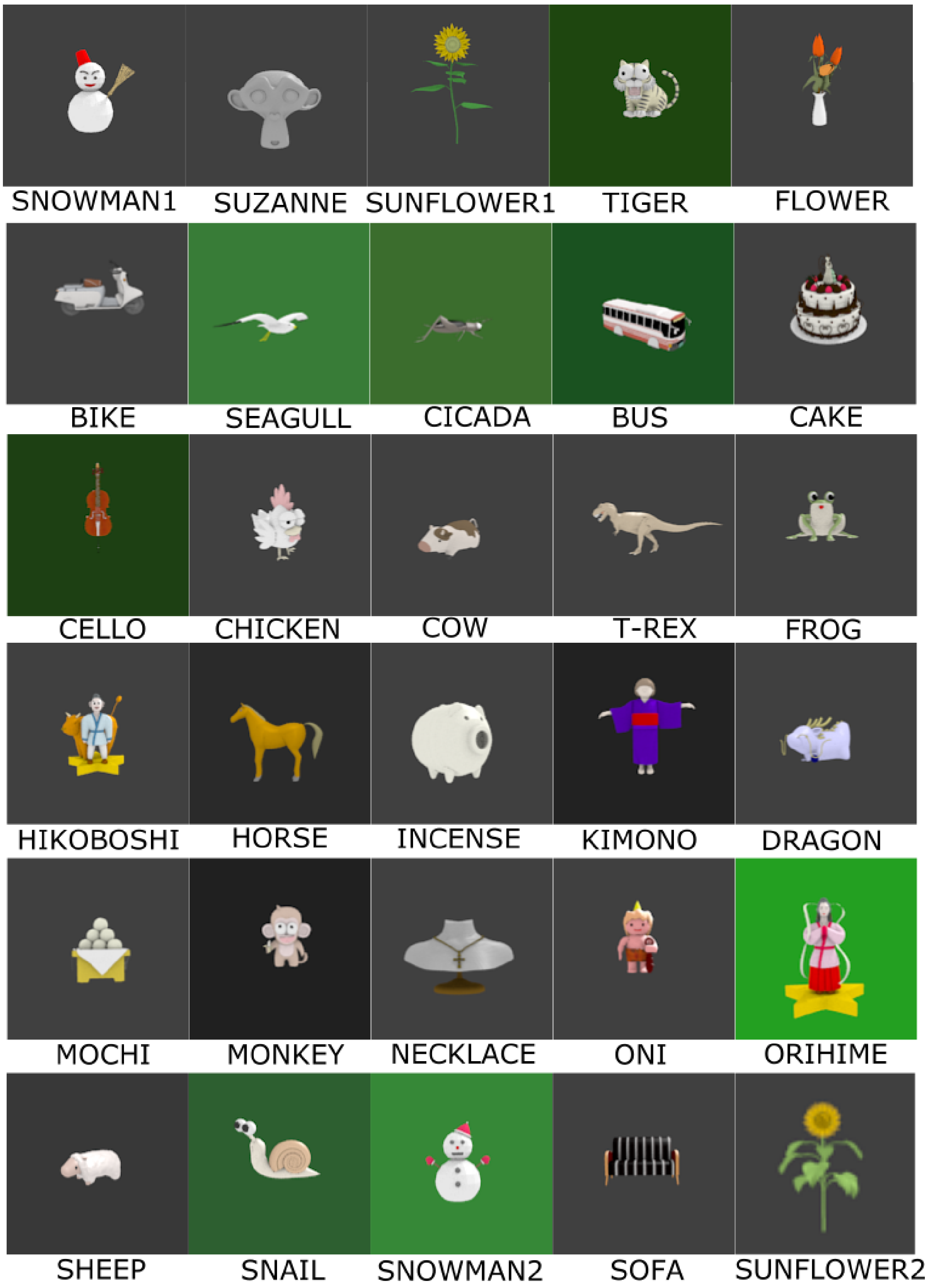


Figure 3.12: 3D Models used in the experiment in section 3.3.3

Table 3.7: Parameter testing of CRN feature generation ($d = 5$)

Query	Comparison	Similarity Number		
		$n = 10$	$n = 20$	$n = 30$
Monkey(0,1)	Monkey(0,10)	**19**	**77**	146
	Oni (0,1)	18	65	**151**
Minidragon(0,45)	Minidragon(0,10)	**15**	**63**	**133**
	Monkey(0,10)	14	54	99
Suzanne(0,1)	Suzanne(0,10)	**12**	27	**77**
	Snowman1(0,10)	7	**35**	51
Snowman1(0,1)	Snowman1(10,1)	**23**	**81**	**155**
	Snowman2(0,1)	8	48	102
Bike(232,69)	Bike(230,71)	**25**	**93**	**184**
	Tiger(0, 341)	20	67	133
	Accuracy	1.0	0.8	0.8

Table 3.8: Parameter testing of CRN feature generation ($d = 10$)

Query	Comparison	Similarity Number		
		$n = 10$	$n = 20$	$n = 30$
Monkey(0,1)	Monkey(0,10)	14	76	142
	Oni (0,1)	**17**	46	115
Minidragon(0,45)	Minidragon(0,10)	12	**54**	**95**
	Monkey(0,10)	**15**	40	84
Suzanne(0,1)	Suzanne(0,10)	**8**	**23**	**45**
	Snowman1(0,10)	6	22	44
Snowman1(0,1)	Snowman1(10,1)	**12**	**62**	**126**
	Snowman2(0,1)	6	41	77
Bike(232,69)	Bike(230,71)	**24**	**81**	**173**
	Tiger(0, 341)	13	56	112
	Accuracy	0.6	1.0	1.0

CRN parameters to be $d = 5$ and $n = 10$.

Chapter 4

Natural Image Skeletonization using Delaunay Triangulation

Skeletonization or skeleton extraction has been used in many area in image processing and computer vision. Although used mainly for animation [53, 54], several other uses of skeleton information are including symmetry detection [32], sensor networks [32]. A more interesting application of application has been proposed in 2D to 3D model creation [55, 56, 57]. Several authors also had already published works using skeleton information in shape analysis, which also includes the retrieval system for both rigid and non-rigid object [58, 25, 59, 60, 61].

Conventional skeletonization relies heavily on a closed-curved boundary object. In another term, this means that the binary image is needed as the input to the skeletonization process. Unfortunately, the closed curved constraint might not be achieved in an image due to some occlusion or the similarity of the object characteristics, such as colors, with the background.

To overcome such problem, grey-scale (natural image) skeletonization came into development. Instead of using a binary image as an input, grey-scale skeletonization use a grey-scale value of the image (or even the color image) as the input for the skeleton extraction method. As mentioned on the first chapter, current grey-scale skeletonization also relies on a semi-closed curve boundary and does not work well for the unconnected (sparse) points. In this chapter, we are trying to solve the grey-scale skeletonization with the sparse points condition.

Let it be a conventional skeletonization or the grey-scale skeletonization, object boundary still plays a major role in determining the result of the skeletonization. Therefore, the first step in our approach is to locate and identify the edge points on the image (the first section). Following this, a novel method to extract the skeleton extraction from these points will be presented on the second section. Finally, the third section consists the result of the experiment and some discussion.

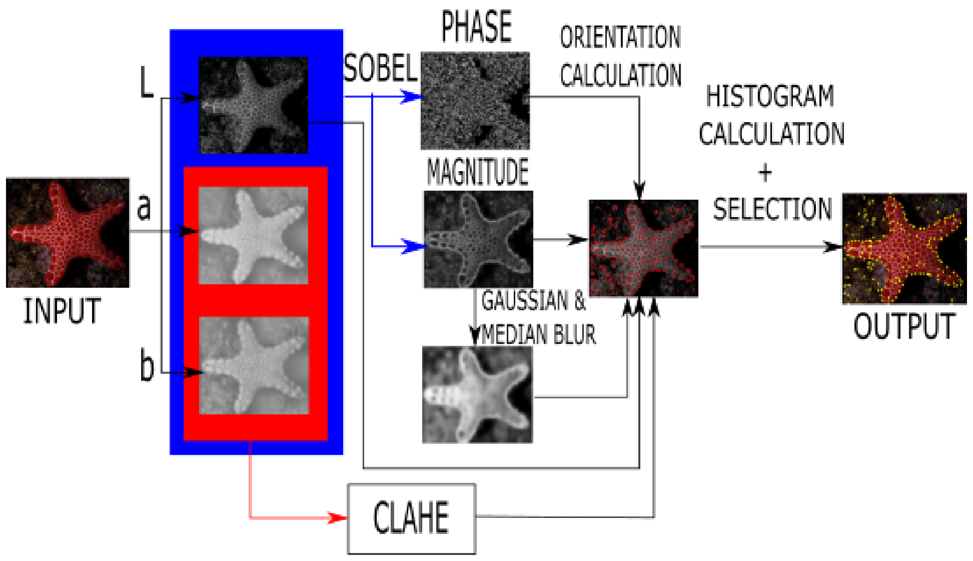


Figure 4.1: Edge extraction step

4.1 Edge Points Extraction

Although Canny edge detection performs relatively well to a general type of image, we avoid using it since Canny operator mainly depends only on the brightness image. As mentioned before, relying only on the brightness image might give a severe loss in the edge points extraction. Instead the following step is used (Fig.4.1). Note that the blue rectangle means that L, a, and b images are sent to the Sobel operation, while the red rectangle means that only the color images (a and b) are sent to the histogram equalization (CLAHE) step.

First, we convert the image into the CIELab space and split the image into each respective channels (L, a, and b). On each channel, we apply a 3×3 Sobel operator for horizontal and vertical direction. There are two benefits of using a Sobel operator. Not only the gradient edge image can be generated, the orientation (of the gradient) can also be found. More of this regarding the orientation will be described later.

By calculating the magnitude of both the images applied with the Sobel operator, a gradient image(s) can be acquired. Clearly, since we are working in L, a and b images, there will be three gradient images. We combine these three gradient images into one single image I by using (4.1). Here, $I(i,j)$ denotes the combined gradient images at $(i{,}j)$ pixel location, while the subscript $(L{,}a{,}b)$ denote each channel respectively. Note that the pixel's value at each channel is normalized before applying (4.1).

$$I(i,j) = max(I_L(i,j), I_a(i,j), I_b(i,j)) \tag{4.1}$$

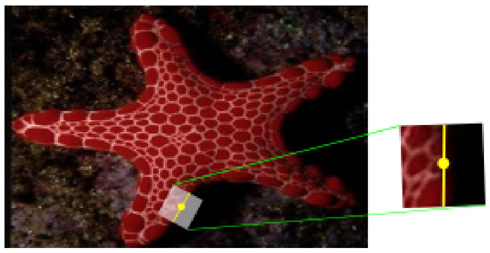


Figure 4.2: Along an orientation, edge point will have a large χ^2 distance between the left part and the right part of the patch.

On the combined gradient image (I), we divide the image using a grid (patch) of n×n, which we set $n = 5$ in this paper. On each grid, we search pixel with the highest intensity. Let these points be the candidate points. To reduce the number of the candidate points and using the fact that a pixel with high intensity value on I indicates higher probability of being an edge point (strong edge point), the first selection using k-means is applied. In our implementation the number of k is set to be 3 and two clusters with the highest centroid values (position) are sent to the next selection step.

A similar approach as in [62, 63] is adopted further to select the edge points from the candidate points. The idea of the method is to create a patch with the point as the center and divide the patch into two halves along the orientation (Fig.4.2). Then, histograms created at each part of the patch (first half and the second half) is compared by calculating their χ^2 distance (4.2).

$$\chi^2_p(l,r) = \sum_{i=0}^{N} \frac{(l-r)^2}{l+r} \tag{4.2}$$

Here l denotes the left part of the patch and r denotes the right part respectively. The higher the difference is the more likely it is to be considered as the edge points. Note that subscript p denotes the input image used for calculation. This means that each candidate point can be represented by p-number of vectors as features. These features will then be sent to the k-means algorithm to be clustered. Similar with the previous step, we use $k = 3$ and choose the points belonging to two clusters which centroid have the highest coordinate position.

As noticed, some points lie in a neighborhood of each other, these points will give artifacts to the end result and therefore to hinder such results, a merging operation is applied. The merging process is as follows: for each point check its vicinity in R radius. If there is (are) point(s) lying inside R, those points shall be merged to generate a new point. This new point coordinate is the average coordinate of all the points to be merged.

Before moving on the next part, there are several things that needs to be addressed, i.e. the orientation at each point and the images used in the histogram calculation.

4.1.1 Orientation Approximation

To approximate the orientation at each point, recall the gradient image generation step. The orientation at each pixel can be approximated by calculating the phase (4.3). Let S_c^x be the horizontal gradient value and S_c^y be the vertical gradient value at (*i,j*). Subscript c denotes the input channel, i.e. L, a, or b.

$$\theta_c(i,j) = \left| \arctan\left(\frac{S_c^y(i,j)}{S_c^x(i,j)} \right) \right|, \quad \theta_c \in \{0, 180\} \tag{4.3}$$

Finally the orientation at a point is determined through a voting of orientation. A histogram with 12 bins is created for each point (i.e. each bin representing 15 degrees increment) and a patch of 3×3 with the point as the center is created at each channel. In total there will be 27 data which will "vote" for the orientation approximation. Bin with the maximum number of element will be chosen as the orientation approximation. For example if the 7-th bin is chosen, the point orientation will be considered to be 90 (= 6×15) degrees.

4.1.2 Histogram Calculation Input

There are several inputs which will be used for histogram calculation. The brightness image (L), two color images (a and b) and texture image. The strength of the method relies on the difference distribution (through histogram calculation) between two parts of a patch. Unfortunately the color images often have a small contrast value, which in turn give us uninformative features. Contrast Limited Adaptive Histogram Equalization (CLAHE) with the limit value as its parameter, is then applied to improve the contrast of the images so that the color images can give us better information to differentiate the candidate points.

Texture image in this paper is generated by applying Gaussian blur along with median blue iteratively to the gradient image *I*. Highly textured object will have a dense high value intensity pixel on the gradient image (Fig.4.1). Applying a Gaussian blur will make parts with high texture to be distinct with the other parts. Median blur will reduce this effect especially on the edge part of the object (maintaining the boundary position).

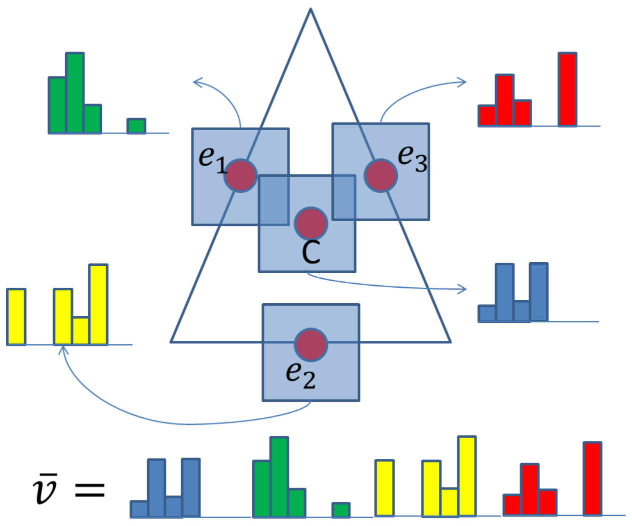


Figure 4.3: An example of calculating a feature from a triangle

4.2 Skeleton Extraction

As mentioned in the first section, we apply the delaunay triangulation to connect the edge points. Delaunay triangulation is used because of its property which hinder a triangulation with a sharp angle. Delaunay triangulation itself had actually been proposed to solve the 2D binary image skeletonization in the past [55, 64].

In 2D binary image skeletonization using Delaunay triangulation, the core skeleton extraction algorithm lies in the grouping of triangles. Several triangles are defined based on the types of edges a triangle has, i.e. sleeve triangle (one of the edges is the contour edge), terminal triangles (two of the edges are the contour edge), and junction triangles (none of the edges are the contour edge). Unfortunately, in grey-scale skeletonization the information of the edges wheteher it belongs to the boundary edges or not is not available. Therefore, a new method to do the skeleton extraction is proposed. The idea of the proposal is to connect triangles sharing the same edges if and only if the triangles belong to the same group.

4.2.1 Triangle Grouping

In order to group the triangles, a feature vector for each triangle is generated. This feature vector consists of the data taken from the brightness image, color images and the texture image. A triangle is considered to be in the same group if the data has similar values one with the other.

Let c be the center of a triangle, e_k is the k-th edge's middle point where

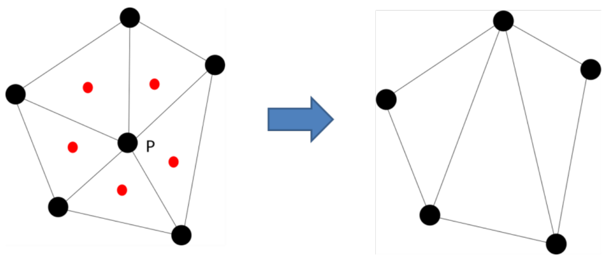


Figure 4.4: Circular point removal example. If point P is connected to triangles with the same group labels (visualize as red points), then it will be deleted from the list and the rest of the points will be re-triangulated.

$k \in 1, 2, 3$. For each of these points, we can create a patch of size $m \times m$ and generate a histogram with h bins. The feature vector $(\bar{v})$ is then generated by concatenating the histogram generated at each point of the triangle. For example a triangle with $h = 16$ will have 256 (= 4 (images from L, a, b and texture image) $\times$16 bins $\times$ (center point + 3 middle point of each edge)) - floating points feature vector (Fig.4.3). Finally the grouping itself is done using the spectral clustering algorithm [65].

For the spectral clustering, a similarity matrix between each triangle is generated. To calculate the similarity of one triangle with another, the exponential of the sum of difference (distance) between the feature vectors $(\bar{v})$ is calculated 4.4. In this paper the Bhattacharyya distance is used (4.5). One nice property of the Bhattacharyya distance is its range which lies between 0 and 1. The subscript c, again is the type of input used (L, a, b, or texture) and H_i denotes the histogram of the i-th triangle. h is the total number of bins.

$$d_{total} = e^{-(d_L + d_a + d_b + d_{texture})} \tag{4.4}$$

$$d_c(H_i, H_j) = \sqrt{1 - \frac{1}{\bar{H}_i \bar{H}_j h^2} \sum_{I=0}^{h} H_i(I) H_j(I)} \tag{4.5}$$

4.2.2 Circular Group Removal

There is a possible result of the triangle grouping which will generate a circular (loop) skeleton as in Fig.4.4. In order to avoid such condition, a point removal algorithm is proposed. This is done by checking the triangle point P and all the triangles with point P as its vertex. If every triangles connected to P has the same group ID, then point P shall be removed. This process is done iteratively r times.

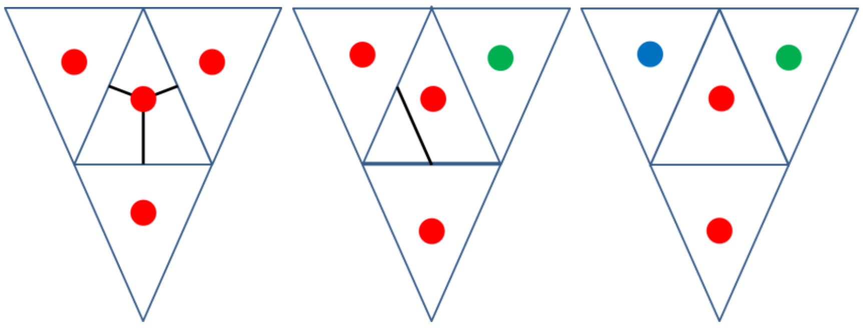

(a) Junction Triangle (b) Sleeve Triangle (c) Terminal Triangle

Figure 4.5: Three types of triangle with its connection (black lines). The colored points represent the group label of the triangle. Color Similarity represents the same group

4.2.3 Extraction

Similar with [55, 64] there are three possible conditions for skeleton extraction step (Fig.4.5). The first one is the junction triangle. Here junction triangle is defined as a triangle with all of its neighboring triangles belonging to the same group. The terminal triangle, when one or none of the neighboring triangle belongs to the same group. The third and final type of triangle is the sleeve triangle where two of the neighboring triangles belong to the same group with the center triangle.

If a triangle is a junction triangle, then the center point of the triangle will be connected to all of the middle point of each edge points. If a triangle is a sleeve triangle, then the middle point of the shared edges between the triangle and its neighboring triangles will be connected. Finally no connection will be generated for the terminal triangle.

4.2.4 Pruning and Smoothing

Pruning operation is applied to cut the redundant (noisy) skeleton branch (Fig.4.6). The green and blue points represent skeleton points connected to a junction point (red). The connection from the blue point to the red point will be removed since it has only one connection to the junction point and is considered to be a noisy branch.The redundant branch is identified by the connection of each skeleton point. If a skeleton point connected only to the junction point, then this point will be removed. Finally the smoothing operation is done using the Bezier curve, where the skeleton points are used as the control points.

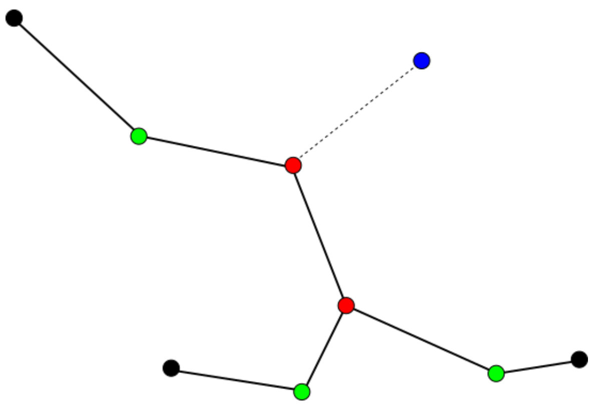


Figure 4.6: Pruning step

4.3 Experimental Result and Discussion

All the experiment were done using C++ language and OpenCV 2.4.9 library. Twenty images taken from Berkeley BSDS 300 dataset [66] and Caltech 101 dataset [67] were used to test the algorithm. Fig.4.8 and Fig.4.9 shows the images used in this experiment. Since they have varying size, all images used were resized into 300×200 resolutions.

The results of the algorithm are compared with a method based on the SSM method in[40]. Since the SSM method depends on the Canny edge extraction, we vary the threshold from 40 to 100 with 10 value increment and take the best F-measure, which will be explained later, to be compared with the proposed method. Note that in the original paper, the second half of the algorithm relies on the background-foreground segmentation of the image. Doing so will alter the problem from the grey skeletonization into the binary image skeletonization. Therefore, we avoid implementing the segmentation part of the algorithm.

Fig.4.7 shows the results including the comparison. Note that for visualization purpose, the skeleton image is overlaid on top of the original input image and colorized. As can be seen from the figures, most of the time the proposed method can represent the skeleton of the object in the image well.

The most clearly seen example is from the starfish image with white background (starfish4 image, 2nd last row in Fig.4.7). Note here that the SSM method, which relies heavily on the edge information only, suffers from the incomplete edge extraction. Therefore, generating many errors. In comparison, the proposed method, although extracting only sparse point of the edge boundary, can generate a generally better skeleton which represents the shape very well. This is due to the point removal step and clustering step of the triangles. Even though there are some errors during the edge (point) extraction, such as points inside the object instead of points on the boundary, these points will be removed during the circular

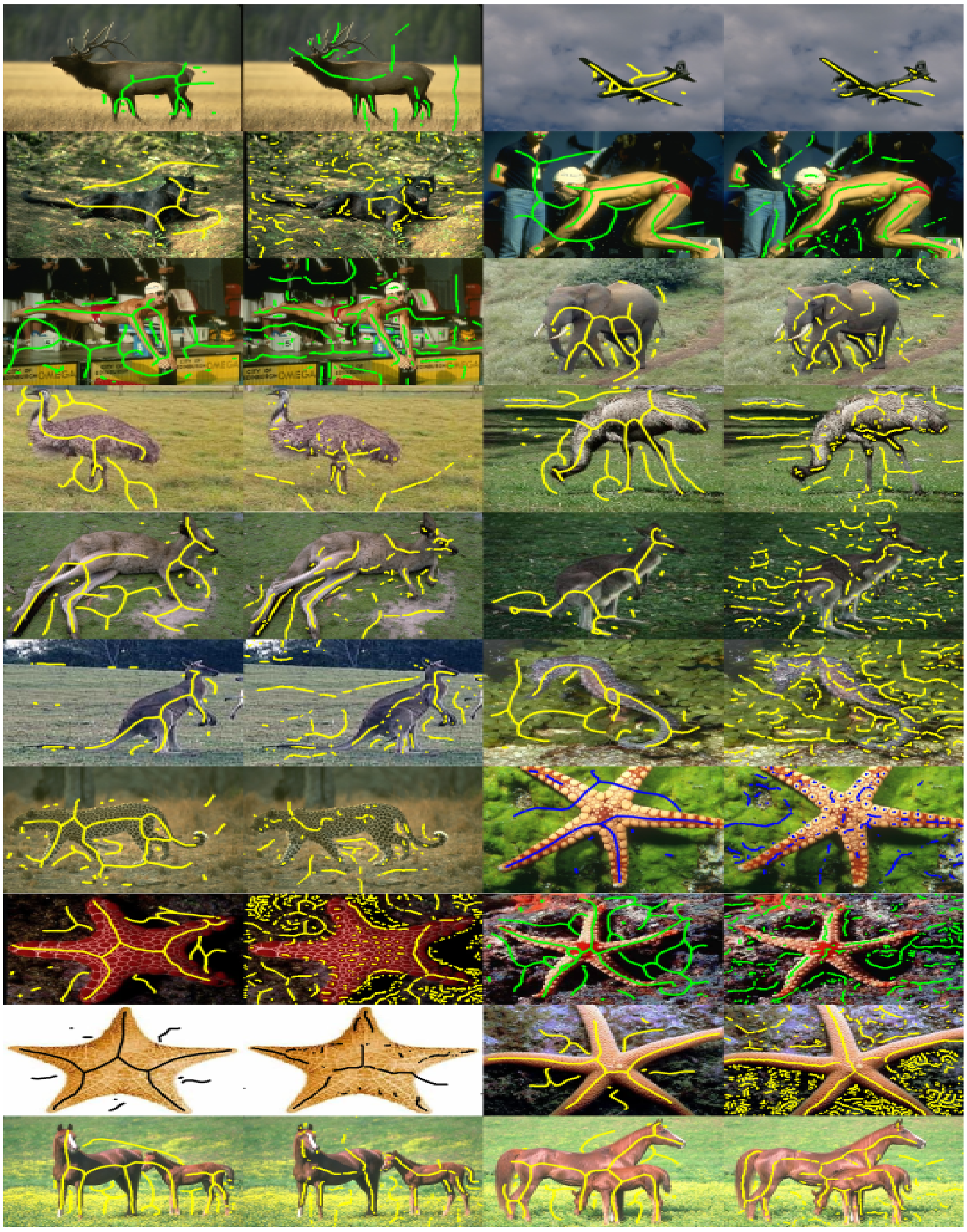


Figure 4.7: Result comparison of the proposed method (column 1 and 3) with the SSM method (column 2 and 4).

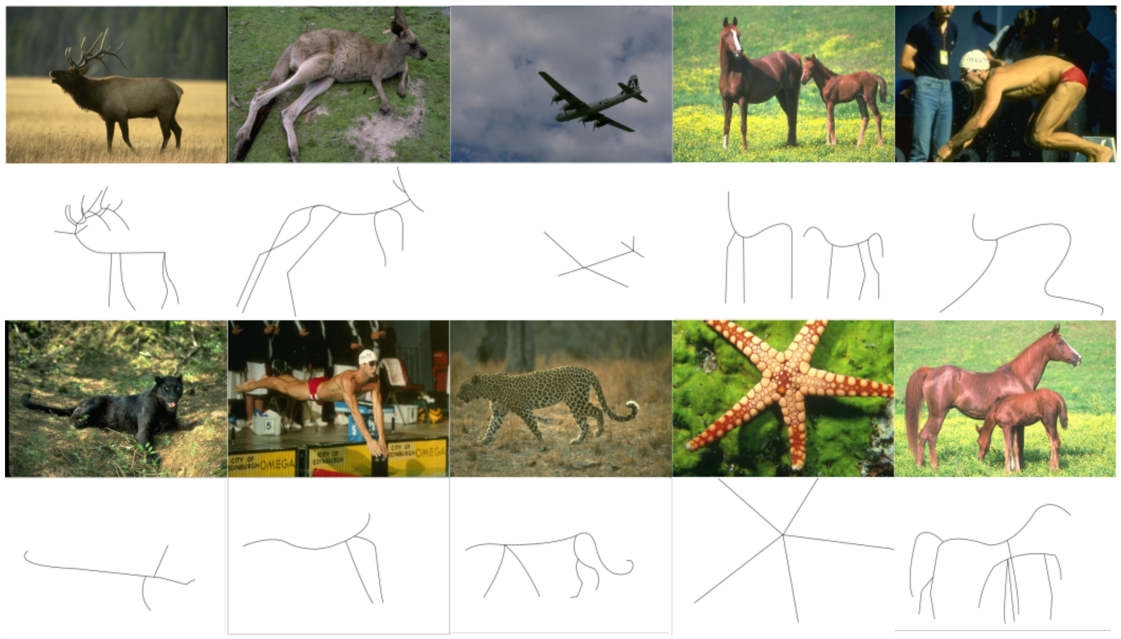


Figure 4.8: Input images from Berkeley Dataset. The second and fourth row are the ground truth images generated by human observer. Starting from the first row: deer, kangaroo1, airplane, horse, swimmer1, panther, swimmer2, cheetah, starfish1, horse2.

point removal, since the point inside the object will have triangles such that the surrounding triangles will have the same features (colors, texture, etc.).

In order to have a better comparison, the generated skeleton image is also compared with the ground truth image created by human observer and then calculated using the F-measure (4.6). The skeleton image is created as a binary image where pixels crossed by the skeleton have value zero (black pixel) and value one (white pixel) otherwise.

$$F = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{4.6}$$

$$Precision = \frac{true\,positive}{true\,positive + false\,positive} \tag{4.7}$$

$$Recall = \frac{true\,positive}{true\,positive + false\,negative} \tag{4.8}$$

Let x_t denotes the pixel where both the ground truth (GT) image and the generated skeleton image (S) have the same zero value (4.9). The sigma notation can be considered as a threshold for a slight offset generated by skeleton. Then the true positive is defined as in (4.10). n and m denote the row and column size

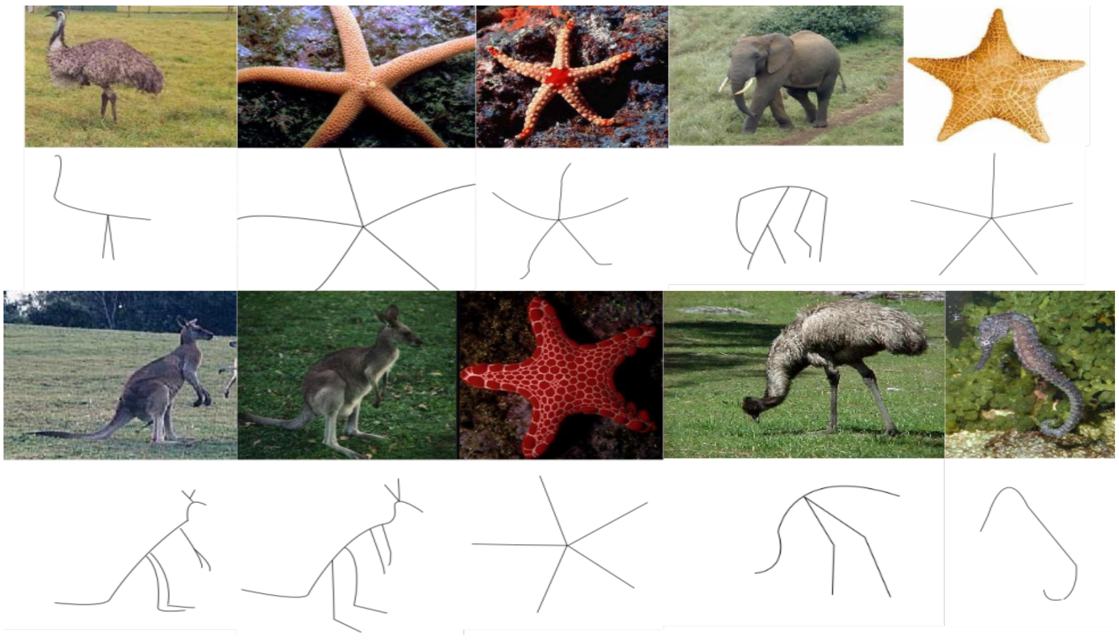


Figure 4.9: Input images from Caltech 101 Dataset. The second and fourth row are the ground truth images generated by human observer. Starting from the first row: emu1, starfish2, starfish3, elephant, starfish4, kangaroo2, kangaroo3, starfish5, emu2, seahorse.

respectively.

$$x_t = \begin{cases} 1, & \text{if } S(i,j) = GT(i,j) \pm \sigma \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

$$true\,positive = \sum_{i=0}^{n}\sum_{j=0}^{m} x_t(i,j) \tag{4.10}$$

Let x_{fp} denotes the pixel which has value one in the skeleton image and value zero in the ground truth image (4.11). Therefore, the false positive in (4.7) is defined as the sum of this x_{fp}. Similarly, let x_{fn} denotes the pixel which has value zero in the skeleton image and value one in the ground truth image (4.12). Then, the false negative in (4.8) is defined as the sum of x_{fn}. Table 4.1 shows the result of the F-measure comparison.

$$x_{fp} = \begin{cases} 1, & \text{if } S(i,j) = 1 \wedge GT(i,j) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

$$x_{fn} = \begin{cases} 1, & \text{if } S(i,j) = 0 \wedge GT(i,j) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

Table 4.1: Precision, Recall and F-measure comparison of SSM and the proposed method

Image	SSM			Proposed		
	Precision	Recall	F-measure	Precision	Recall	F-measure
deer	0.236	0.147	0.182	0.350	0.262	**0.300**
airplane	0.468	0.319	0.379	0.345	0.514	**0.413**
panther	0.091	0.206	0.127	0.120	0.327	**0.175**
swimmer1	0.197	0.352	**0.253**	0.159	0.430	0.233
swimmer2	0.122	0.154	0.136	0.163	0.297	**0.210**
elephant	0.242	0.202	0.220	0.240	0.282	**0.259**
emu	0.100	0.098	0.099	0.147	0.386	**0.213**
emu2	0.088	0.115	0.100	0.143	0.402	**0.211**
kangaroo	0.203	0.191	0.197	0.170	0.318	**0.222**
kangaroo2	0.089	0.121	0.103	0.244	0.283	**0.262**
kangaroo3	0.144	0.159	0.151	0.264	0.302	**0.282**
seahorse	0.035	0.108	0.053	0.124	0.312	**0.177**
cheetah	0.143	0.112	0.126	0.270	0.572	**0.367**
starfish	0.096	0.104	0.100	0.214	0.357	**0.268**
starfish2	0.025	0.072	0.037	0.140	0.319	**0.195**
starfish3	0.015	0.034	0.021	0.140	0.504	**0.219**
starfish4	0.263	0.181	0.214	0.313	0.455	**0.371**
starfish5	0.111	0.180	0.137	0.219	0.253	**0.235**
horse	0.283	0.257	0.270	0.241	0.525	**0.333**
horse2	0.270	0.265	0.268	0.237	0.386	**0.294**
Average	0.161	0.169	0.158	0.212	0.374	**0.262**

Almost all of the image being tested has better F-measure compared with the SSM Method. Only one of them, i.e. the swimmer1 image, does have a better F-measure than the proposed method. This is possibly due to the slight shift of the generated skeleton. For example see Fig.4.10. Assume that the image on the left of Fig.4.10 are the proposed method and the right part is the existing method. The red curve is the ground truth skeleton, the gray area is the thresholded part and the black curve is the generated skeleton.

Notice also that, even though most of the results have low valued F-measure, the proposed method can still represent the object well as mentioned before (refer again to starfish4 image in Fig.4.7 for visualization). One side note needs to be addressed here. The reader might wonder why the image deer has lower F-measure compared with the proposed method, although SSM method seems to have a better result visually. This is due to the higher number of "error" skeleton generated by

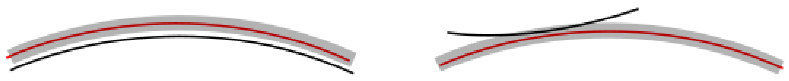


Figure 4.10: Although generated better skeleton, since it does not intersect the ground truth skeleton, the proposed method has lower F-measure

Table 4.2: Parameter test

Parameter	Value
CLAHE limit	2, 4
Patch size	3, 5, 9, 15
Bin size	20, 30, 40
Removal iteration (r)	1, 3, 5

the SSM method on the background image which reduces the precision and recall values.

Parameters, such as the limit for CLAHE, the patch and bin sizes when grouping the triangles and the number of iteration for circular point removal (r) will determine the result. Every image, of course, has their best parameter for the best result. In order to have a better guidance in regard to selecting the parameters, the following experiment is done. For each image, we vary the parameters values as in Table 4.2. Then for each image we calculate the F-measures. In total there are 1440 data collected (= 2 CLAHE limit $\times$ 4 patch sizes $\times$ 3 bin sizes $\times$ 3 r $\times$ 20 images). Table 4.3 shows the summary of the parameter test.

Notice from Table 4.3, compared to the other parameters, the average F-measure does not differ much (less than 0.001 difference), this means that the bin size has a relatively low influence on the result. CLAHE limit parameters show that the limit of value of 2 give a better result. As expected, applying too much contrast adjustment to the image give a worse solution since part of the object will be different one and another resulting in a strict triangle grouping.

For the patch size, the larger size seems to have a better result, since smaller size will have a very little information of the area. Finally the removal iteration r. Here it shows that 3 gave us the best result. Similar with the CLAHE limit parameter, too much removal will result in lost of skeleton information and too little iteration will give a looping skeleton.

Table 4.3: Summary (average F-measure) of the parameter testing

Parameter	Average F-measure
CLAHE limit:	
2	0.198
4	0.192
Patch Size:	
3	0.192
5	0.194
9	0.196
15	0.199
Bin Size:	
20	0.1954
30	0.1957
40	0.1946
Removal Iteration (r)	
1	0.177
3	0.205
5	0.204

Chapter 5

Object Specific Non-rigid (Articulated) 3D Model Retrieval using Scale-Rotation Invariant Shape Context on Skeleton

It has been shown that by using skeleton information, 3D model retrieval can be achieved. Notice here, that the retrieved object usually does not mean the same object will be retrieved. What usually retrieved is object with the same structure as the query object. In this chapter we are trying to solve this drawback, i.e. retrieving the same object as the query (image) even though they have different poses.

5.1 Shape Feature

Assuming that the skeleton is already given, the problem is then how to match the shape of the model of the query with the database. Therefore, we need a feature (descriptor) that can match the shape or appearance of the 3D model. Shape context [68, 69] is a feature that can be used to record the shape of an object.

Shape context works by first extracting the edge information of the object. External and internal edge information are used for the calculation of the shape context desciptor. From the edges, a uniform distance spaced points are taken as sampling points, i.e. points with equal spacing. For each points, a shape context descriptor is generated.

The descriptor is generated by checking the distribution of the position of neighboring points with respect to the current point. In order to capture the distribution information, it is natural the make a histogram. The input to the his-

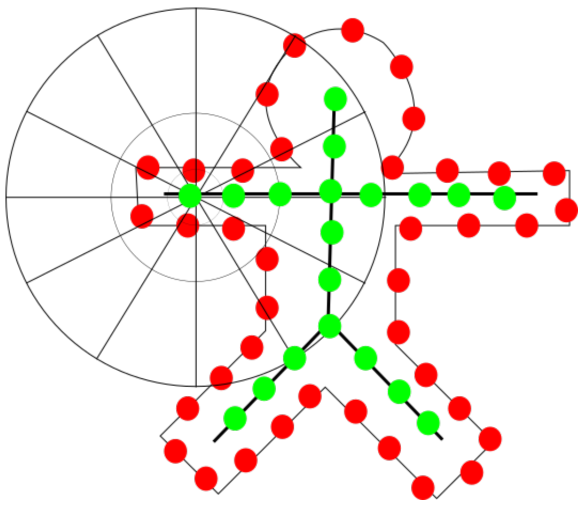


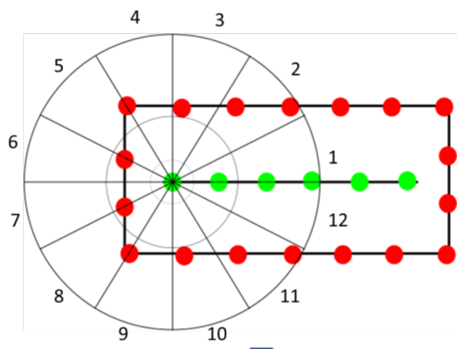
Figure 5.1: Skeleton shape context

togram are then comprised of the log-polar information of the neighboring points. In Fig.5.3 the center of the log-polar histogram (the wheel-like figure) is the point of which the shape context descriptors will be generated. Points lying on the bins will be counted for the respective bins. In the example, 12 angular bins and 2 log distance bins are used. All the points (red circles) will have its own shape context descriptor and will be used for the matching between the points. In order to calculate the histogram distance, originally the χ^2 distance is used.

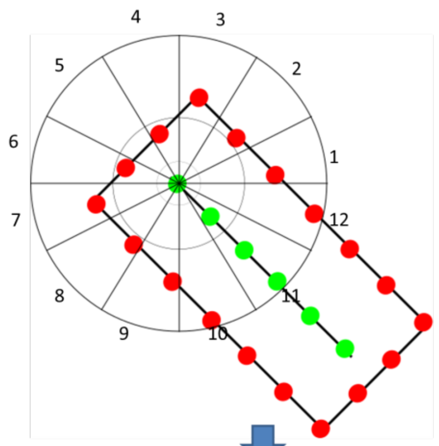
While it might work well for rigid object, using only the shape context descriptor to the query will not work for non-rigid object. This is because of the appearance of the object might be completely different in differing poses and viewing-angle.

To minimize such effect, we propose to combine the shape context with the skeleton information (Fig.5.1). Instead of generating the sampling points on the contour of the objects, we generate them on the skeleton instead. By doing so, it reduces the number of points need to be checked (and therefore reduces the matching computation time). In further section, we will also prove that by using the skeleton information, a better retrieval rate can also be achieved.

Notice that the shape context feature is not rotation and scale invariant (Fig.5.2). As can be seen, by rotating the object, the generated log-histogram (the box under each object figure) generated a different result eventhough it should be the same since basically the object is the same. To minimize the rotation effect, we propose using the cross correlation when calculating the similarity between the log-histograms. Let H_q and H_r be the log-histogram of the query point and reference point respectively. The cross correlation can be regarded as the convolution of the reference onto the query log-histogram (5.1). $g(i, j)$ denotes the output of



0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0
0	2	1	1	0	0	0	1	0	1	2	0
1	2	3	4	5	6	7	8	9	10	11	12



0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0
1	1	1	0	0	0	1	1	1	1	0	0
1	2	3	4	5	6	7	8	9	10	11	12

Figure 5.2: Rotation and Scale variant Shape Context

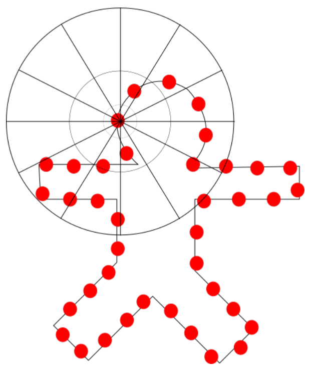


Figure 5.3: An example of shape context descriptor

the convolution. Finally, the value that defines the simmilarity between two histograms can be regarded as the maximum value of the cross correlation calculation (5.2).

$$g(i, j) = \sum_{k,l} H_q(i + k, j + l)H_r(k, l) \tag{5.1}$$

$$d(H_q, H_r) = \arg\max_{i,j} g(i, j) \tag{5.2}$$

In order to minimize the scale effect, the k-nearest neighbor of the outer contour points is used. The distance of the skeleton point with the k-th nearest neighbor will then be used as the base radius of the log-histogram calculation for the shape context feature. Let r be the radius, i.e. the k-th nearest neighbor distance, then the log-histogram will have a radius of $(1 + \lambda)r$. The outer contour information is acquired by first segmenting the input image using the grabcut algorithm and then extracted using the Canny edge extraction algorithm. From the edge information, the sampling points with uniform distance are then extracted. Notice that when calculating the shape context feature, the inner contour or edge information is preserved (also be used).

5.2 Matching Step

The cost of matching between the query and the object in the database is calculated from the cross matching of the query features and the object features. Let f_i^q be the i-th (histogram) feature of the query (superscript q) and $i \in \{0, 1, \ldots, n\}$ with n is the number of features of the query. Similarly, let f_j^r be the j-th feature of the reference (superscript r) and $j \in \{0, 1, \ldots, m\}$ with m is the number of features of the reference. Therefore, the total pair of features will then be $n \times m$.

Let p_{ij} be the cost of matching feature f_i^q and f_j^r using (5.2). This can be represented in a matrix form of size $n \times m$. The pair $p_{i,j}$ will be considered to be a valid pair if $p_{i,j}$ if it has the maximum value on the i-th column and j-th row. Let P_{qr} be the set of matching pairs of the query and reference features. The similarity (S_{qr}) of the query and the reference object is then calculated by the average cost of the pairs in P_{qr} (5.3), where N is the total number of element and i denotes the i-th element of the set P_{qr}.

$$S_{qr} = \frac{1}{N} \sum_{i=0}^{N} P_{qr}[i] \tag{5.3}$$

An object in the database will be represented by several images of the object taken from several differing viewing angle (Fig. 5.4). Following the notation previously defined, notice here that n (the number of feature of the query) will be less than the number of feature of the reference m. For simplicity, assume that the number of features generated in an image is constant f number. Therefore, the query will have $n = f$ number of feature, while one object in the database will have $m = N_i \times f$ number of features, where N_i is the number of images representing the object. The intuition behind this is to minimize the effect of the difference of the shape and information in viewing angle direction, i.e. local information is sufficient enough to be used in matching the object in differing viewing angle.

5.3 Experiment Result

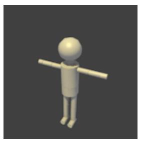
The first experiment is done to compare the results of using only the shape context feature and the results of using our proposal, the scale-rotation invariant on the skeleton. Fig.5.5 shows the models with standard pose that are used and stored in the database. For each object 21 orientations were used to represent the object (similar with Fig.5.4). Along with the image database, the skeleton information is also stored in the database. Each image has a resolution of 300 $\times$ 300 pixels.

The parameters setting is 12 angular bins and 3 distance bins. The sampling distance on the edge is fixed to be 3. For the shape context only feature, the radius when calculating the log-histogram was fixed as 90 and when comparing the log-histograms, χ^2 distance was used. The proposal on the other hand used the cross correlation method described in section 5.1 with $k = 10$. Both of the methods used the cross matching method described in section 5.2. Fig.5.6 shows the test pose for the experiment. The test model is rendered similarly like the image in Fig., i.e. rendered in different viewing angle. In total there were 21 images rendered, each will be the test image independently. This means that there will be 21 data to assess our statement. Table (5.1 - 5.3) shows the result of using only the shape

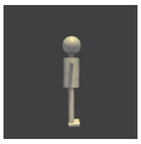
(a) x:-30, z = -90

(b) x:-30, z = -60

(c) x:-30, z = -30

(d) x:-30, z = 0

(e) x:-30, z = 30

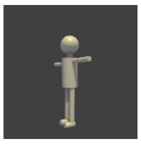
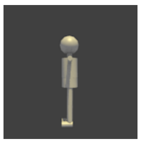
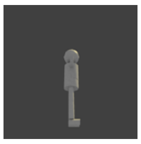
(f) x:-30, z = 60

(g) x:-30, z = 90

(h) x:0, z = -90

(i) x:0, z = -60

(j) x:0, z = -30

(k) x:0, z = 0

(l) x:0, z = 30

(m) x:0, z = 60

(n) x:0, z = 90

(o) x:30, z = -90

(p) x:30, z = -60

(q) x:30, z = -30

(r) x:30, z = 0

(s) x:30, z = 30

(t) x:30, z = 60

(u) x:30, z = 90

Figure 5.4: Rendered 3D models for one class of object in the database

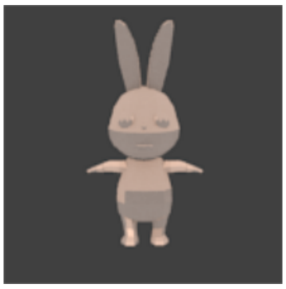
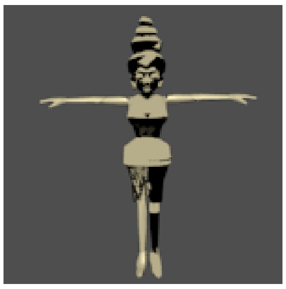
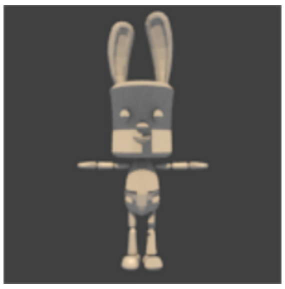
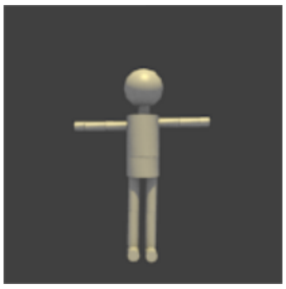
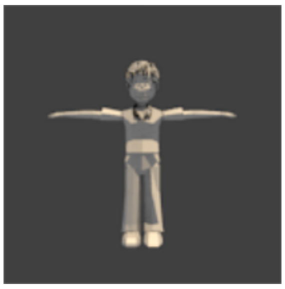

(a) Bunny

(b) Cherry

(c) Enchu

(d) Template

(e) Student

Figure 5.5: Models in the database

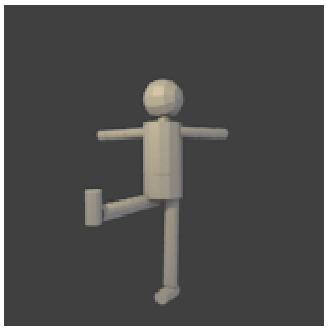


Figure 5.6: Test model with different pose (Test Pose)

Table 5.1: Shape Context Result on Test Pose with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**5.077**	**6.492**	6.031	**6.103**	**6.098**	9.153	**5.927**
Student	8.591	9.361	**5.929**	7.699	8.490	8.722	9.405
Bunny	6.815	8.354	8.162	7.954	7.387	9.026	7.404
Cherry	7.637	8.372	6.340	7.192	7.411	**7.625**	7.239
Echu	7.322	10.025	7.581	9.100	7.951	8.352	7.628

context and Table (5.4 - 5.5) shows the result of combining the skeleton and shape context information.

Notice here that, since the shape context used the χ^2 as the calculation method when comparing two histograms and our proposal used the cross correlation, the decision on which object to be retrieved will be different. While the proposed method considered the maximum value as the best matching candidate, the shape context considered the minimum value as the best matching candidate.

The bold number represents the smallest (or largest) cost value and therefore the retrieved object. For example in Table 5.1 the query is the template model with z = 0, and the bold number belongs to the template model, then it is a correct retrieval. In the contrary, for example in the same table, when the query is z = -60, using only shape context descriptor, the (falsely) retrieved object is cherry since it has the smallest value. All the query belongs to the Template model, therefore, the bold number should be in the same row as the Template model (first row). As can be seen from the tables, adding the skeleton information gave us (18/21 =)85.7% accuracy for the test pose, while using only shape context it gave us only (14/21 =) 67%.

To further assess the proposed method, several test queries were tested and similarly, the retrieval rate is calculated. Figure 5.7 shows the query objects.

Table 5.2: Shape Context Result on Test Pose with x = 30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**5.892**	**7.963**	5.929	**7.214**	6.929	**5.945**	**6.431**
Student	6.828	9.277	6.823	8.080	**6.710**	7.082	8.369
Bunny	7.928	10.507	8.043	8.529	10.048	7.789	7.185
Cherry	5.965	9.644	**5.714**	7.453	7.142	7.261	7.567
Echu	7.450	9.522	6.435	9.871	8.581	6.718	8.077

Table 5.3: Shape Context Result on Test Pose with x =-30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**4.616**	8.689	7.803	**7.847**	**6.073**	**5.428**	6.760
Student	6.118	**7.390**	8.900	9.973	7.385	7.401	8.316
Bunny	7.359	7.995	8.034	8.238	6.399	6.348	**6.207**
Cherry	8.199	7.886	**7.625**	8.716	7.797	7.965	6.725
Echu	7.164	8.146	8.167	9.373	7.220	6.581	7.888

Similar with the previous experiment, each object will be rendered 21 times from different viewing angle and therefore, there will be 21 data collected for each object query. The detail of the results can be viewed in appendix B.1 - B.15. Table 5.7 shows the summary of the results. For the student pose objects, there are slight parameter changes, i.e. 5 distance bins instead of 3 and $k = 20$ for the best result.

As can be seen from the results shown in Table 5.7, our proposal generate a promising result of achieving 93.32% correct retrieval rate. These results however, due to the manually chosen parameter setting, i.e. for the best performance each object has their own parameter setting. For further direction, an automatic parameters selection should be considered.

Table 5.4: Shape Context including skeleton result on Test Pose with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**13.781**	**15.968**	**17.000**	**16.000**	**16.333**	**16.833**	**16.045**
Student	13.317	14.889	15.375	14.760	15.333	16.125	15.417
Bunny	11.000	10.675	13.583	13.040	12.875	14.219	13.571
Cherry	12.131	14.098	14.258	14.050	14.063	14.125	14.468
Echu	12.414	14.173	14.639	13.676	14.510	14.298	14.008

Table 5.5: Shape Context including skeleton result on Test Pose with x = 30

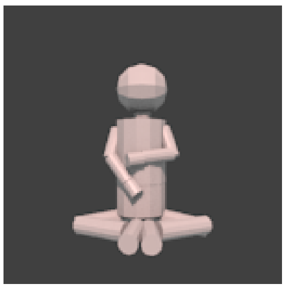
Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**16.111**	**16.810**	16.333	**15.944**	12.987	**16.500**	15.667
Student	15.000	14.816	**16.417**	14.109	**13.320**	14.494	**15.833**
Bunny	13.500	14.107	13.500	12.500	11.177	14.375	14.750
Cherry	14.396	14.452	14.571	12.597	13.021	13.400	14.400
Echu	14.333	13.355	13.619	13.976	12.332	113.556	15.000

Table 5.6: Shape Context including skeleton result on Test Pose with x =-30

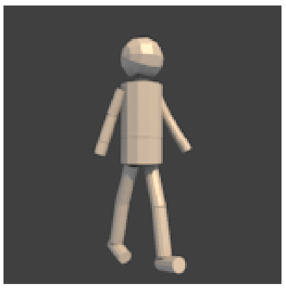
Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**15.717**	**15.619**	**15.762**	**15.500**	**16.110**	**16.111**	**14.733**
Student	14.283	14.149	14.725	14.616	14.460	13.901	14.444
Bunny	12.906	11.636	14.000	13.339	12.120	12.687	12.130
Cherry	14.286	13.770	15.167	14.173	13.897	14.763	13.454
Echu	13.485	13.339	14.114	14.000	14.631	13.635	12.457

Table 5.7: Summary of the result using our proposal

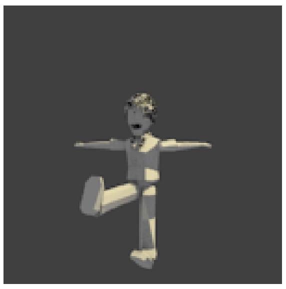
Object	Accuracy
Template Pose 2	100%
Template Pose 3	95.2%
Student Pose 1	85.7%
Student Pose 2	95.2%
Student Pose 3	90.5%
Average	93.32%



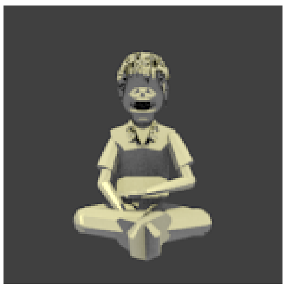
(a) Template pose 2



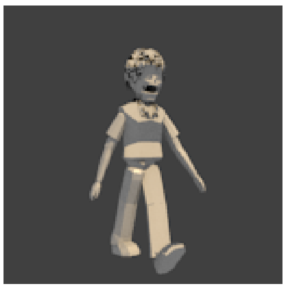
(b) Template pose 3



(c) Student pose 1



(d) Student pose 2



(e) Student pose 3

Figure 5.7: Query Objects

Chapter 6

Conclusion

There are mainly three parts in this (view-based) 3D model retrieval study: Rigid 3D model retrieval, skeletonization and non-rigid 3D model retrieval system. In Rigid 3D model retrieval, we propose a novel method using the skewness map approach. By using the skewness map approach, with 3 models in the database the system can enhance the speed of retrieval 4 times faster than the conventional method of using the SURF method. In matching times, the system can enhance the speed up to 14 times of the conventional method. Furthermore, the proposed method can also approximate the orientation of the query object with averagely less than 1 degrees of error both in x and y direction. By sacrificing an increase of error of the orientation approximation, using our proposal CRN descriptor, the system can further speed up the total computation time 11 times faster than the conventional SURF method.

As like the other view-based methods, our approach has some limitations. First, the objects retrieved depends on the orientation the object in the input image taken. Let say that we have a cup with some objects inside the cup. Our proposal will only retrieve the cup and not the objects inside the cup. In other words, unless the cup in the database has the same or similar objects already inside the cup, our proposal will not be able to separately retrieve all the items. Since the skewness map depends on the shape of the object, the speed-up process will not work for a round object, such as ball, since it will have the same skewness value irrespective of the viewing angle.

We also propose a new method in solving the grey-scale skeletonization problem by utilizing the Delaunay triangulation. A new skeletonization based on the triangle grouping and sparse points condition is proposed in the second parts of this study. By using our approach, we can get a better result visually (subjectively) and numerically (objectively) of achieving 1.7 times higher F-measure compared to another method.

One of the limitation in our proposal here is that the object has to have a uniform color or texture distribution. Thus, object with a lot contrast, like human with bright and contrast colored clothes, will not be applicable for our proposal since the skeleton extracted will be based on the clothes instead of the whole object (human pose). Occlusion proves still to be a challenging work in skeletonization, thus our work also has a limitation that part of objects occluded will not be able to be extracted correctly.

Finally on the last part of the study regarding the non-rigid (articulated) 3D models retrieval, using a combination of shape context and skeleton information of the query and the reference images, we show a promising result of retrieving the correct (the exact) same model even though they have different pose. An assumption in our current implementation is that the skeleton can be extracted correctly. Our proposal achieves 93.3% accuracy on retrieving the correct object. There are still some parameter issues of the proposed method. In current implementation, the parameters are still chosen manually to give the best result. Particularly in determining the number of k for the k-nearest neighbors (which is made constant in current implementation) for each point. For further direction on this research, an adaptive selection of k should be considered.

Most of the results described in this dissertation can be found on several journals and conference papers as follow:

- **Rigid 3D Model Retrieval using Skewness Map:**

  1. V. Sintunata, Kurumi Kaminishi and T.Aoki, "Skewness Map: Estimating Object Orientation for High Speed 3D Object Retrieval System", International Journal of Intelligent Engineering Informatics, 2017. (in press)
  2. V.Sintunata and T.Aoki, "High Speed 3D Object Orientation Estimation for High Speed 3D Object Retrieval", in 4th International Conference on Information and Communication Technology (ICoICT), 2016.
  3. V.Sintunata and T.Aoki, "High Speed 3D Object Retrieval using Skewness Value", IEEE Region 10 Symposium (TENSYMP), 2016.
  4. V.Sintunata and T.Aoki, "3D Object Retrieval using Skewness Database", International Conference on Control, Decision, and Information Technology (CoDIT), 2016.

- **Natural Image Skeletonization using Delaunay Triangulation:**

  1. V.Sintunata and T.Aoki, "Skeletonization in Natural Image using Delaunay Triangulation", Advances in Science, Technology and Engineering Systems Journal, vol. 2, no.3, pp.1013-1018, 2017. (in press)

2. V.Sintunata and T.Aoki, ”Grey-scale Skeletonization using Delaunay Triangulation”, IEEE International Conference on Consumer Electronic - Taiwan, 2017.
3. V.Sintunata and T.Aoki, ”Skeleton Extraction in Cluttered Image based on Delaunay Triangulation”, IEEE International Symposium on Multimedia (ISM), 2016.

Publication List

Journal:

1. V.Sintunata and T.Aoki, ”Skeletonization in Natural Image using Delaunay Triangulation”, Advances in Science, Technology and Engineering Systems Journal, vol. 2, no.3, pp.1013-1018, 2017. (in press, related to Chapter 4)

2. V. Sintunata, Kurumi Kaminishi and T.Aoki, ”Skewness Map: Estimating Object Orientation for High Speed 3D Object Retrieval System”, International Journal of Intelligent Engineering Informatics, 2017. (in press, related to Chapter 3)

3. V. Sintunata and T. Aoki, ”Color Segmentation based Depth Adjustment for Image Reconstruction from a Single Input Image”, International Journal of Computer Theory and Engineering, vol. 8, no. 2, April 2016. (unrelated)

4. V. Sintunata and T. Aoki, ”Shape from Shading using Color Segmentation based Depth Adjustment”, International Journal of Enhanced Research in Science Technology and Engineering, vol. 3, issue 11, pp.64-72, 2014. (unrelated)

5. T.Aoki and V.Sintunata, ”Direct Joint Detection from Humanoid 3D Models without using Skeleton Information”, International Journal of Electronics Communication and Engineering, vol. 5, issue 3, May 2014. (unrelated)

Conference:

1. V.Sintunata and T.Aoki, ”Grey-scale Skeletonization using Delaunay Triangulation”, IEEE International Conference on Consumer Electronic - Taiwan, 2017. (related to Chapter 4)

2. V.Sintunata and T.Aoki, ”Skeleton Extraction in Cluttered Image based on Delaunay Triangulation”, IEEE International Symposium on Multimedia (ISM), 2016. (related to Chapter 4)
   DOI: 10.1109/ISM.2016.0080.

3. V.Sintunata and T.Aoki, ”Object Orientation Estimation for High Speed 3D Object Retrieval”, in 4th International Conference on Information and Communication Technology (ICoICT), 2016. (related to Chapter 3)
   DOI: 10.1109/ICoICT.2016.7571933

4. V.Sintunata and T.Aoki, ”High Speed 3D Object Retrieval using Skewness Value”, IEEE Region 10 Symposium (TENSYMP), 2016. (related to Chapter 3)
   DOI: 10.1109/TENCONSpring.2016.7519435

5. V.Sintunata and T.Aoki, ”3D Object Retrieval using Skewness Database”, International Conference on Control, Decision, and Information Technology (CoDIT), 2016. (related to Chapter 3)
   DOI: 10.1109/CoDIT.2016.7593547.

6. O. Hirvola, T.Viitanen, V.Sintunata and T.Aoki, ”Improved Image Quality in Fast Inpainting with Omnidirectional Filling”, International Conference on Image, Vision and Computing (ICIVC), 2016. (unrelated)
   DOI: 10.1109/ICIVC.2016.7571269.

7. N.T.H.Van, V. Sintunata, and T.Aoki, ”Automatic Image Colorization based on Feature Lines”, in Proc. 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, vol.4: VISAPP, pp.126-133, 2016. (unrelated)

8. T.Aoki and V.Sintunata, ”Template Matching Skeletonization based on Gauss Sphere Representation”, IEEE International Symposium on Multimedia (ISM), 2013. (unrelated)
   DOI:10.1109/ISM.2013.19.

9. T.Aoki and V.Sintunata, ” Automatic Animation Skeleton Extraction Method based on Vertex Gauss Sphere Representation”, International Conference on Computational Intelligence Modelling and Simulation (CIMSIM), 2013. (unrelated)
   DOI: 10.1109/CIMSim.2013.21

Appendix A

Result of SMSURF and SMCRN

Table A.1: Accuracy Comparison of SMSURF and SMCRN

Query	Ground Truth		SMSURF				SMCRN			
	x	y	x	y	Δx	Δy	x	y	Δx	Δy
Bike	232	69	232	69	0	0	232	69	0	0
	148	51	148	51	0	0	150	53	2	2
	271	85	271	85	0	0	271	85	0	0
Bird	222	166	225	164	3	2	222	166	0	0
	275	223	275	223	0	0	275	223	0	0
	175	193	175	193	0	0	175	193	0	0
Bugs	325	9	324	8	1	1	324	5	1	4
	123	276	123	275	0	1	123	276	0	0
	355	77	355	77	0	0	355	77	0	0

Continued to next page

Continuation of Table A.1

Query	Ground Truth		SMSURF				SMCRN			
	x	y	x	y	Δx	Δy	x	y	Δx	Δy
Bus	133	12	133	12	0	0	133	12	0	0
	195	138	195	138	0	0	195	138	0	0
	325	193	325	193	0	0	325	193	0	0
Cake	77	49	77	49	0	0	77	49	0	0
	280	6	282	7	-2	-1	70	6	210	0
	288	61	288	61	0	0	288	61	0	0
Cello	283	217	283	217	0	0	283	217	0	0
	148	193	148	193	0	0	148	193	0	0
	25	24	25	24	0	0	25	24	0	0
Chicken	255	357	255	357	0	0	255	357	0	0
	107	155	107	155	0	0	107	115	0	0
	47	250	47	250	0	0	47	250	0	0
Cow	200	262	200	262	0	0	200	262	0	0
	193	5	193	5	0	0	193	5	0	0
	207	304	207	304	0	0	207	304	0	0
Dino	55	129	55	129	0	0	55	129	0	0
	307	158	307	158	0	0	307	158	0	0
	126	141	126	141	0	0	126	141	0	0
Flower	82	146	86	147	4	1	87	145	5	1
	295	246	303	246	8	0	300	241	5	5
	300	339	300	339	0	0	300	339	0	0

Continued to next page

Continuation of Table A.1

Query	Ground Truth		SMSURF				SMCRN			
	x	y	x	y	Δx	Δy	x	y	Δx	Δy
Frog	130	163	129	164	1	1	129	164	1	1
	311	97	311	97	0	0	311	97	0	0
	141	135	141	135	0	0	141	135	0	0
Hikoboshi	256	277	256	277	0	0	256	277	0	0
	70	288	70	288	0	0	70	288	0	0
	186	42	186	42	0	0	186	42	0	0
Horse	43	41	43	41	0	0	43	41	0	0
	292	318	292	318	0	0	292	318	0	0
	250	34	251	36	1	2	242	42	6	8
Incense Holder	265	65	265	65	0	0	265	65	0	0
	226	200	226	200	0	0	226	200	0	0
	241	354	241	354	0	0	241	354	0	0
Kimono	344	174	344	174	0	0	344	174	0	0
	202	122	202	122	0	0	202	122	0	0
	289	343	289	343	0	0	289	343	0	0
Mini Dragon	305	143	302	145	3	2	142	130	163	13
	184	354	184	354	0	0	184	354	0	0
	36	291	36	291	0	0	36	291	0	0
Mochi	140	37	140	37	0	0	140	37	0	0
	191	201	191	201	0	0	191	201	0	0
	107	166	106	166	0	0	110	164	3	2

Continued to next page

Continuation of Table A.1

Query	Ground Truth		SMSURF				SMCRN			
	x	y	x	y	Δx	Δy	x	y	Δx	Δy
Monkey	35	301	38	301	3	0	319	277	284	24
	246	27	246	27	0	0	246	27	0	0
	24	89	24	89	0	0	24	89	0	0
Necklace	71	107	71	107	0	0	71	107	0	0
	159	86	159	86	0	0	159	86	0	0
	202	332	202	333	0	1	208	332	6	0
Oni	64	244	65	246	1	2	67	245	3	1
	352	220	352	220	0	0	352	220	0	0
	146	205	146	205	0	0	146	205	0	0
Orihime	36	274	36	274	0	0	36	274	0	0
	18	94	18	94	0	0	18	94	0	0
	286	239	286	239	0	0	286	239	0	0
Sheep	29	75	29	75	0	0	29	75	0	0
	50	209	50	209	0	0	50	209	0	0
	230	135	230	135	0	0	230	135	0	0
Snail	84	13	84	13	0	0	84	13	0	0
	337	201	337	201	0	0	337	201	0	0
	68	300	68	300	0	0	68	300	0	0
Snowman1	99	8	99	8	0	0	99	8	0	0
	216	5	216	5	0	0	216	5	0	0
	322	26	322	26	0	0	322	26	0	0

Continued to next page

Continuation of Table A.1

Query	Ground Truth		SMSURF				SMCRN			
	x	y	x	y	Δx	Δy	x	y	Δx	Δy
Snowman2	15	81	358	80	343	1	3	82	12	1
	318	276	318	276	0	0	318	276	0	0
	292	284	292	284	0	0	292	284	0	0
Sofa	85	336	85	336	0	0	85	336	0	0
	116	187	116	187	0	0	116	187	0	0
	24	131	24	131	0	0	24	131	0	0
Sunflower	218	100	218	100	0	0	218	100	0	0
	201	303	201	303	0	0	201	303	0	0
	339	19	339	19	0	0	339	19	0	0
Sunflower2	140	312	140	312	0	0	140	312	0	0
	152	190	152	190	0	0	152	190	0	0
	172	193	172	193	0	0	172	193	0	0
Suzanne	271	308	271	308	0	0	271	308	0	0
	45	24	45	24	0	0	45	24	0	0
	167	152	167	155	0	3	167	155	0	3
Tiger	206	249	206	249	0	0	206	249	0	0
	254	72	254	72	0	0	254	72	0	0
	178	134	177	133	1	1	184	127	6	7

Table A.2: Time Comparison of SMSURF and SMCRN (in seconds)

Query	SMSURF	SMCRN
Bike	4378.8	1108.48
	3655.95	796.326
	1133.08	291.231
Bird	280.439	91.705
	2063.1	679.925
	3622.72	1124.02
Bugs	4360.49	1809.67
	5600.94	2263.9
	1260.98	420.916
Bus	8874.65	1762.07
	6217.19	1738.17
	4746.37	1106.48
Cake	11334.7	1853.69
	5907.81	1299.04
	14050.4	2162.45
Cello	3979.05	1557.93
	1692.78	678.565
	653.758	231.908
Chicken	4846.74	1552.17
	5721.5	2019.18
	2962.64	902.437
Cow	3392	1263.39
	5135.44	2078.56
	6975.3	2575.22
Dino	308.124	96.58
	6228.05	1609.26
	261.814	88.29
Flower	1144.46	363.791
	2452.24	1010.7
	1780.48	601.85

Continued to next page

Continuation of Table A.2

Query	SMSURF	SMCRN
Frog	2628.85	732.15
	7641.97	2065.42
	4147.17	1088.16
Hikoboshi	6871.78	1850.58
	5512.13	1464.21
	8062.06	2432.88
Horse	7106.68	2162.6
	4520.52	1388.92
	1126.18	383.455
Incenseholder	5587.77	1769.33
	9187.36	2360.89
	9255.19	2321.44
Kimono	2534.05	962.235
	4418.67	1384.44
	2993.24	1109.22
Mini Dragon	1573.25	456.324
	5021.51	1664.17
	3946.56	1318.7
Mochi	10155.5	2731.74
	6946.99	1839.38
	7258.37	1925.86
Monkey	4041.49	1572.28
	4643.25	1707.33
	4900.55	1769.85
Necklace	6945.09	2336.19
	4583.02	1406.48
	5860.77	1718.39
Oni	2532.97	940.906
	5070.14	1744.84
	5232.91	1807.01
Orihime	8902.56	2576.27
	8521.95	1911.42
	6691.23	1888.74

Continued to next page

Continuation of Table A.2

Query	SMSURF	SMCRN
Sheep	3226.11	1226.54
	5870.74	1905.65
	5587.72	2124.73
Snail	1990.6	551.44
	2190.83	604.387
	5916.22	1665.31
Snowman1	2201.29	761.238
	2631.08	774.406
	3045.5	920.861
Snowman2	7669.29	2681.75
	5805.37	2449.03
	4488.03	1254.2
Sofa	5074.53	1598.94
	3726.48	1061.43
	10331	2417.94
Sunflower	5540.76	1562.21
	2903.2	921.282
	4896.7	1193.04
Sunflower2	6267.78	1814.95
	6329.53	2118.38
	6585.1	2253.31
Suzanne	6711.29	2060.64
	1374.74	395.43
	1714.62	469.342
Tiger	893.825	233.992
	5153.31	980.466
	1333.66	286.92

Appendix B

Result of Scale-Orientation Invariant Shape Context on Skeleton

Table B.1: Shape Context including skeleton result on Template Pose 2 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**18.000**	**16.833**	**16.905**	**16.708**	**16.286**	**17.952**	**16.476**
Student	17.111	16.417	16.024	15.509	16.310	15.533	15.488
Bunny	14.521	13.155	13.536	13.381	13.810	14.161	13.338
Cherry	15.542	14.408	14.286	13.768	13.420	14.503	14.252
Echu	15.778	14.516	14.158	14.589	14.925	15.878	14.933

Table B.2: Shape Context including skeleton result on Template Pose 2 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**17.133**	**18.333**	**18.103**	**15.744**	**16.767**	**17.133**	**16.310**
Student	16.485	17.133	17.238	14.982	15.881	15.743	15.417
Bunny	13.977	15.000	15.016	12.987	14.003	14.045	13.082
Cherry	14.975	16.350	16.036	14.750	14.906	15.233	13.901
Echu	15.233	16.533	16.143	14.718	15.500	15.200	14.069

Table B.3: Shape Context including skeleton result on Template Pose 2 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**17.000**	13.966	**17.963**	**15.505**	**16.500**	**15.185**	**16.667**
Student	15.942	**14.892**	16.907	14.292	15.611	14.374	16.060
Bunny	13.197	13.440	14.889	12.671	13.633	12.642	13.714
Cherry	13.020	13.787	15.819	14.036	14.293	13.503	13.767
Echu	14.403	13.973	16.148	13.577	14.524	13.710	16.000

Table B.4: Shape Context including skeleton result on Template Pose 2 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**15.893**	**17.833**	**17.889**	**17.000**	**14.935**	**15.589**	**16.400**
Student	11.721	15.917	16.425	15.438	13.909	15.011	14.700
Bunny	13.083	13.563	14.439	13.294	11.449	12.611	13.500
Cherry	13.037	15.000	15.317	13.612	13.219	13.447	13.803
Echu	13.347	14.917	15.611	14.548	13.027	14.025	13.971

Table B.5: Shape Context including skeleton result on Template Pose 2 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**15.238**	**15.762**	**16.133**	**15.944**	**16.722**	**17.778**	**16.500**
Student	14.643	15.022	14.979	14.958	15.426	16.635	15.611
Bunny	12.856	13.491	13.450	13.042	13.345	14.133	12.785
Cherry	14.207	13.949	13.848	13.875	14.455	14.965	14.000
Echu	13.779	13.650	13.466	14.109	14.594	15.267	14.444

Table B.6: Shape Context including skeleton result on Template Pose 2 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	**17.556**	**17.000**	**14.687**	**15.867**	**16.792**	**16.042**	**16.778**
Student	16.500	15.691	14.011	15.400	15.875	15.548	15.500
Bunny	13.958	13.857	12.463	12.900	13.906	13.009	13.500
Cherry	14.089	15.000	13.406	13.824	14.906	13.695	14.024
Echu	15.333	14.796	13.039	14.867	15.125	13.282	14.778

Table B.7: Shape Context including skeleton result on Student Pose 1 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	16.775	15.427	16.229	16.188	15.781	14.646	15.099
Student	**19.833**	**17.564**	**19.222**	**20.083**	**18.542**	**16.172**	**17.638**
Bunny	16.143	14.544	15.650	16.694	14.450	13.431	14.524
Cherry	17.567	16.871	17.013	18.357	15.267	15.929	15.015
Echu	18.177	15.718	17.467	18.406	17.521	14.813	17.267

Table B.8: Shape Context including skeleton result on Student Pose 1 with x = 30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	14.996	17.952	17.375	17.425	14.571	14.895	13.333
Student	15.349	**19.371**	**19.114**	**20.108**	**15.547**	14.196	**14.833**
Bunny	14.328	16.289	15.877	17.711	13.361	14.764	12.257
Cherry	**16.163**	18.180	16.634	18.550	15.210	14.547	13.866
Echu	15.076	17.902	17.643	18.023	14.737	**15.152**	14.667

Table B.9: Shape Context including skeleton result on Student Pose 1 with x = -30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	16.738	18.222	17.061	16.388	16.876	17.337	16.838
Student	**18.611**	**20.086**	**18.851**	16.333	**18.182**	**20.778**	**18.759**
Bunny	16.411	17.259	16.083	15.807	15.693	16.819	16.741
Cherry	16.663	19.528	17.890	15.234	15.251	17.070	16.965
Echu	17.933	18.764	17.955	**17.333**	17.079	19.617	18.171

Table B.10: Shape Context including skeleton result on Student Pose 2 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	18.128	18.710	18.591	19.386	17.782	17.387	18.473
Student	**21.101**	**22.200**	**20.808**	**22.950**	**20.506**	**21.254**	**22.567**
Bunny	17.524	18.244	16.756	19.242	16.640	17.259	17.080
Cherry	19.476	18.087	18.709	20.857	18.511	18.779	20.714
Echu	18.873	19.602	18.375	21.091	18.594	18.975	20.028

Table B.11: Shape Context including skeleton result on Student Pose 2 with x = 30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	17.557	18.462	17.000	16.408	18.092	17.071	16.677
Student	**19.215**	**21.667**	**18.546**	**18.451**	19.628	**18.690**	**19.144**
Bunny	16.965	17.735	15.450	16.181	16.505	16.020	15.366
Cherry	17.977	20.560	17.171	17.998	**19.906**	18.113	16.968
Echu	18.598	20.096	17.251	16.803	18.979	17.596	17.700

Table B.12: Shape Context including skeleton result on Student Pose 2 with x = -30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	16.993	18.340	19.091	17.707	19.153	19.250	18.265
Student	**18.977**	**20.271**	**20.960**	18.743	**22.059**	**19.742**	**21.056**
Bunny	16.554	17.676	17.980	16.849	18.175	18.375	17.700
Cherry	18.434	18.283	18.809	**18.917**	20.151	18.964	20.194
Echu	17.947	18.721	19.641	18.430	19.761	18.728	19.296

Table B.13: Shape Context including skeleton result on Student Pose 3 with x = 0

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	18.558	17.590	14.889	15.812	17.966	17.020	15.495
Student	**21.208**	**19.644**	**17.150**	**17.318**	**21.455**	**18.697**	**17.337**
Bunny	17.096	15.536	15.078	14.450	16.498	16.454	14.769
Cherry	19.825	17.212	16.069	16.643	19.130	18.302	16.427
Echu	19.050	18.252	16.019	15.846	19.636	17.075	16.202

Table B.14: Shape Context including skeleton result on Student Pose 3 with x = 30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	19.375	14.363	17.075	14.236	16.976	16.931	15.316
Student	**20.806**	**16.525**	**19.200**	**17.500**	18.297	**19.611**	**18.861**
Bunny	17.977	14.439	15.863	13.622	16.199	15.669	15.409
Cherry	19.696	16.388	17.573	16.968	**18.651**	16.413	17.583
Echu	17.683	15.829	16.332	15.292	17.240	17.754	17.917

Table B.15: Shape Context including skeleton result on Student Pose 3 with x = -30

Model	Query						
	z = 0	z = 30	z = 60	z = 90	z = -30	z = -60	z = -90
Template	17.297	18.596	18.250	20.556	18.304	17.894	21.575
Student	**20.091**	**19.111**	**21.213**	**23.444**	**21.676**	**21.120**	**25.100**
Bunny	16.472	16.708	18.444	19.074	18.278	17.361	21.033
Cherry	17.181	17.365	20.107	21.079	18.000	19.835	23.413
Echu	18.939	18.855	19.056	21.630	18.702	18.713	22.650

Bibliography

[1] Y. Gao, M. Wang, D. Tao, R. Ji, and Q. Dai, "3-d object retrieval and recognition with hypergraph analysis," *IEEE Trans. Image Processing*, vol. 21, no. 9, pp. 4290 – 4303, 2012.

[2] K. Lu, R. Ji, J. Tang, and Y. Gao, "Learning-based bipartite graph matching for view-based 3d model retrieval," *IEEE Trans. Image Processing*, vol. 23, no. 10, pp. 4553 –4563, 2014.

[3] M. Wang, Y. Gao, K. Lu, and Y. Rui, "View-based discriminative probabilistic modeling for 3d object retrieval and recognition," *IEEE Trans. Image Processing*, vol. 22, no. 4, pp. 1395 – 1407, 2013.

[4] S. A. A. Shah, M. Bennamoun, F. Boussaid, and A. A. El-Sallam, "3d-div: A novel local surface descriptor for feature matching and pairwise range image registration," in *IEEE International Conference on Image Processing (ICIP)*, 2013.

[5] C. Feinen, J. Czajkowska, M. Grzegorzek, and L. J. Latecki, "3d object retrieval by 3d curve matching," in *IEEE International Conference on Image Processing (ICIP)*, 2014.

[6] P. Papadakis, I. Pratikakis, T. Theoharis, and S. Perantonis, "PANORAMA: a 3D shape descriptor based on panoramic views for unsupervised 3D object retrieval," *International Journal of Computer Vision*, vol. 89, no. 2, pp. 177–192, 2010.

[7] B. Gong, J. Liu, X. Wang, and X. Tang, "Learning semantic signatures for 3D object retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 369–377, 2013.

[8] A. Tankus, N. Sochen, and Y. Yeshurun, "A new perspective [on] shape from shading," in *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV'03)*, 2003.

[9] E. Prados and O. Faugeras, "Shape from shading: a well-posed problem?" in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005.

[10] J.-D. Durou, M. Falcone, and M. Sagona, "Numerical methods for shape-from-shading: A new survey with benchmarks," *Computer Vision and Image Understanding*, vol. 109, no. 1, pp. 22–43, 2007.

[11] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape from shading: A survey," *IEEE Trans. Pattern Analysis and Machine Learning*, vol. 21, no. 8, pp. 690–706, 1999.

[12] Z. Zhang, Y. Gao, and T. Caelli, "Colour adjustment and specular removal for non-uniform shape from shading," in *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2010.

[13] M. Rahman, T. W. Chow, and S.-Y. Cho, "A segmentation based approach for shape recovery from multi-color images," in *International Conference in Control, Automation, Robotics and Vision*, 2010.

[14] S. Mahmoudi and M. Daoudi, "3d models retrieval by using characteristic views," in *Proceedings of International Conference on Pattern Recognition*, 2002.

[15] L. J. Latecki and R. Lakämper, "Application of planar shape comparison to object retrieval in image databases," *Pattern Recognition*, vol. 35, no. 1, pp. 15–29, 2002.

[16] C. Yang, O. Tiebe, P. Pietsch, C. Feinen, U. Kelter, and M. Grzegorzek, "Shape-based object retrieval by contour segment matching," in *IEEE International Conference on Image Processing*, 2014.

[17] A.-A. Liu, W.-Z. Nie, Y. Gao, and Y.-T. Su, "Multi-modal clique-graph matching for view-based 3d model retrieval," *IEEE Trans. Image Processing*, vol. 25, no. 5, pp. 2103–2116, 2016.

[18] H. Shin and T. Igarashi, "Magic canvas: Interactive design of a 3-d scene prototype from freehand sketches," in *Graphics Interface 2007*, 2007.

[19] Z.Lian, A.Godil, B.Bustos, M.Daoudi, J.Hermans, S.Kawamura, Y.Kurita, G.Lavoué, H.V.Nguyen, R.Ohbuchi, Y.Ohkita, Y.Ohnishi, F.Porikli, M.Reuter, I.Sipiran, D.Smeets, P.suetens, H.Tabia, and D.Vandermeulen, "SHREC'11 track: Shape retrieval on non-rigid 3D watertight meshes," in *Eugrographics Workshop on 3D Object Retrieval*, 2011.

[20] ——, "A comparison of methods for non-rigid 3D shape retrieval," *Pattern Recognition*, vol. 46, no. 1, pp. 449 – 461, 2013.

[21] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R. Guler, L. Lai, C. Li, H. Li, F. Limberger, R. Martin, R. Nakanishi, A. Neto, L. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, and R. Wilson, "SHREC'15 track: Non-rigid 3D shape retrieval," in *Eurographics Workshop on 3D Object Retrieval*, 2015.

[22] Z. Lian, A. Godil, X. Sun, and J. Xiao, "CM-BOF: visual similarity-based 3D shape retrieval using Clock Matching and Bag-of-Features," *Machine Vision and Applications*, vol. 24, pp. 1685–1704, 2013.

[23] P. Li, H. Ma, and A. Ming, "View-based 3D model retrieval using two-level spatial structure," in *IEEE Conference on Image Processing*, 2011.

[24] A. Bronstein, M. Bronstein, L. Guibas, and M. Ovsjanikov, "Shape Google: Geometric words and expressions for invariant shape retrieval," *ACM. Trans. Graphics*, vol. 30, no. 1, 2011.

[25] X. Bai, X. Wang, L. J. Latecki, W. Liu, and Z. Tu, "Active skeleton for non-rigid object detection," in *IEEE International Conference on Computer Vision*, 2009.

[26] Y. N. Wu, Z. Si, H. Gong, and S.-C. Zhu, "Learning active basis model for object detection and recognition," *International Journal of Computer Vision*, vol. 90, no. 2, pp. 198–235, 2010.

[27] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.

[28] T. Zhang and C. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[29] L. Lam, S.-W. Lee, and C. Y. Suen, "Thinning methodologies-a comprehensive survey," *IEEE Trans. Pattern Analysis and Machine Learning*, vol. 14, no. 9, pp. 869–885, 1992.

[30] T. K. Dey and J. Sun, "Defining and computing curve-skeletons with medial geodesic function," in *SGP '06 Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006, pp. 143–152.

[31] M. Couprie, D. Coeurjolly, and R. Zrour, "Discrete bisector function and euclidean skeleton in 2d and 3d," *Image and Vision Computing*, vol. 25, no. 10, pp. 1543–1556, 2007.

[32] W. Liu, H. Jiang, X. Bai, G. Tan, C. Wang, W. Liu, and K. Cai, "Distance transform-based skeleton extraction and its applications in sensor networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1763–1772, 2013.

[33] W.-P. Choi, K.-M. Lam, and W.-C. Siu, "Extraction of the euclidean skeleton based on a connectivity criterion," *Pattern Recognition*, vol. 36, no. 3, pp. 721–729, 2003.

[34] A. Lieutier, "Any open bounded subset of $\mathbb{r}^n$ has the same homotopy type than its medial axis," *Computer-Aided Design*, vol. 36, no. 11, pp. 1029–1046, 2004.

[35] X. Bai, L. J. Latecki, and W.-Y. Liu, "Skeleton pruning by contour partitioning with discrete curve evolution," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 449–462, 2007.

[36] X. You and Y. Y. Tang, "Wavelet-based approach to character skeleton," *IEEE Trans. Image Processing*, vol. 16, no. 5, pp. 1220–1231, 2007.

[37] N. Widynski, A. Moevus, and M. Mignotte, "Local symmetry detection in natural images using a particle filtering approach," *IEEE Trans. Image Processing*, vol. 23, no. 12, pp. 5309–5322, 2014.

[38] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Multiscale symmetric part detection and grouping," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 117–134, 2013.

[39] X. Du, S. Zhu, and J. Li, "Skeleton Extraction via Structure-Adaptive Anisotropic Filtering," in *Proc. International Conference on Internet Multimedia Computing and Service (ICMCS'14)*, 2014.

[40] Q. Li, X. Bai, and W. Liu, "Skeletonization of gray-scale image from incomplete boundaries," in *Proc. International Conference on Image Processing (ICIP'08)*, 2008.

[41] L. J. Latecki, Q. nan Li, X. Bai, and W. yu Liu, "Skeletonization using ssm of the distance transform," in *Proc. International Conference on Image Processing (ICIP'07)*, 2007.

[42] Z. Tu, "Auto-context and its application for high-level vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[43] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[44] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISAPP'09*. INSTICC Press, 2009, pp. 331–340.

[45] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[46] L. Li and C. L. Tan, "Recognizing planar symbols with severe perspective deformation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 755–762, 2010.

[47] Z. Luo, D. Luo, X. Fan, X. Zhou, and Q. Jia, "A shape descriptor based on new projective invariants," in *IEEE International Conference on Image Processing (ICIP)*, 2013.

[48] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2004*, vol. 23, no. 3, pp. 309–314, 2004.

[49] D. Pollock, "Smoothing with cubic splines," Queen Mary and Westfield College, The University of London, Tech. Rep., 1993.

[50] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[51] J. Tangelder and R.C.Veltkamp, "A survey of content based 3D shape retrieval methods," *Multimedia Tools and Applications*, vol. 39, no. 3, pp. 441–471, 2008.

[52] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[53] J. Pan and J. J. Zhang, "Sketch-Based Skeleton-Driven 2d Animation and Motion Capture," *Trans. on Edutainment VI*, pp. 164–181, 2011.

[54] S. Messmer, S. Fleischmann, and O. Sorkine-Hornung, “Animato: 2D Shape Deformation and Animation on Mobile Devices,” in *Proc. SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*, 2016.

[55] T. Igarashi, S. Matsuoka, and H. Tanaka, “Teddy: A Sketching Interface for 3D Freeform Design,” in *Proc. Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH’99)*, 1999.

[56] Y. Gingold, T. Igarashi, and D. Zorin, “Structured annotations for 2D-to-3D modeling,” *Transactions on Graphics*, vol. 28, no. 5, 2009.

[57] N. R. Twarog, M. F.Tappen, and E. H. Adelson, “Playing with puffball: simple scale-invariant inflation for use in vision and graphics,” in *Symposium on Applied Perception*, 2012.

[58] X. Bai and L. J. Latecki, “Path Similarity Skeleton Graph Matching,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, pp. 1282–1292, 2008.

[59] M. van Eede, D. Macrini, A. Telea, C. Sminchisescu, and S. Dickinson, “Canonical skeletons for shape matching,” in *IEEE International Conference on Pattern Recognition*, 2006.

[60] D. Pickup, X. Sun, P. L. Rosin, and R. R. Martin, “Skeleton-based canonical forms for non-rigid 3D shape retrieval,” *Computational Visual Media*, vol. 2, no. 3, pp. 232–243, 2016.

[61] S. Lin, Y. Guo, Y. Liang, Q. Chen, and Y. Wu, “3D model retrieval based on skeleton,” in *IEEE International Conference on Networking, Architecture and Storage*, 2015.

[62] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik, “Using Contours to Detect and Localize Junctions in Natural Images,” in *Proc. Computer Vision and Pattern Recognition (CVPR’08)*, 2008.

[63] D. R.Martin, C. C.Fowlkes, and J. Malik, “Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[64] L. Prasad, “Rectification of the chordal axis transform skeleton and criteria for shape decomposition,” *Image and Vision Computing*, vol. 25, pp. 1557–1571, 2007.

[65] A. Y.Ng, M. I. Jordan, and Y. Weiss, “On Spectral Clustering: Analysis and an algorithm,” in *International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01)*, 2001.

[66] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A Database of Human Segmented Natural Images and Its Application to Evaluation Segmentation Algorithms and Measuring Ecological Statistics,” in *Proc. International Conference on Computer Vision (ICCV'01)*, 2001.

[67] L. Fei-Fei, R. Fergus, and P. Perona, “Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories,” in *Proc. Conference on Computer Vision and Pattern Recognition (CVPR'04)*, 2004.

[68] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 24, pp. 509–522, 2002.

[69] S. Belongie and J. Malik, “Matching with shape contexts,” in *Content-based Access of Image and Video Libraries*, 2002.