Real-time Shape Retrieval for Robotics using Skip Tri-Grams

Yi Li and Konstantinos Bitsakos
University of Maryland Institute for Advanced Computer Studies
University of Maryland, College Park MD 20742
{liyi, kbits}@umiacs.umd.edu

Cornelia Fermuller and Yiannis Aloimonos
Center for Automation Research
University of Maryland, College Park MD 20742
{fer,yiannis}@cfar.umd.edu

Abstract

The real time requirement is an additional constraint on many intelligent applications in robotics, such as shape recognition and retrieval using a mobile robot platform. In this paper, we present a scalable approach for efficiently retrieving closed contour shapes. The contour of an object is represented by piecewise linear segments. A skip Tri-Gram is obtained by selecting three segments in the clockwise order while allowing a constant number of segments to be "skipped" in between. The main idea is to use skip Tri-Grams of the segments to implicitly encode the distant dependency of the shape. All skip Tri-Grams are used for efficiently retrieving closed contour shapes without pairwise matching feature points from two shapes. The retrieval is at least an order of magnitude faster than other state-of-the-art algorithms. We score 80% in the Bullseye retrieval test on the whole MPEG 7 shape dataset [11]. We further test the algorithm using a mobile robot platform in an indoor environment. 8 objects are used for testing from different viewing directions, and we achieve 82% accuracy.

1. Introduction

The intelligent systems try to match human vision in various tasks such as object recognition and shape retrieval. When it comes to robots exploring the world, the additional constraint of real-time computation limits the number of algorithms that can be used. An efficient and effective memory organization is necessary for the robot so that it can store and retrieve objects. In this paper, we present a shape representation, called *"Shape Memory"*, for this purpose.

This representation approximates the contour of an object as many piecewise linear segments. Starting from any segment, we consider a "sentence" of segments in clockwise order. Each group of three consecutive segments is called a Tri-Gram. Tri-Grams are proposed by Shannon who used them to reproduce English written text [19]. A skip Tri-Gram [4] is a generalization of a Tri-Gram, which is obtained by selecting three segments in order while allowing some segments to be "skipped" in between. In our implementation, we skip a constant number of segments. Skip Tri-Grams are used to implicitly encode the distant geometric dependencies of sequential contour segments. We store the skip Tri-Grams into a generic and compact array that can be used for retrieval and recognition.

The advantages of our representation are:

1. Shape Memory is a very efficient representation for retrieving closed contour shapes without pairwise matching feature points from two shapes. Retrieval is more than an order of magnitude faster than other state-of-the-art algorithms, with reasonable accuracy.

2. Shape Memory is scalable because the memory size is linear with respect to the size of the training set. When new shapes are added, no re-training and changing of the existing memory structure are required. Therefore, both the training and the testing are very efficient.

3. Shape Memory has the ability to memorize shape instances. We demonstrate this ability by reconstructing shapes from contour segments.

The paper is organized as follows. A brief review of related topics is presented in Sec. 2, and our representation is described in Sec. 3. Sec. 4 shows experiments including shape reconstruction and shape retrieval using a mobile platform. We conclude the paper in Sec. 5.

2. Related Work

Recognizing objects using mobile platforms recently regains the attention in related fields [21]. Shape is one of the key features of the objects. There are two major ways to characterize shapes: landmark points and codebooks of boundary curves. Landmark point based approaches are often applied to close contours. Points are usually obtained by uniformly sampling the boundaries of the silhouettes. Then each point is described by its spatial relationships with respect to all other points. The best known representation in this category is Shape Context [2]. To handle articulated objects, Ling and Jacobs [12] extend Shape Context using the Inner Distance. Biswas *et al.* [3] use the Inner Distance for fast shape retrieval. Most existing algorithms in this category are computationally demanding, partially because they are based on pairwise comparison. Given two shapes, many algorithms need to compare every feature point from one shape against all feature points from another one. Furthermore, these point descriptors are often used in a nearest-neighbor classification framework [2], [25].

The idea of representing shapes using codewords dates back to the 70s. Chain Code [8] uses eight directions to describe the relationships of adjacent pixels. Fu [9] describes boundary strings with grammars, and uses these grammars to parse the strings. Recently, codebooks are built from class-discriminative boundary fragments [15, 16, 20, 7, 24]. In these approaches, positions of object centroids often need to be stored explicitly and used as additional geometric constraints. Our approach regards object boundaries as sequences of codewords, and a major difference between our approach and previous work is that we also describe the distant dependencies of shape using skip Tri-Grams [4] of boundary segments.

There has been much work on organizing and retrieving shapes using indexing techniques. Given a dataset of objects, shape indexing is usually applied to store and retrieve each object from the dataset and compare it against other objects in the scene. Geometric Hashing [23] uses a hash table to store the descriptors for recognition. Rigoutsos and Hummel [17] develop a Bayesian approach based on hashing. Clemens and Jacobs use the hashing technique to index 3D models from descriptors in 2D images [5]. Our approach is different in two ways. Our descriptor encodes distant dependency, and object retrieval is performed using a probabilistic voting framework.

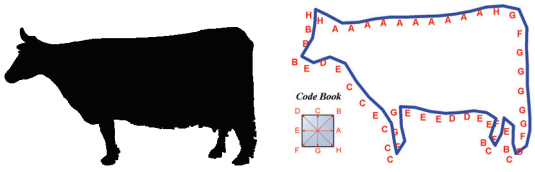
3. Representing Contours using Tri-Grams

Our goal is to have a shape representation which stores local relationships, while still retains distant geometric information. Instead of using additional constraints such as vectors to object centroid's position, we propose using the skip Tri-Gram model to achieve this goal.

By reconstructing the original shape, we show that our skip Tri-Gram based approach retains the distant dependency without explicitly storing other information. Then we present an algorithm for shape retrieval using a probabilistic voting framework. We call the skip Tri-Gram based memory representation *"Shape Memory"*, because it constitutes a generic way to store and retrieve arbitrary shapes. In the subsequent discussion, we borrow terms from the NLP community such as codewords, codebook, and string.

3.1. From Shapes to Strings

Shapes are represented by pieces of contour. Given a silhouette, it is natural to uniformly sample the boundary. We traverse the silhouette, and simply quantize the orientation of the line segments between adjacent sample points. We use eight different orientations, denoted by the English characters "A" to "H", as codewords, which together make up the codebook. Figs. 1a and 1b display the silhouettes before and after quantization. Fig. 2 shows all the sentences in the MPEG 7 shape dataset using this representation.



a b

Figure 1. Quantize a shape to a sequence of codewords (b). We uniformly sample 50 points for each shape, and quantize the orientation of the adjacent points using the codebook.

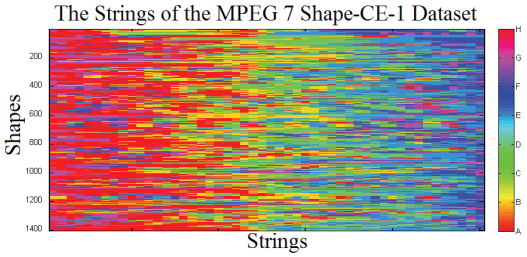


Figure 2. The strings of the shapes in the MPEG 7 dataset. Each row is a string that represents the corresponding shape in the dataset, where codewords are visualized using colors (best viewed in color).

3.2. Brief Introduction to the Skip Tri-Gram Model

Tri-Grams of a string are a set of tuples of all possible adjacent codewords in the string[1]. For $N = 1, 2, 3$, they are called "unigram", "bigram", and "trigram", respectively. Table 1a gives an example of all the "unigrams", "bigrams", "trigrams" and "4-grams" of the string "ABCABCABDABD". We use skip Tri-Grams to represent the whole string and encode distant shape dependency.

Table 1. Examples of N-Grams and skip Tri-Grams: a) N-Grams (N=1,2,3,4) of the string "ABCABCABDABD"; b) skip Tri-Grams of the string "ABCDEFG" (distance = 0,1,2).

unigram	bigram	trigram	4-gram
A	AB	ABC	ABCA
B	BC	BCA	BCAB
C	CA	CAB	CABC
D	BD	ABD	CABD
	DA	BDA	ABDA
		DAB	BDAB
			DABD

0	1	2
ABC	ACE	ADG
BCD	BDF	BEA
CDE	CEG	CFB
DEF	DFA	DGC
EFG	EGB	EAD
FGA	FAC	FBE
GAB	GBD	GCF

3.2.1 Definition

- **The skip Tri-Grams at distance j**: Starting from any codeword of the string S, a skip Tri-Gram is obtained by selecting two more codewords in the sentence order while skipping exactly j codewords in between (Table 1b).

3.3. Represent Shapes using Skip Tri-Grams

Our representation of a set of shapes is a 3D binary array $\mathcal{SM}$. As shown in Fig. 3, each vertical slice through $\mathcal{SM}$ is a 2D binary array $\mathcal{SH}$ which represents a single shape. Each row in $\mathcal{SH}$ represents a Tri-Gram. Each column represents the same Tri-Gram at different distances. We first present our notation, and then we describe the construction of $\mathcal{SM}$.

3.3.1 Notation

- S: a shape string.
- L_S: the number of landmark points when sampling a boundary. It is also the length of the shape string.
- $S[i]$: the i^{th} codeword in S.
- TG_S^j: the set of all Tri-Grams of the string S at distance j.
- $\mathcal{J}$: the maximum distance. Usually $\mathcal{J} \leq \lfloor L_S/2 \rfloor$.
- g: a Tri-Gram.
- CB: the codebook of size $|CB|$.
- $w_i, i = 1, ..., |CB|$: codewords in the codebook CB.

[1]We regard the last codeword in the string to be the next to the first one.

3.3.2 Representing a Single Shape

We organize TG_S^j ($j = 0, .., \mathcal{J}$) into a 2D binary array $\mathcal{SH}$ to represent a shape S. For each distance j, the number of possible trigrams in TG_S^j is $|CB|^3$. Thus we represent S by a binary vector of length $|CB|^3$, whose entries are set to "1" if the corresponding trigram occurs in TG_S^j.

The $\mathcal{J}$ vectors are combined into a $|CB|^3 \times \mathcal{J}$ matrix $\mathcal{SH}$ (Fig. 3). Each row represents all possible trigrams from "AAA" to "HHH", while each column represents the distance from "0" to "$\mathcal{J}$ ". This is our representation for an individual shape. To simplify the notation, we use $\mathcal{SH}(g, j)$ to access the cell in the structure corresponding to a skip Tri-Gram g at distance j.

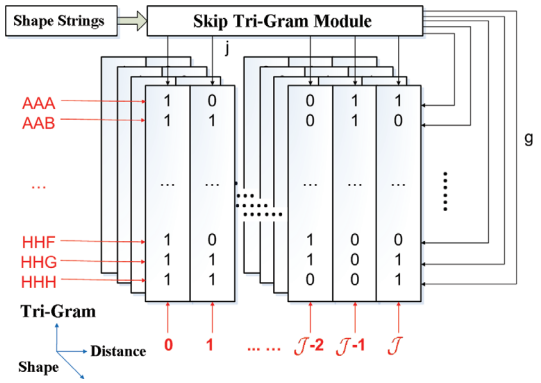


Figure 3. Shape Memory. A shape is represented by a set of skip Tri-Grams at different distances. Here we illustrate a vertical slice ($\mathcal{SH}$) in the memory, which represents a single object. The entry (i, g, j) in the memory is set to "1" when the object i has the Tri-Gram g at distance j (black arrows). The entries in this slice can be directly accessed using Tri-Gram g (row) at distance j (column) (red arrows).

3.3.3 Memory Organization of All Shapes

Our goal is to organize shapes in the training dataset into a compact 3D representation. We have a set of object shapes with category labels, each represented by a $\mathcal{SH}$. Intuitively, shapes in the same category need to be placed together. To sort the shapes of a category, we first compute the *edit distance* [10] between two shape strings using dynamic programming. Then we apply a greedy algorithm in the same category[2].

[2]We choose edit distance because it is fast, but it can be replaced by any pairwise similarity measurement.

For each sorted string of a category, we compute the corresponding $\mathcal{SH}$ using the representation in Sec. 3.3.2. We organize the $\mathcal{SH}$s of a category based on their order (see the "Shape" dimension in Fig. 3). We repeat this procedure for different categories, and the result is a $|CB|^3 \times \mathcal{J} \times N$ binary array. We denote this memory organization as Shape Memory $\mathcal{SM}$.

Pairwise similarity measurements are only computed for shapes in the same category when building the Shape Memory. Given a set of N closed boundary shapes grouped in C categories with N_C shapes per category, the time complexity of sorting the memory is $O(CN_C^2)$, which is $O(N_C N)$ because of $N = CN_C$. When N_C is small (*e.g.*, $N_C = 20$ in the MPEG 7 shape dataset), the sorting algorithm is fast. The complexity of building the memory content is $O(N_C N + |CB|^3 \mathcal{J} N)$. Given a fixed codebook and the maximum distance (*e.g.* $|CB| = 8$ and $\mathcal{J} = 14$ in our implementation), the complexity scales linearly with the number of training shapes N.

3.4. Algorithm for Reconstructing Shapes from Contour Segments

Given the $\mathcal{SH}$ of an arbitrary shape string S, we use a depth first search (DFS) approach to reconstruct S. Our goal is to generate a string S of length L_S that satisfies $\mathcal{SH}$. The necessary condition for a string S to satisfy $\mathcal{SH}$ is that the corresponding entries of all possible skip Tri-Grams of S are "1". Assume that we have a substring that satisfies the necessary condition and we want to generate the next codeword[3]. For any possible choice w, we only need to check all Tri-Grams whose last codeword is w in the new string obtained by appending w to the substring. This approach is equivalent to a depth first search procedure. Algorithm 1 describes this procedure.

Since a string represents a shape, the last codeword in the string is next to the first one. Depending on the starting codeword, Algorithm 1 might return multiple solutions that are identical after shifting. The time complexity for the depth first search depends on $\mathcal{J}$. In our experiment, it only takes a fraction of a second to generate the string.

3.5. Shape Retrieval using Shape Memory

We use a simple probabilistic voting scheme for shape retrieval. $\mathcal{SM}$ is considered as a lookup table for all training shapes. We cast probability votes to existing shapes that contain skip Tri-Grams of the test string. Note that this approach does not require pairwise matching when retrieving shapes. Therefore, shape retrieval is very fast.

[3]The sufficient condition can be easily checked after reconstructing the whole string. Thus, only the necessary condition is described in this paper.

Algorithm 1 Shape Reconstruction

Input	:	
$\mathcal{SH}$	:	The representation of shape S
Output	:	
S_{cur}	:	The reconstructed string
Procedure	:	

1) Randomly initialize $S_{init} \leftarrow w_1 w_2 w_3 w_4$ using the Tri-Grams $w_1 w_2 w_3$ and $w_2 w_3 w_4$ of S at distance 0;
2) execute function verify(S_{init}, 1)

FUNCTION verify(S_{cur}, n)
IF $n \geq L_S$
  PRINT S_{cur} and RETURN;
ELSE
  $O \leftarrow \{w_1, ..., w_8\}$;
  For $w \in CB$
    Obtain S_{guess} by appending w to S_{cur};
    FOR possible distance j
      $i_1 = |S_{guess}| - 2j - 2$;
      $i_2 = |S_{guess}| - j - 1$;
      $g = S_{guess}[i_1] S_{guess}[i_2] w$;
      IF $\mathcal{SH}(g, j) == 0$
        $O \leftarrow O - \{w\}$;
  $\forall\, w \in O$
    Obtain S_{next} by appending w to S_{cur};
    Verify(S_{next}, n+1);

3.5.1 The Algorithm

Using the $\mathcal{SM}$ as the lookup table, each skip Tri-Gram g of the test string S casts probabilistic votes for all possible shapes that have g. In our representation, shapes of a category are sorted based on their similarities. Therefore, we use a Gaussian kernel to approximate the probability distribution of the similarity measurement. We normalize the results ($score_i$) such that $\sum score_i = 1$, and select the best candidate for recognition.

$$i = arg \max_i (score_i) \tag{1}$$

In addition, to evaluate the retrieval performance, we use the Bullseye test (see Sec. 4.2). The algorithm is summarized in Algorithm 2.

Given a test string of length L_{test}, the number of Tri-Grams to be tested in the shape retrieval is at most $2 \times \lfloor \frac{L_{test}}{2} \rfloor \times L_{test}$, and each vote cast takes constant time. Therefore the time complexity is $O(L_{test}^2 N)$ on a dataset containing N shapes. For a fixed length test sequence, the complexity is $O(N)$.

Algorithm 2 Shape Retrieval

Input	:	
S_{test}	:	Test string and
	:	its relative orientation string
$\mathcal{SM}$	:	Shape Memory
Output	:	
$score_i$	:	The score vector
		of all objects in memory

Procedure :

1) Compute $TG^j_{S_{test}}$ for possible j;
2) Initialize array p to zero;
3) FOR $g \in TG^j_{S_{test}}$
   IF object i has g at distance j in $\mathcal{SM}$
   FOR object k in the same category as i
   $p_k = p_k + e^{-(\frac{i-k}{2})^2}$;
5) $score_i = p_i / \sum_i p_i, i = 1, ..., N$;

4. Experiments

Four experiments have been performed to show the effectiveness and the efficiency of our representation. First, the Bullseye test is used for evaluating the performance of Shape Memory in retrieving shapes. The second experiment tests the Shape Memory on the UCF [13] dataset. In the third experiment, we show that Shape Memory can effectively used for real objects retrieval. Using a mobile platform equipped with a quad stereo system (Fig. 7), Shape Memory succeeds in recognizing 7 out of 8 objects from different viewing directions. Finally, we demonstrate that our representation also has the ability to reconstruct individual shapes in the training set from partial contour segments.

The MPEG 7 dataset [11] is used to train the Shape Memory in all the experiments. Three additional datasets are used for testing.

In all the experiments, we produce sentences by 1) uniformly sampling 50 points from each shape boundary in clockwise direction ($L_S = 50$), and 2) quantizing the line segments to eight directions ($|CB| = 8$).

4.1. Training the Shape Memory using the MPEG 7 Shape Dataset

The MPEG 7 shape dataset is widely used for shape matching algorithm. It contains 70 categories, each consisting of 20 binary images. Fig. 4a shows some examples, one for each category.

The dataset is one of the largest public shape datasets, and contains many real world shapes. This allows us to perform the retrieval using real world data.

Figure 4. Examples of the MPEG 7 Shape dataset.

4.2. Evaluating Shape Retrieval Performance using the Bullseye Test

The performance of shape retrieval is measured by the Bullseye test as follows. For each shape, we perform a leave-one-out test using Algorithm 2 and we count the top 40 candidates. The retrieval rate is the ratio of correct hits of the top 40 to the maximum possible number of hits (20×1400).

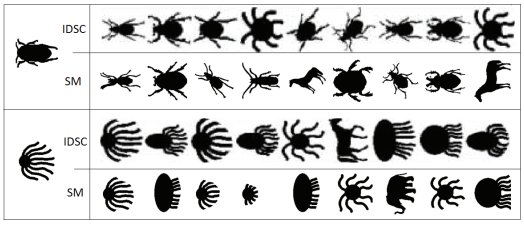


Figure 5. Two retrieval examples of MPEG 7 dataset, compared to the IDSC [12].

To handle mirror shapes, we compute for each object the sequences in both the original and the reverse order (*i.e.,* counter-clockwise). For this experiment, we use $\mathcal{J} = \lceil \frac{L_S}{10} \rceil$. Fig. 5 shows two retrieval results, one for the IDSC [12] and the other for our proposed method. Our false positive shapes are different from [12], and our order is different as well.

Table 2. Performance (Bullseye) and runtime comparison on the whole MPEG 7 CE-Shape-1 dataset. Every image in the dataset is used to perform the leave-one-out test using Algorithm 2.

Algorithm	Retrieval Rate	Appr. Total Runtime
CSS [14]	75.44%	> hour
Visual Parts [11]	76.45%	> hour
Curve Edit [18]	78.17%	> hour
Gen. Model [22]	80.03%	> hour
SC+D_{SC} [12]	64.59%	
SC+TPS [2]	76.51%	> hour
IDSC+D_{SC} [12]	68.83%	
IDSC+DP [12]	85.40%	> hour
Shape Indexing [3]	81.8%	∼10 minutes
Hierarchical [6]	87.7%	> hour
Proposed	**80.01%**	**∼2.5 minutes**

We further compare the runtime and the performance of other existing algorithms (Table 2). The Bullseye test on every shape in the MPEG 7 shape dataset is run on a standard PC with 1GB memory and a Pentium Core 2 Duo 2.8 GHz CPU. Referring to Table 2, the total runtime of our approach is significantly less than all others. Most previous algorithms are based on pairwise matching, resulting in the runtimes on the whole dataset in the order of hours[4]. We mark this as ">hour" in Table. 2. Compared to [3] which is not based on pairwise matching, our approach is faster, while the performance is marginally worse. Since we do not explicitly handle articulated objects, our performance is slightly worse than the performance of those techniques that handle this problem [12], [6] with the benefit of shorter runtime.

4.3. Shape Retrieval using UCF dataset

The purpose of this experiment is to test whether Shape Memory can correctly associate shapes in a different dataset to similar shapes in the training dataset.

The UCF dataset [13] is one of the largest gait datasets. Each gait image depicts the silhouette of a person. The closest shape of a human being in the MPEG 7 dataset is the silhouette of "Child".

We randomly choose 40 sequences in UCF dataset from persons that do not carry a briefcase. In total, 8038 gaits from the dataset are used in this experiment.

Fig. 6 shows the top 19 retrieved images for a test gait image. 16 out of 19 images are correctly retrieved as images of human.

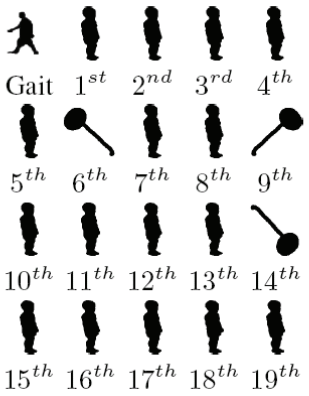


Figure 6. Retrieval using a gait image in the UCF dataset [13]. The gait image in the UCF dataset is shown on the top left. The remaining images are the retrieval results in the MPEG 7 dataset, sorted by rank.

[4]For example, in [12] each match takes 0.31s. Thus, the total runtime on the whole dataset would be approximately $0.31 \times \binom{1400}{2}$ seconds ($\sim$ 80 hrs).

Figure 7. The quad stereo system on a mobile platform used in Sec. 4.4.

In this experiment, a retrieval is correct if the top ranked candidate in the MPEG 7 dataset belongs to category "Child". We present the accuracy of 10 sequences in Table 3.

Table 3. The accuracy of persons in the UCF [13] dataset. A retrieval is correct if the top ranked candidate in the MPEG 7 dataset belongs to category "Child". We show the number of correct retrieval and the number of total images in the sequence.

Sequence#	Accuracy	Sequence#	Accuracy
03506G0ARB	184/196	03629G0ARB	141/151
03510G0ALB	192/211	03635G0ARB	240/256
03523G0ARB	178/193	03652G1ALB	196/208
03572G0ALB	168/187	03657G1ARB	188/200
03605G0ALB	203/229	03661G1BLB	179/184

4.4. Retrieving Real Objects using a Mobile Platform

We test the Shape Memory using a mobile platform equipped with a quad stereo system for depth and color segmentation [1] (Fig. 7). MPEG 7 dataset is used for training in this experiment.

8 objects are used in this experiment for testing: two bottles, two mugs, a car model, a Christmas bell, a jar, and a shoe (Fig. 8, column 1). We captured 20 images per object from different viewing directions. The quad stereo system segments the object from the background, as shown in Fig. 8, column 2. Then, the silhouettes are used for shape retrieval using the Shape Memory. The top 3 candidates are shown in the last 3 columns.

The speed for segmentation is approximately 1 second/frame for 640 by 480 images, and the time for shape retrieval is 0.05 second/frame on average. Table 4 further shows the retrieval accuracy for all the objects. The average accuracy is 82%. 7 objects except the "jar" can be correctly retrieved statistically. The poor performance of "jar" is caused by the imperfect segmentation.

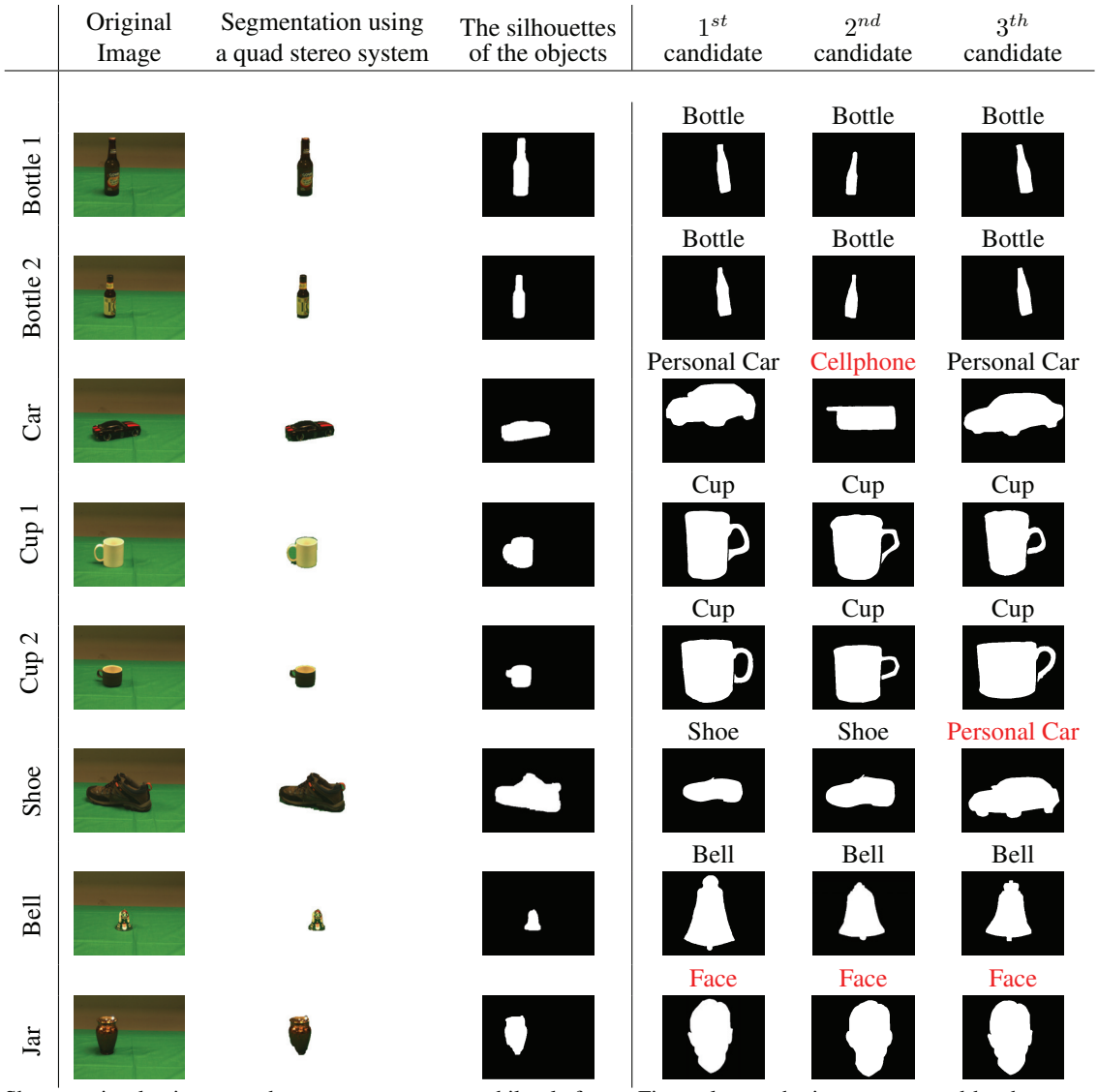


Figure 8. Shape retrieval using a quad stereo system on a mobile platform. First column: the images captured by the system; Second column: image segmentation using the quad stereo system; Third column: the silhouettes of the objects; Columns 4-6: the top 3 retrieval candidates using the Shape Memory. The class names of the candidates are shown on top, and the text are in red if the retrieval is incorrect (best viewed in color).

Table 4. The accuracy of shape retrieval with real world objects. A retrieval is correct if the top ranked candidate is in the same class as the test. The accuracy is defined as the number of correct retrievals divided the number of total ones. In this experiment we captured 20 images from different viewing directions.

Object	Accuracy	Object	Accuracy
Bottle 1	95%	Bottle 2	90%
Mug 1	95%	Mug 2	90%
Car	85%	Shoe	85%
Bell	95%	Jar	15%

4.5. Shape Reconstruction

The goal of this test is to demonstrate that the skip Tri-Grams representation encodes the distant shape information and can be used to reproduce the original structure. Given the Tri-Grams of shape S at distance from 0 to $\mathcal{J}$, we test the Shape Memory's ability to reproduce S for different values of $\mathcal{J}$ using Algorithm 1. This experiment is performed on all the shapes in the MPEG 7 dataset, and the average results are displayed in Fig. 9.

The accuracy is measured by:

- Hit Rate: Algorithm 1 may return different solutions, depending on how much the distant information is en-

coded. We divide the total number of the correct reconstructed strings by all the reconstructed results.

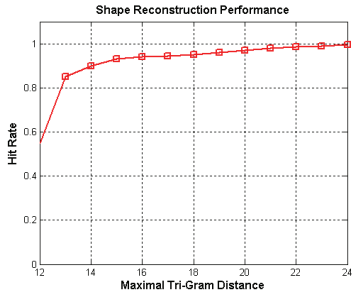


Figure 9. Performance on shape reconstruction.

Intuitively, larger values of $\mathcal{J}$ correspond to more global information. As shown in Fig. 9, it is likely to reconstruct the original shape when $\mathcal{J} > 12$ (*e.g.*, the Hit Rate is greater than 80%). It is noted that when $\mathcal{J} = 13$, the distance between the first and the last codeword in a Tri-Gram is 26, which is approximately half of the length of the shape string. The results indicate that the skip Tri-Gram implicitly encodes the distant geometry of the original shapes.

5. Conclusion

We presented an efficient, effective and scalable memory organization for fast retrieval and reconstruction of shapes. Shape Memory implicitly encodes the geometry of shape using the skip Tri-Gram model. The experiments using the mobile platform show that the approach is very efficient and its accuracy is reasonable to the best algorithms in the field.

References

[1] Anonymous. Color and depth segmentation using four cameras. *under review*.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[3] S. Biswas, G. Aggarwal, and R. Chellappa. Efficient indexing for articulation invariant shape matching and retrieval. In *CVPR*, 2007.

[4] M. K. Brown, A. Kellner, and D. Raggett. *Stochastic Language Models (N-Gram) Specification, W3C Working Draft 3*, 2001. `http://www.w3.org/TR/ngram-spec/`.

[5] D. T. Clemens and D. W. Jacobs. Space and time bounds on indexing 3d models from 2d images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(10):1007–1017, 1991.

[6] P. Felzenszwalb and J. Schwartz. Hierarchical matching of deformable shapes. In *CVPR*, 2007.

[7] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, Jan 2008.

[8] H. Freeman. Computer processing of line-drawing images. *ACM Comput. Surv.*, 6(1):57–97, 1974.

[9] K.-S. Fu. *Synthetic Methods in Pattern Recognition*. Academic Press, 1974.

[10] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Jan 1997.

[11] L. J. Latecki, R. Lakämper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages 424–429, 2000.

[12] H. Ling and D. W. Jacobs. Using the Inner-Distance for classification of articulated shapes. In *CVPR*, pages 719–726, 2005.

[13] Z. Liu and P. Grother. The humanid gait challenge problem: Data sets, performance, and analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(2):162–177, 2005.

[14] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space, 1996.

[15] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *ECCV*, pages 575–588, 2006.

[16] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR*, pages 3–10, 2006.

[17] I. Rigoutsos and R. Hummel. A bayesian approach to model matching with geometric hashing. *Comput. Vis. Image Underst.*, 62(1):11–26, 1995.

[18] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On aligning curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1):116–125, 2003.

[19] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[20] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *ICCV*, pages 503–510, 2005.

[21] SRVC. `http://www.semantic-robot-vision-challenge.org/`.

[22] Z. Tu and A. L. Yuille. Shape matching and recognition - using generative models and informative features. In *ECCV*, pages 195–209, 2004.

[23] H. Wolfson and Y. Lamdan. Geometric hashing: A general and efficient model-based recognition scheme. In *ICCV*, pages 238–249, 1988.

[24] X. Yu, L. Yi, C. Fermuller, and D. Doermann. Object detection using shape codebook. In *BMVC*, 2007.

[25] H. Zhang and J. Malik. Learning a discriminative classifier using shape context distances. In *CVPR*, pages 242–247, 2003.