Evaluating Contour Segment Descriptors

Cong Yang[1], Oliver Tiebe[1], Kimiaki Shirahama[1], Ewa Łukasik[2]
and Marcin Grzegorzek[1]

Abstract

Contour Segment (CS) is the fundamental element of partial boundaries or edges in shapes and images. So far, CS has been widely used in many applications, including object detection and matching, sketch-based object retrieval, open curve matching, shape matching, etc. To increase the matching accuracy and efficiency, a variety of CS descriptors have been proposed. A CS descriptor is formed by a chain of boundary or edge points and is able to encode the geometric configuration of a CS globally and partially. Because many different CS descriptors exist, a structured overview and quantitative evaluation are required in the context of CS matching and different applications. Therefore, the principal object of this paper is to assess the invariance properties, distinctiveness and computation complexity of 27 CS descriptors in a structured way. Firstly, the analytical invariance properties of CS descriptors are explored with respect to scaling, rotation and transformation. In addition, the distinctiveness of CS descriptors is assessed experimentally on three datasets. Lastly, the computation complexity of CS descriptors is studied by both theoretical analysis and runtime examination. From the theoretical and experimental results, we find that CS length and matching algorithm affect the performance of CS matching while matching algorithms have more affection. The results further reveal that, with different combinations of CS descriptors and matching algorithms, several requirements in terms of matching speed and accuracy can be fulfilled. Furthermore, a proper combination of CS descriptors

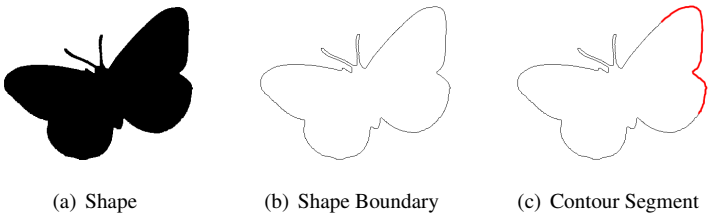
[1]Cong Yang, Oliver Tiebe, Kimiaki Shirahama and Marcin Grzegorzek are with the Research Group for Pattern Recognition, Institute for Vision and Graphics, University of Siegen, Siegen, D-57076, Germany. E-mail: cong.yang@uni-siegen.de.
[2]Ewa Łukasik is with the Laboratory of Operational Research and Artificial Intelligence, Institute of Computing Science, Poznan University of Technology, Pl. Marii Skłodowskiej-Curie 60-965 Poznan, Poland.

can improve the matching accuracy over the individual descriptors.

Keywords: Contour Segment, Contour Segment Descriptor, Open Curve Matching
2010 MSC: 00-01, 99-00

1. Introduction

A Contour Segment (CS) is a fragment of shape boundary which is constructed by a chain of connected boundary points. As shown in Figure 1, compared to the shape boundary which is defined by a circular sequence of boundary points, a CS describes the partial information of a shape boundary. For each boundary point in CS, we call it a CS point. The main motivation for CS is that the connectedness of shape boundary points is not ensured in practice due to the noise affections [1] and manual operations [2]. Thus, viewing CSs as local patches with any length instead of full boundary points provides more flexibility against boundary instabilities. As illustrated in Figure 2, psychophysical studies [3, 4, 5] show that we can recognise objects using CSs alone. Therefore, CS is an important element for computer vision tasks [6, 5].



(a) Shape (b) Shape Boundary (c) Contour Segment

Figure 1: Shape boundary and a contour segment (marked with the red line).

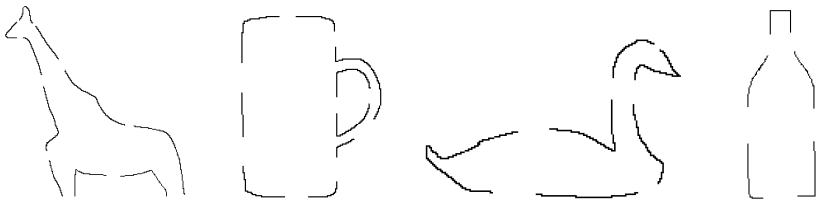
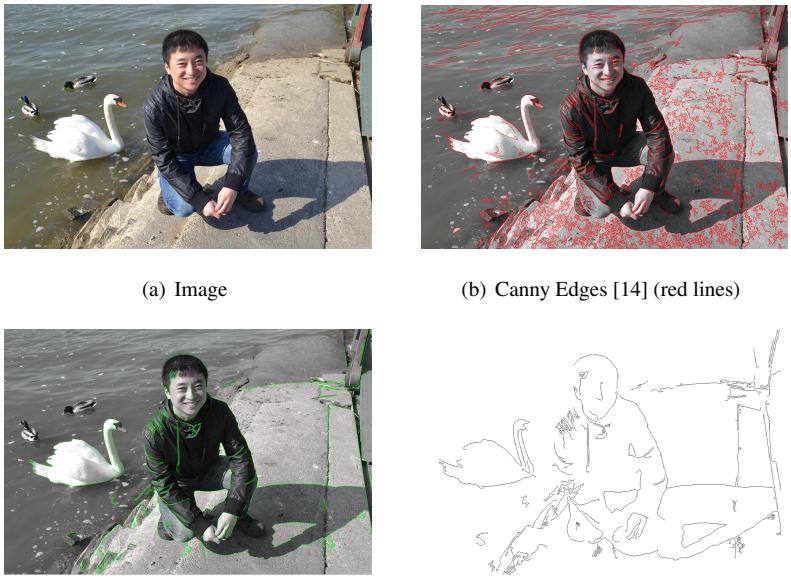


Figure 2: Human beings are able to interpret spatially arranged CSs to recognise the object classes.

Benefit from these properties, CS plays a key role in many different applications including object detection [7, 8] and matching [9], etc. This is because an object shape cannot be ideally segmented from an image due to the background clutter [1] and object overlapping [10]. To overcome these, one typical method is to represent the object boundaries and background using CSs [11, 12, 13]. Specifically, given an image (Figure 3(a)), object boundaries and background edges are firstly generated using intensity gradients [14] and zero-crossings [15] approaches (Figure 3(b)). After that, an edge reduction process is applied using edge dropping [16, 17] and linking [18] (Figure 3(c)) methods. With the above mentioned processes, an image is represented by various of CSs with different lengths and deformations. Depending on CSs, the object detection and matching tasks are addressed by solving the contour vs non-contour classification problem [19, 20].



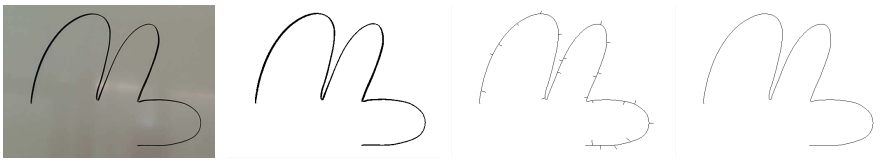
(a) Image (b) Canny Edges [14] (red lines)

(c) Edge Reduction [18] (blue lines) (d) CSs

Figure 3: CS generation from a colour image.

Moreover, CS is also commonly used for the application of sketch-based object retrieval [21, 2] since the sketches are normally not occlusive and only partially matched to object boundaries. Figure 4 illustrates a regular approach for representing sketching

information. Firstly, a sketching shape is segmented using the colour difference between the sketching and the background (Figure 4(b)). In order to remove the boundary noise and reduce the complexity of sketch matching, skeletonisation methods [22, 23] are employed to the generated medial axis of the sketching shape (Figure 4(c)) since it can largely preserves the extent and connectivity of the original shape while throwing away most of the original shape pixels. After that, by removing small branches in the medial axis using skeleton pruning methods [24, 25], a CS is generated that represents the geometrical and topological features of the original sketching (Figure 4(d)). Finally, the sketch-based object retrieval task is applied by matching the CSs from sketches and images.



(a) Hand Sketching (b) Segmented Shape (c) Shape Skeleton (d) CS

Figure 4: CS generation from a hand sketching.

CS is not only used in the specific applications we introduced above, but also regarded as a fundamental element of general applications like open curve matching [26, 9, 27]. Similar to CS, open curve is also a fragment of a shape boundary but with larger deformation and length than CSs. Open curve matching means finding similar parts between two open curves and then calculate their similarity. Essentially, open curve matching is employed by both object detection and recognition applications since the lengths and deformations of generated edges in an image are not uniformed (Figure 3(c) and 3(d)). Thus, we use CS to represent the partial features of an open curve, then search the similar parts by CS matching. Figure 5 illustrates four examples of open curve and that we use CS to find the similar parts among them. This strategy can also be used for shape matching by searching the similar parts among shape boundaries [28]. With these motivations, in this paper, one of our experiments is applied by the application of open curve matching.

In order to efficiently match CSs, proper CS descriptors and matching algorithms

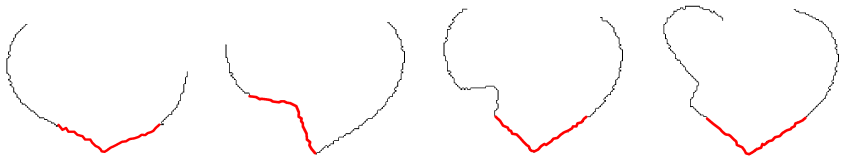


Figure 5: Searching similar parts among four open curves using CS (marked with the red colour) matching. Some open curves may have multiple similar parts, but here we only mark one for better visualisation.

are required. Since a CS is formed by a chain of CS points, a descriptor is required to encode the geometric configuration of those CS points globally and partially. More specifically, among all CS points, we normally select some sample points and generate CS descriptors by considering the geometrical relationship between those points. If sample points are selected roughly, CS descriptors represent the coarse-grained CS features. If sample points are selected densely, CS descriptors describe the fine-grained CS features. For CS matching, traditional matching algorithms like Hungarian [29], Dynamic Programming (DP) [30] or Dynamic Time Warping (DTW) [31] are normally used for searching the correspondences between sample points based on CS descriptors.

Although many CS descriptors have been proposed, it is unclear how the discrimination power of each of these descriptors is, and how similar or different it is to the other descriptors. Moreover, we do not know which matching method is more suitable for different CS descriptors. Lastly, the efficiency of each descriptor is also unclear. Therefore, this paper studies the invariance properties, matching performance, computation complexity and runtime of 27 CS descriptors and their matching algorithms in a structured way. Figure 6 illustrates the pipeline of the evaluating steps and their correlated sections (marked with the red colour). In order to conveniently describe the structure of the evaluated CS descriptors, we firstly define and formulate the CS. After that, 27 CS descriptors are introduced within three groups depending on their character. For each CS descriptor, we assign its matching algorithms by considering the feature structure. Specifically, for some CS descriptors, due to the structure of feature vectors, traditional matching algorithms like Hungarian, DP and DTW cannot be

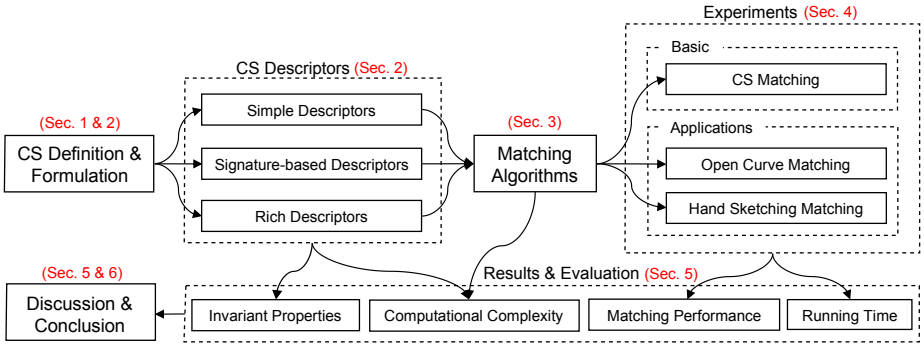


Figure 6: Pipeline for evaluating CS descriptors. Correlated sections are marked with the red texts.

applied properly. In this case, we only evaluate and compare their performance using different distance functions like correlation [32], histogram intersection (HI) [33], χ^2-statistics [34] and Hellinger [35].

Based on the experiment results, as illustrated in Figure 6, our evaluation is applied by taking four properties into account. Firstly, we analyse and compare a taxonomy of invariant properties. The taxonomy is derived by considering the requirements of shape descriptors with respect to the robustness and efficiency [36]. Using these requirements, a systematic analysis approach is adopted to provide a set of invariance properties of CS descriptors such as scale invariance, rotation invariance, etc. Secondly, we theoretically analyse the computational complexity of both feature generation and CS matching on 27 CS descriptors. Thirdly, the matching performance of CS descriptors is analysed experimentally using different combinations of CS descriptors and matching algorithms via three experiments including two applications. Lastly, the runtime in the CS matching experiment is collected and compared. With discussions and observations on the four properties above, we drew the recommendation for different application scenarios by balancing the matching accuracy and speed of CS descriptors.

As discussed above, CS has been used in several applications [7, 9, 2, 26] and many CS descriptors [2, 37, 38, 39] have been proposed. However, most authors only introduce their CS descriptors without any comparison. Though there are some papers

that survey shape representation techniques [36, 40] or compare a handful of CS descriptors in their specific applications [8], there is no paper that integrally surveys and evaluates existing CS descriptors. Thus, the first contribution of this paper is that we survey and evaluate the existing CS descriptors in a structured way. Secondly, in order to evaluate CS descriptors in different scenarios, we design and introduce 3 datasets for CS matching, open curve matching and hand sketching matching. Thirdly, formed on evaluations, we recommend the combinations of CS descriptors and matching algorithms for meeting different requirements in terms of accuracy and speed. Overall, when choosing a CS descriptor for open curve matching with time complexity not being of primary importance, the best choice is Point Triangle descriptor with DP [30] matching algorithm. To obtain a stable and promising performance while using less time for feature generation and matching, Partial Contour and Chord Distribution with DP are the best choice. If we want to apply a fast open curve matching and obtain relatively promising results, Cendistance with Hellinger vector distance method is the best choice.

This paper is organised as follows: In the next section, the mathematical definition of CS is presented. Moreover, 27 CS descriptors are introduced by their types. Furthermore, we introduce the distance functions, open curve matching algorithms in Section 3. Three experiments are designed and presented in Section 4. In Section 5, the experiment results are given and discussed. Finally, conclusions are drawn in Section 6.

2. Contour Segment Descriptors

In this section, we first formulate the definition of CS. After that, three groups of CS descriptors are introduced and formulated: Simple CS descriptors, Signature-based CS descriptors and Rich CS descriptors (Table 1). Here, CS descriptors are classified and grouped based on the structure of their feature vectors. The evaluated descriptors are mainly selected from two sources: (1) Directly designed for CS representation, including most rich CS descriptors. (2) Originally designed for shape representation, but can be used or modified into CS description, including simple, signature-based and

some rich CS descriptors.

Name	Notation	Name	Notation	Name	Notation
Area	$\boldsymbol{f}_1$	Comcoor	$\boldsymbol{f}_{10}$	Point Triangle	$\boldsymbol{f}_{19}$
Circularity	$\boldsymbol{f}_2$	Cendistance	$\boldsymbol{f}_{11}$	Contour Context	$\boldsymbol{f}_{20}$
Eccentricity	$\boldsymbol{f}_3$	Tangent	$\boldsymbol{f}_{12}$	Beam Angle	$\boldsymbol{f}_{21}$
Bending	$\boldsymbol{f}_4$	Curvature	$\boldsymbol{f}_{13}$	Partial Contour	$\boldsymbol{f}_{22}$
Rectangularity	$\boldsymbol{f}_5$	Area Function	$\boldsymbol{f}_{14}$	Opt Partial Contour	$\boldsymbol{f}_{23}$
LineRatio	$\boldsymbol{f}_6$	Triangle Area	$\boldsymbol{f}_{15}$	Chord Distribution	$\boldsymbol{f}_{24}$
Convexity	$\boldsymbol{f}_7$	Chord Length	$\boldsymbol{f}_{16}$	Length Direction	$\boldsymbol{f}_{25}$
Solidity	$\boldsymbol{f}_8$	Turning Angle	$\boldsymbol{f}_{17}$	Line Segment	$\boldsymbol{f}_{26}$
Dislength	$\boldsymbol{f}_9$	Height Function	$\boldsymbol{f}_{18}$	Sub Box	$\boldsymbol{f}_{27}$
$\boldsymbol{f}_1$ - $\boldsymbol{f}_9$: Simple descriptors, $\boldsymbol{f}_{10}$ - $\boldsymbol{f}_{17}$: Signature-based descriptors, $\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{27}$: Rich descriptors					

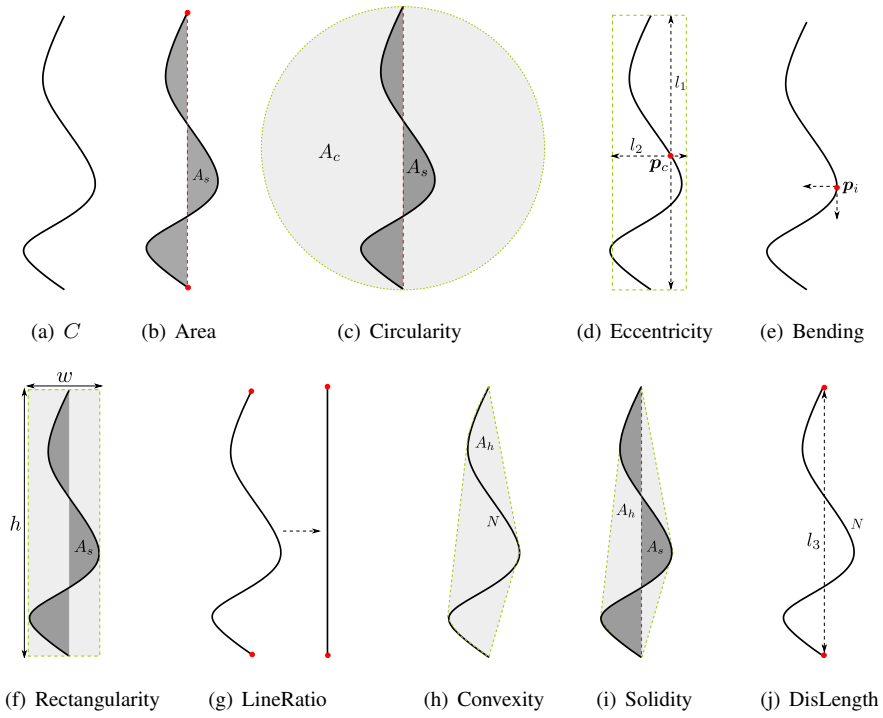
Table 1: Three types of CS descriptors and their notations.

No matter in which group, CSs share some common characteristics like one pixel width and two endpoints, etc. In order to regulate and standardise our next descriptions, for a CS, we assume the following restrictions: (1) With one pixel width. (2) With two endpoints, which only have one neighbour point. (3) No intersection point: except two endpoints, the rest points only have two neighbour points. Formed on these restrictions, we denote a CS with C. Let $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_N$ be the CS points along with the boundary path. For a single CS point $\boldsymbol{p}_i$ where $i = 1, 2, \cdots, N$, it can be represented by its Cartesian coordinate $[x_i, y_i]$ in the image. We set CSs to have the same number of points. This is because on the one hand, it is easier for us to fairly evaluate the performance of CS descriptors using the same matching methods. On the other hand, as discussed in Section 1, open curve-based matching can be decomposed into multiple partial curve matching tasks using CSs with the same number of points.

2.1. Simple CS Descriptors

There are nine simple CS descriptors evaluated in this paper. A simple CS descriptor is a vector that represents the global features of a CS. CS descriptors in this group are normally generated by considering the global CS geometry and their feature vectors are one-dimensional. The motivation of simple CS descriptors is built on the claim that one often applies too sophisticated methods to practical problems [41]. In contrast,

it is desired to find a method that is both simple and generally applicable. Moreover, each descriptor should be semantically simple. Thirdly, combing descriptors should introduce a new perspective. Lastly, some correlation between descriptors is acceptable.
145 Keeping these motivations in mind, we survey and revise nine simple CS descriptors from shape survey papers [40, 36] and applications [41, 42]. In general, these simple descriptors usually can only discriminate CS with large differences. Therefore, they are not used as standalone descriptors but usually used as filters to eliminate false hits or combined with other rich descriptors to discriminate CSs.



(a) C (b) Area (c) Circularity (d) Eccentricity (e) Bending

(f) Rectangularity (g) LineRatio (h) Convexity (i) Solidity (j) DisLength

Figure 7: Simple descriptors for a CS C in (a).

Area: As shown in Figure 7(b), the area descriptor [40, 41] $\boldsymbol{f}_1$ is calculated as the area A_s (dark grey area) between the straight line (red dotted line) connecting the CS endpoints (red points) and the CS itself (Figure 7(a)). In order to ensure the scale

invariance, $\boldsymbol{f}_1$ is normalised by the length [2] of CS N:

$$\boldsymbol{f}_1 = \frac{A_s}{N} \quad . \tag{1}$$

Circularity: Circularity [40, 41] $\boldsymbol{f}_2$ illustrates how similar the CS C is to a circle. As shown in Figure 7(c), circularity is calculated by:

$$\boldsymbol{f}_2 = \frac{A_s}{A_c} \quad . \tag{2}$$

where A_c denotes the area of the minimum CS surrounding circle (light grey area).

Eccentricity: Eccentricity [42, 41] $\boldsymbol{f}_3$ can be uniquely defined as the ratio of length of major axis to minor axis that cross each other orthogonally in the middle of the CS. We first find the middle point $\boldsymbol{p}_c$ of CS C (the red point in Figure 7(d)), then eccentricity is calculated by:

$$\boldsymbol{f}_3 = \frac{l_1}{l_2} \quad . \tag{3}$$

where l_1 and l_2 are the lengths of major axis and minor axis to the CS minimum bounding rectangle on $\boldsymbol{p}_c$, respectively.

Bending: Bending [36, 42] $\boldsymbol{f}_4$ is defined by the average bending energy. It captures the degree of a CS bending energy. For instance, the circle is the shape with the minimum bending energy. $\boldsymbol{f}_4$ is calculated by:

$$\boldsymbol{f}_4 = \frac{1}{N} \sum_{i=1}^{N} K(i)^2 \quad . \tag{4}$$

where $K(i)$ denotes the curvature of point $\boldsymbol{p}_i$ in CS (Figure 7(e)).

Rectangularity: Rectangularity [41] $\boldsymbol{f}_5$ presents how rectangular a CS is , i.e. how much the CS fills its minimum bounding rectangle (Figure 7(f)):

$$\boldsymbol{f}_5 = \frac{A_s}{w \cdot h} \quad . \tag{5}$$

where w and h are the width and height of the CS minimum bounding rectangle, A_s is the CS area introduced in $\boldsymbol{f}_1$.

[2] Based on the CS definition, $p_1, p_2, \cdots, p_N$ are the CS points along with the CS path, therefore, N is the length of a CS.

LineRatio: Sometimes a CS should be compared against a template. Clearly, a straight line must be one of the simplest and most general choice. LineRatio [42, 41] $\boldsymbol{f}_6$ illustrates how similar a CS is to a straight line (Figure 7(g)). $\boldsymbol{f}_6$ is calculated by:

$$\boldsymbol{f}_6 = \frac{h}{N} \quad . \tag{6}$$

where h is the height of the CS minimum bounding rectangle and N is the CS length.

Convexity: Convexity [41] $\boldsymbol{f}_7$ is defined as the ratio of the convex hull [43] over that of the CS length. Convexity captures the minimal convex covering of a CS. It can be considered as an elastic ribbon stretched around the CS. A straightforward measure for Convexity can be defined as:

$$\boldsymbol{f}_7 = \frac{A_h}{N} \quad . \tag{7}$$

where A_h denotes the CS convex hull area (Figure 7(h)) and N is the length of CS.

Solidity: As shown in Figure 7(i), solidity [41] $\boldsymbol{f}_8$ describes the extent to which the CS is convex or concave and is defined by:

$$\boldsymbol{f}_8 = \frac{A_s}{A_h} \quad . \tag{8}$$

where A_s is the CS area introduced in $\boldsymbol{f}_1$ and A_h is the convex hull area introduced in $\boldsymbol{f}_7$. Solidity is an indicator that captures the concave-convex condition of a CS. For instance, the solidity of a convex CS is always 1.

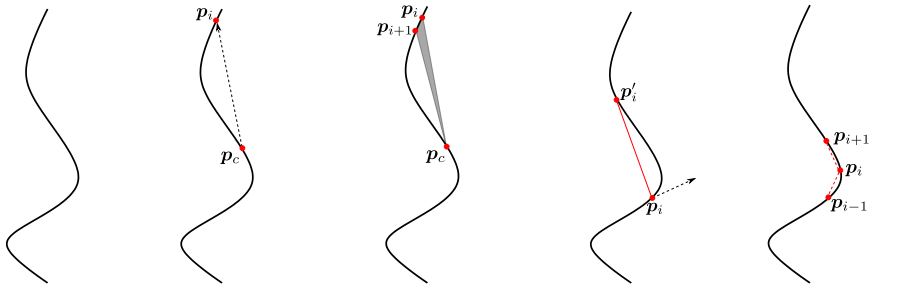
Dislength: Dislength [9] $\boldsymbol{f}_9$ illustrates the skewness power of CS. As shown in Figure 7(j), it can be defined by the ratio between distance of endpoints l_3 and the CS length N:

$$\boldsymbol{f}_9 = \frac{l_3}{N} \quad . \tag{9}$$

2.2. Signature-based CS Descriptors

In this part, we introduce eight signature-based CS descriptors. CS signature is the modified version of Shape Signature [36]. A shape signature represents a shape by a one dimensional function derived from shape boundary points. For a CS, we analogically reduce the CS point coordinates into one dimension. Signature-based descriptors can capture the perceptual features of CSs and are often combined with some other feature extraction algorithms like Fourier descriptors [44, 45, 46, 47, 48] and Wavelet

Descriptors [49, 50, 51]. The motivation of signature-based CS descriptors is that we want to represent a CS with a midterm order of feature vector than the simple and rich CS descriptors. Thus, signature-based CS descriptors could offer a proper way for balancing matching accuracy and speed.



(a) C (b) Cendistance (c) AreaFunction (d) ChordLength (e) Turning Angle

Figure 8: Some signature-based descriptors for a CS C in (a).

Comcoor: Comcoor [40] descriptor $\boldsymbol{f}_{10}$ is an abbreviation of Complex Coordinates. Complex Coordinates is mainly designed for transforming a CS from a two-dimensional representation to a one-dimensional one. Let $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_N$ be the points of CS C, $\boldsymbol{p}_c$ be the middle point. Based on the definition in this section, CS points can be represented by its Cartesian coordinate. Therefore, the Complex Coordinates function is:

$$f_{10}^i = [x_i - x_c] + [y_i - y_c] \quad . \tag{10}$$

where $[x_c, y_c]$ is the coordinate of $\boldsymbol{p}_c$ and $[x_i, y_i]$ is the coordinate of any point in C, $i = 1, 2, \cdots, N$. Finally, $\boldsymbol{f}_{10} = [f_{10}^1, f_{10}^2, \cdots, f_{10}^N]$.

Cendistance: Cendistance [36] descriptor $\boldsymbol{f}_{11}$ is an abbreviation of Centroid Distance Function which can be used for the representation of centroid-based time series. The main characteristics of this descriptor is (1) simplicity and (2) the extracted series can be standardised and re-sampled to the same format since it uses angles. Similar to Comcoor, as shown in Figure 8(b), for a CS C, we first find its middle point $\boldsymbol{p}_c$ which can be represented by $[x_c, y_c]$. After that, the Centroid Distance Function is defined as

the angle between $\boldsymbol{p}_c$ and the CS point $\boldsymbol{p}_i$:

$$f_{11}^i = |\langle x_c - x_i, y_c - y_i \rangle| \quad . \tag{11}$$

where $[x_i, y_i]$ is the coordinate of a CS point $\boldsymbol{p}_i$, $i = 1, 2, \cdots, N$. Finally, $\boldsymbol{f}_{11} = [f_{11}^1, f_{11}^2, \cdots, f_{11}^N]$.

Tangent: Tangent [36] descriptor $\boldsymbol{f}_{12}$ is an abbreviation of Tangent Angle Function. The Tangent Angle Function at point $\boldsymbol{p}_i$ can be defined as

$$f_{12}^i = \tan^{-1}\left(\frac{y_i - y_{i-w}}{x_i - x_{i-w}}\right) \quad . \tag{12}$$

where w is a window for calculating f_{12}^i more accurately. In our experiment, we set $w = 5$ and $(i - w) = 1$ if $(i - w) <= 0$. Finally, $\boldsymbol{f}_{12} = [f_{12}^1, f_{12}^2, \cdots, f_{12}^N]$. Since Tangent CS descriptor requires window for feature generation, it is sensitive to noise. However, compared to most signature-based CS descriptors, this one is more flexible since we can control its accuracy by changing the size of window.

Curvature: Curvature [36] descriptor $\boldsymbol{f}_{13}$ is very important for capturing salient perceptual characteristics and invariant under rotations and translations. Let $K(i)$ be the curvature of a CS point $\boldsymbol{p}_i$. $K(i)$ is calculated by three successive points $\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$. Specifically, the curvature of a circle drawn through them is simply four times the area of the triangle formed by these three points divided by the product of its three sides. Using the coordinates of the points this is given by:

$$K(i) = \frac{2|(x_i - x_{i-1})(y_{i+1} - y_{i-1}) - (x_{i+1} - x_{i-1})(y_i - y_{i-1})|}{\sqrt{((x_i - x_{i-1})^2 + (y_i - y_{i-1})^2)((x_{i+1} - x_{i-1})^2 + (y_{i+1} - y_{i-1})^2)((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2)}} \tag{13}$$

where $[x_{i-1}, y_{i-1}]$, $[x_i, y_i]$ and $[x_{i+1}, y_{i+1}]$ are the Cartesian coordinate of CS point $\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$, respectively. $K(i)$ can be considered as an approximation to the CS's curvature at the CS point $\boldsymbol{p}_i$. In order to ensure the scale invariance, $K(i)$ is normalised by the mean absolute curvature:

$$K'(i) = \frac{K(i)}{\frac{1}{N}\sum_{i=1}^{N} |K(i)|} \quad . \tag{14}$$

Finally, $\boldsymbol{f}_{13} = [K'(1), K'(2), \cdots, K'(N)]$.

Area Function: Area Function [36] descriptor $\boldsymbol{f}_{14}$ is calculated by the triangle area between the middle and another two CS points (Figure 8(c)). More specifically, the

area of triangle f_{14}^i is formed by the triplet: $(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}, \boldsymbol{p}_c)$, where $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$ and $\boldsymbol{p}_c$ are three points on CS, $\boldsymbol{p}_c$ is the middle point. The Area Function descriptor $\boldsymbol{f}_{14}$ for C can be represented by the whole CS points as a set of feature vectors: $\boldsymbol{f}_{14} = [f_{14}^1, f_{14}^2, \cdots, f_{14}^N]$. Area Function is simple and can be used to collect fine-grained features since the distance between $\boldsymbol{p}_i, \boldsymbol{p}_{i+1}$ is small and it can capture the small deformations of a CS. Moreover, the coarse-grained features are also preserved as we collect the whole areas from $(\boldsymbol{p}_i, \boldsymbol{p}_{i+1})$ to a fix point $\boldsymbol{p}_c$.

Triangle Area: Different from Area Function, Triangle Area [52] descriptor $\boldsymbol{f}_{15}$ is computed directly from area of triangles formed by CS points. Triangle area provides useful information about CS features such as the convexity/concavity at each CS point. Therefore, this descriptor provides high discrimination capability. Specifically, let N be the number of CS points, $i \in [1, N]$ and $t_s \in [1, \frac{N}{2} - 1]$. For each three points $\boldsymbol{p}_{i-t_s}$, $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+t_s}$, their Cartesian coordinates are $[x_{i-t_s}, y_{i-t_s}]$, $[x_i, y_i]$ and $[x_{i+t_s}, y_{i+t_s}]$. The triangle area is formed by:

$$f_{15}^i = \frac{1}{2} \det \begin{pmatrix} x_{i-t_s} & y_{i-t_s} & 1 \\ x_i & y_i & 1 \\ x_{i+t_s} & y_{i+t_s} & 1 \end{pmatrix} \quad . \tag{15}$$

With this equation, when the CS path is traversed in clock-wise direction, positive, negative and zero values of triangle area mean convex, concave and straight-line points, respectively. Finally, $\boldsymbol{f}_{15} = [f_{15}^1, f_{15}^2, \cdots, f_{15}^N]$. Comparing to Area Function, Triangle Area is more flexible since the gap between three points can be modified by changing t_s. Thus, it can generate multi-scale (i.e. of different triangle side lengths) signatures at all boundary points.

Chord Length: Chord Length [53] descriptor $\boldsymbol{f}_{16}$ is derived by the distance between the CS point and its reference point. As shown in Figure 8(d), for each CS point $\boldsymbol{p}_i$, its chord length f_{16}^i is the shortest distance between $\boldsymbol{p}_i$ and another CS point $\boldsymbol{p}_i'$ so that $\boldsymbol{p}_i\boldsymbol{p}_i'$ is perpendicular to its tangent vector at $\boldsymbol{p}_i$, $i = 1, 2, \cdots, N$. For scale invariance, f_{16}^i is normalised by the CS length N. Finally, $\boldsymbol{f}_{16} = [f_{16}^1, f_{16}^2, \cdots, f_{16}^N]$. Chord Length descriptor is robust to fine-grained deformations since chord lengths are calculated using different reference points rather than only one middle point like

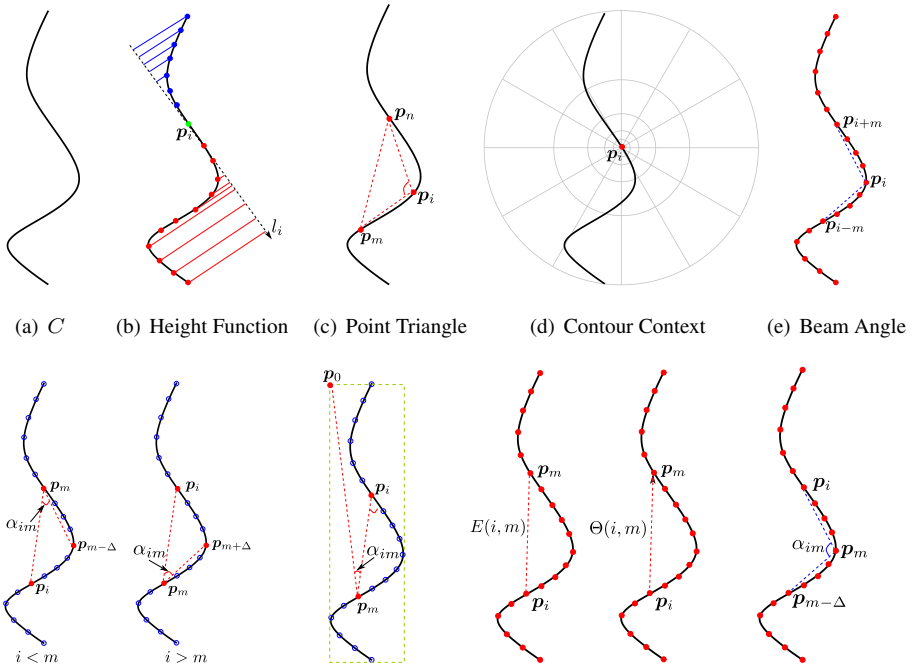
Area Function, etc. For example, if the middle point of a CS is changed because of deformation or noise, most chord lengths remain the same since each CS point may have different reference point for calculating its chord length.

Turning Angle: Compared to most signature-based CS descriptors, Turning Angle [54] descriptor $\boldsymbol{f}_{17}$ is more straightforward and efficient to compute. As shown in Figure 8(e), for a CS point $\boldsymbol{p}_i$, its neighbouring points $\boldsymbol{p}_{i-1}$ and $\boldsymbol{p}_{i+1}$, $\angle \boldsymbol{p}_{i-1}\boldsymbol{p}_i\boldsymbol{p}_{i+1}$ form a turning angle. For the points which have only one neighbouring point, we set their turning angle as 0. Therefore, for each CS point $\boldsymbol{p}_i$, it can be represented by its turning angle f_{15}^i. The turning angle descriptor $\boldsymbol{f}_{17}$ for C can be represented by the whole CS points as a set of feature vectors: $\boldsymbol{f}_{17} = [f_{17}^1, f_{17}^2, \cdots, f_{17}^N]$. Similar to Area Function, Turning angle captures the fine-grained features of a CS by using adjacent points. However, it has less ability for preserving coarse-grained deformation since the generated turning angles are globally isolated. On the contrary, area functions are not isolated since they are all connected by the middle point of a CS.

2.3. Rich CS Descriptors

CS rich descriptors capture the CS geometrical features in both fine- and coarse-grained levels. Compared to simple and signature-based CS descriptors, the feature vector of rich descriptors has more dimensions and varieties. Therefore, rich CS descriptors carry more information of the original CS. However, considering the computational complexity, rich CS descriptors require more runtime for feature generation as the features are normally generated by considering the relationship between every CS sample points. In this paper, we introduce and evaluate ten rich descriptors.

Height Function: Height Function [55] descriptor $\boldsymbol{f}_{18}$ was proposed for matching and recognizing 2D object silhouettes. As shown in Figure 9(b), the CS is represented by a fixed number of sample points. For each sample point, a height function is defined based on the distances of the other sample points to its tangent line. The motivation of this descriptor is to represent a CS point by considering its relations to all other sample points in the same direction. This way, the hight functions of the sample points represent the geometric changes and deformations of a CS. More specifically, for a sample point $\boldsymbol{p}_i$, its tangent line is l_i, we obtain a feature vector representing the height



(a) C (b) Height Function (c) Point Triangle (d) Contour Context (e) Beam Angle

(f) Partial Contour (g) Opt Partial Contour (h) Length Direction (i) Chord Distribution

Figure 9: Some rich descriptors for a CS C in (a). This group of CS descriptors will use the same matching and distance algorithms for evaluation.

values:

$$F_i = (f_i^1, f_i^2, \cdots, f_i^N)^T \quad . \tag{16}$$

where f_i denotes the distance from a sample point to the tangent line l_i. By using height function for every sample point, we define a matrix $\boldsymbol{f}_{18}$ for representing a CS C:

$$\boldsymbol{f}_{18} = (F_1, F_2, \cdots, F_N) \quad . \tag{17}$$

We observe that $\boldsymbol{f}_{18}$ is a $N \times N$ matrix with column i being the CS descriptor F_i of the sample point $\boldsymbol{p}_i$. In order to make the descriptor scale invariant, we perform row-wise normalisation on $\boldsymbol{f}_{18}$ by dividing by the maximal absolute value of each row.

Point Triangle: Point Triangle [56] descriptor $\boldsymbol{f}_{19}$ is a simple and intuitive CS descriptor. It is inspired by Carlsson [57] in which they only consider qualitative orientation

of each triangle: oriented clock or counterclockwise. Point triangle descriptor provides a full quantitative description of each triangle. This leads to a significant increase in descriptive power. As shown in Figure 9(c), given a CS point $\boldsymbol{p}_i \in C$, the point triangle descriptor of point $\boldsymbol{p}_i$ denoted by F_i is a histogram of all triangles spanned by $\boldsymbol{p}_i$ and all pairs of points $\boldsymbol{p}_m, \boldsymbol{p}_n \in C$, where point $\boldsymbol{p}_i$, $\boldsymbol{p}_m$, $\boldsymbol{p}_n$ must be different and $i, m, n = 1, 2, \cdots N$. Especially, F_i is a histogram with three bins including the angles $\angle \boldsymbol{p}_m \boldsymbol{p}_i \boldsymbol{p}_n$, and two distances $\boldsymbol{p}_i \boldsymbol{p}_m$ and $\boldsymbol{p}_i \boldsymbol{p}_n$. In order to distinguish the two distances, we require that the triangle $\boldsymbol{p}_n \boldsymbol{p}_i \boldsymbol{p}_m$ is oriented clockwise. In order to make the descriptor scale invariant, the distances are normalized by the average pairwise distance of points in C. Finally the point triangle descriptor $\boldsymbol{f}_{19}$ of C is a joint histogram of all points.

Contour Context: Contour Context [58] descriptor $\boldsymbol{f}_{20}$ was introduced based on the basic idea of Shape Context [59]. For each point $\boldsymbol{p}_i$ among N CS points, they consider the $N-1$ vectors obtained by connecting $\boldsymbol{p}_i$ to all other points. The key motivation is that the distribution over relative positions of each CS point is a robust, compact, and highly discriminative descriptor. More specifically, as has been defined in [59], for a point $\boldsymbol{p}_i$ on the CS, they compute a coarse histogram f_i of the relative coordinates of the remaining contour points,

$$f_{i,m} = \#\{\boldsymbol{q} \neq \boldsymbol{p}_i \quad : \quad (\boldsymbol{q} - \boldsymbol{p}_i) \in bin(m)\} \quad , \tag{18}$$

where $\#\{\cdot\}$ is the function that extracts feature values for log-polar histogram and $\boldsymbol{q}$ denotes the CS points except $\boldsymbol{p}_i$. They use five bins for $\log r$ and 12 bins for θ (Figure 9(d)). Thereafter, for each CS point $\boldsymbol{p}_i$, its contour context feature F_i can be represented as a 60-dimensional feature vector.

Beam Angle: The basic idea of Beam Angle [6] descriptor $\boldsymbol{f}_{21}$ is to represent each CS point by a weighted Beam Angle Histogram (BAH). Different from Turning Angle $\boldsymbol{f}_{17}$, this descriptor represents a CS point using multiple angles with different weights. With this, beam angle descriptor can mitigate the uncertainty in CS representation since it down-weights the interaction of distant CS points. More specifically, at CS point $\boldsymbol{p}_i, i = 1, 2, \cdots, N$, the beam angle is subtended by lines $(\boldsymbol{p}_{i+m}, \boldsymbol{p}_i)$ and $(\boldsymbol{p}_i, \boldsymbol{p}_{i-m})$, as illustrated in Figure 9(e). $\boldsymbol{p}_{i+m}$ and $\boldsymbol{p}_{i-m}$ are two neighbouring equi-distant points

by m points along CS from $\boldsymbol{p}_i$, $m = 1, 2, \cdots N'$, where N' is determined by experiment. BAH is a weight histogram, where the weight of angle $\angle \boldsymbol{p}_{i+m}\boldsymbol{p}_i\boldsymbol{p}_{i-m}$ is computed as $\exp(-\kappa m)$ where $k = 0.01$. BAH is invariant to translation, in-plane rotation, and scale. Experimentally, we find that BAH with 12 bins ($N' = 12$) gives optimal and stable results. Therefore, for a CS point $\boldsymbol{p}_i$, it can be represented by a 12-dimensional BAH F_i:

$$F_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,12}) \quad . \tag{19}$$

where f_i denotes one beam angle of $\boldsymbol{p}_i$ with weight. The BAH descriptor $\boldsymbol{f}_{21}$ can be represented by the whole CS points as a set of feature vectors.

Partial Contour: Partial Contour [60] descriptor $\boldsymbol{f}_{22}$ was proposed for object category localisation. The descriptor is calculated by the relative orientations between lines that connect the CS sampled points. More specifically, they calculate angles α_{im} between a line connecting the points $\boldsymbol{p}_i$ and $\boldsymbol{p}_m$ and a line to a third point relative to the position of the previous two points. This angle is defined

$$\alpha_{im} = \begin{cases} \angle(\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_m}, \overrightarrow{\boldsymbol{p}_m\boldsymbol{p}_{m-\Delta}}) & \text{if } i < m \\ \angle(\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_m}, \overrightarrow{\boldsymbol{p}_m\boldsymbol{p}_{m+\Delta}}) & \text{if } i > m \\ 0 & \text{otherwise} \end{cases} \quad . \tag{20}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_m$ are the i^{th} and m^{th} points in the sequence of sampled points of the CS and Δ is an offset parameter of the descriptor ($\Delta = 5$ for all experiments), $i, m = 1, 2, \cdots, N$. See Figure 9(f) for an illustration of the choice of points along the CS. The third point is chosen depending on the position of the other two points to ensure that the selected point is always inside the CS. This allows them to formulate the descriptor as a self-containing descriptor of any of its parts. For a CS point $\boldsymbol{p}_i$, the angles α_{im} are calculated between every pair of points along a contour. In such a way a CS defined by a sequence of N points is described by an $N \times N$ matrix $\boldsymbol{f}_{22}$ where an entry in row i and column m yields the angle α_{im}. The Partial Contour descriptor has several important properties. First, its angular description makes it translation and rotation invariant. Second, providing a local (close to matrix diagonal) and global (far from

matrix diagonal) description. Thirdly, the definition as a self-containing descriptor allows to implicitly retrieve partial matches.

Opt Partial Contour: Opt Partial Contour [8] descriptor $\boldsymbol{f}_{23}$ was proposed for object detection. Different from the Partial Contour ($\boldsymbol{f}_{22}$), $\boldsymbol{f}_{23}$ has a different sampling strategy to define the angles. Moreover, they analyse angles defined by lines connecting a CS-dependent reference point and the CS points. With these strategies, both fine- and coarse-grained features can be captured since they consider both relative angles between points on and out the CS. Thus, opt partial contour descriptor has higher discriminate power and is able to distinguish between different regular CSs as e. g. the semi-circles.

More specifically, for a CS C with N points, they define an $N \times N$ descriptor matrix $\boldsymbol{f}_{22}$ with the diagonal equals to zero. Every entry $\alpha_{im}(i \neq m)$ is defined by the angle between a line connecting the points $\boldsymbol{p}_i$ and $\boldsymbol{p}_m$ and a line from $\boldsymbol{p}_m$ to a reference point $\boldsymbol{p}_0$, which is defined by the upper left corner of the CS surrounding bounding box. In such a way, they define

$$\alpha_{im} = \angle(\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_m}, \overrightarrow{\boldsymbol{p}_m\boldsymbol{p}_0}) \quad . \tag{21}$$

where $i, m = 1, 2, \cdots, N$. For a single CS, the angles are calculated over all possible point combinations, yielding the descriptor matrix $\boldsymbol{f}_{23}$.

Chord Distribution: Chord Distributions [38] descriptor $\boldsymbol{f}_{24}$ was proposed for partial shape matching. A chord is a line joining two points of a region boundary, and the distribution of their lengths and angles was used as shape descriptor before, as e. g. by Cootes et al. [61]. This descriptor uses such chords to exploit the available point ordering information for the subsequent order-preserving assignment matching. In comparison, the contour context descriptor $\boldsymbol{f}_{20}$ loses all the ordering information due to the histogram binning and for that reason does not consider the influence of the local neighbourhood on single point matches.

This descriptor is based on angles α_{im} which describe the relative spatial arrangement of the sampled points. As shown in Figure 9(i), an angle α_{im} is calculated between a chord $\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_m}$ from a reference point $\boldsymbol{p}_i$ to another CS point $\boldsymbol{p}_m$ and a chord

$\overrightarrow{\boldsymbol{p}_m\boldsymbol{p}_{m-\Delta}}$ from $\boldsymbol{p}_m$ to $\boldsymbol{p}_{m-\Delta}$ by

$$\alpha_{im} = \angle(\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_m}, \overrightarrow{\boldsymbol{p}_m\boldsymbol{p}_{m-\Delta}}) \quad . \tag{22}$$

where $i, m = 1, 2, \cdots, N$ and $\boldsymbol{p}_{m-\Delta}$ is the point that comes Δ positions before $\boldsymbol{p}_m$ in the sequence as is illustrated in Figure 9(i). For a single CS, the angles are calculated over all possible point combinations, yielding the descriptor matrix $\boldsymbol{f}_{24}$.

Length Direction: Length Direction [62] descriptor $\boldsymbol{f}_{25}$ was proposed for object detection. Given a CS C with N points, for every point $\boldsymbol{p}_i \in C$, $i = 1, 2, \cdots, N$, they consider both the length and direction of the vector from $\boldsymbol{p}_i$ to other points in C. The length and direction features are saved by two matrices. This structure is more flexible for CS matching since it can be adopted to different matching algorithms.

More specifically, as shown in Figure 9(h), given a CS C with point sequence $C = \boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_N$, they compute two matrices, one representing all lengths and the second one representing all pairwise orientations of vectors from each $\boldsymbol{p}_i$ to each $\boldsymbol{p}_m \in C$, $m = 1, 2, \cdots, N$. The distance $E(i, m)$ from $\boldsymbol{p}_i$ to $\boldsymbol{p}_m$ is defined as Euclidean distance in the log space

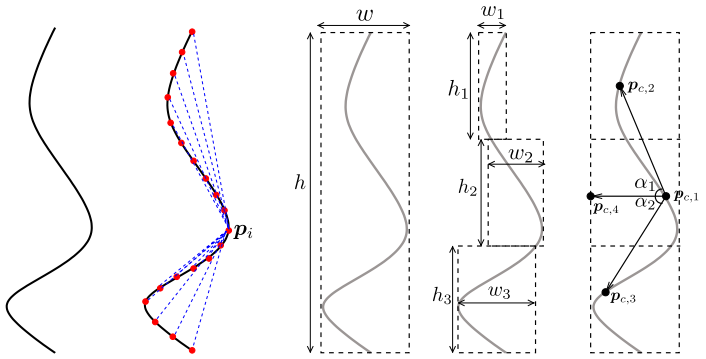
$$E(i, m) = \log(1 + \|\overrightarrow{\boldsymbol{p}_i} - \overrightarrow{\boldsymbol{p}_m}\|_2) \quad . \tag{23}$$

They add one to Euclidean distance to make the $E(i, m)$ positive. The orientation $\Theta(i, m)$ from $\boldsymbol{p}_i$ to $\boldsymbol{p}_m$ is defined as the orientation of vector $\overrightarrow{\boldsymbol{p}_i} - \overrightarrow{\boldsymbol{p}_m}$:

$$\Theta(i, m) = \angle(\overrightarrow{\boldsymbol{p}_i} - \overrightarrow{\boldsymbol{p}_m}) \in [-\pi, \pi] \quad . \tag{24}$$

Based on Eq. 23 and Eq. 24, a CS C is encoded in two $N \times N$ matrix E and Θ. Therefore, the length direction descriptor of a C is $\boldsymbol{f}_{25} = \{E, \Theta\}$.

Line Segment: Line Segment [39] descriptor $\boldsymbol{f}_{26}$ was designed for shape classification. Line segment descriptor is generated based on straight-line segment statistics. There are several characteristics of this descriptor: (1) It is simple and intuitive as it is directly generated based on connection lines between CS points. (2) It preserves the coarse-grained features hierarchically using different scales. (3) Built on the feature vectors, the distance between two CSs can be directly calculated by vector distance methods.



(a) C (b) Line Segment (c) Sub Box

Figure 10: Some rich descriptors for a CS C in (a). This group of CS descriptors will use their own distance methods for evaluation.

As shown in Figure 10(b), for each $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_N\}$ in the CS, they are able to compute a line segment to any other point. For scale invariance, they define a continuous portion of contour with size n, where n is given in terms of percentage (n is set from 5% to 85% in all experiment). So, given a starting point $\boldsymbol{p}_i$, the end point $\boldsymbol{p}_m$ of a line segment $\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_m}$ is selected by

$$m = (i + \frac{n}{100} * N) \mod N \quad . \tag{25}$$

where $i = 1, 2, \cdots, N$. This strategy for selecting the end point guarantees that they select the same curve portion in a CS at different scales. In order to characterise a CS through its set of line segments, they compute the average μ and standard deviation σ of all line segments at scale n. This yields the following feature vector:

$$F_n = [\mu_n, \sigma_n] \quad . \tag{26}$$

Thus, line segment descriptor $\boldsymbol{f}_{26}$ can be presented by a feature vector which analyses the CS at different sizes of the CS portion n by concatenating F_n.

Sub Box: Sub Box [9] descriptor $\boldsymbol{f}_{27}$ was proposed for shape matching. Different from most rich descriptors which preserve both fine- and coarse-grained CS features, Sub Box descriptor only preserves the coarse- and medium-grained CS features by using the full- and sub-bounding boxes. Thus, the distance between CSs can be calculated in

a negligible time while preserving a relatively promising accuracy.

As shown in Figure 10(c), in order to generate sub box descriptor, CS bounding box and equally high sub-boxes ($h_1 = h_2 = h_3$) are used for feature extraction: $\boldsymbol{p}_{c,1} \rightarrow \boldsymbol{p}_{c,4}$ is oriented vertically; $\boldsymbol{p}_{c,1}$, $\boldsymbol{p}_{c,2}$ and $\boldsymbol{p}_{c,3}$ are middle points of the top, middle, and bottom CS sub-segments, respectively. For a CS, its first element f_1 is equal to the CS point number N. Euclidean distance between CS endpoints determines the second element f_2. The area between the straight line connecting the CS endpoints and the CS itself is used as the third feature f_3. Before computing remaining features, CS is transformed into a normalised vertical orientation to ensure rotation invariance. The rest features $f_4, \cdots, f_{11}$ are calculated by the bounding box of the whole CS as well as the three equally high sub-boxes:

$$\begin{array}{cccc} f_4 = \dfrac{h}{w} & f_5 = \dfrac{h_1}{w_1} & f_6 = \dfrac{h_2}{w_2} & f_7 = \dfrac{h_3}{w_3} \\ f_8 = \dfrac{w_3 h_3}{w_1 h_1} & f_9 = \dfrac{w_2 h_2}{w_1 h_1} & f_{10} = \alpha_1 & f_{11} = \alpha_2 \end{array} \quad . \tag{27}$$

They also divide the non-ratio elements (f_1, f_2 and f_3) by a half of the bounding box perimeter for scale invariance. Finally, $\boldsymbol{f}_{27} = [f_1, f_2, \cdots, f_{11}]$.

3. Contour Segment Matching

We introduce the methods for calculating the distance $D(C_1, C_2)$ between two CSs C_1 and C_2. In particular, the matching and distance methods depend on types of CS descriptors presented in the previous section. After that, we introduce the open curve matching approach which is used for our experiments.

3.1. CS Matching: Simple Descriptors

For the simple descriptors $\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_9$, their feature vectors contain only one single value. Therefore, we calculate the distance $D(C_1, C_2)$ by:

$$D(C_1, C_2) = \frac{|\boldsymbol{f}_{1,j} - \boldsymbol{f}_{2,j}|}{\boldsymbol{f}_{1,j} + \boldsymbol{f}_{2,j}} \quad . \tag{28}$$

where $\boldsymbol{f}_{1,j}$ and $\boldsymbol{f}_{2,j}$ are the simple descriptors of C_1 and C_2, respectively, $j = 1, 2, \cdots, 9$. Here we can also calculate the distance $D(C_1, C_2)$ using other methods like simple difference. However, it may happen that the big difference values could dominate the CS retrieval results. Thus, Eq. 28 is designed to normalise the difference.

3.2. CS Matching: Signature-based Descriptors

Since signature-based descriptors $\boldsymbol{f}_{10}, \boldsymbol{f}_{11}, \cdots, \boldsymbol{f}_{17}$ are composed by one-dimensional feature vectors, there are two methods for calculating the distance $D(C_1, C_2)$: Point matching and vector distance. Let $\boldsymbol{f}_{1,j}$ and $\boldsymbol{f}_{2,j}$ be the signature-based CS descriptors for C_1 and C_2, respectively. Based on the description in Section 2.2, $\boldsymbol{f}_{1,j} = [f_{1,j}^1, f_{1,j}^2, \cdots, f_{1,j}^N]$ and $\boldsymbol{f}_{2,j} = [f_{2,j}^1, f_{2,j}^2, \cdots, f_{2,j}^N]$, $j = 10, 11, \cdots, 17$.

With the point matching method, we first calculate the differences $d(f_{1,j}^m, f_{2,j}^n)$ between point signature-based values ($m, n = 1, 2, \cdots, N$). Then, a matrix of differences between all CS points in C_1 and C_2 is generated:

$$d(f_{1,j}^m, f_{2,j}^n) = \begin{pmatrix} d(f_{1,j}^1, f_{2,j}^1) & d(f_{1,j}^1, f_{2,j}^2) & \cdots & d(f_{1,j}^1, f_{2,j}^N) \\ d(f_{1,j}^2, f_{2,j}^1) & d(f_{2,j}^2, f_{2,j}^2) & \cdots & d(f_{1,j}^2, f_{2,j}^N) \\ \vdots & \vdots & \cdots & \vdots \\ d(f_{1,j}^N, f_{2,j}^1) & d(f_{1,j}^N, f_{2,j}^2) & \cdots & d(f_{1,j}^N, f_{2,j}^N) \end{pmatrix} . \quad (29)$$

In order to find an optimum match of CS points from C_1 to C_2, we use the matching algorithms like DTW [31], DP [30] and Hungarian [29] on the matrix expressed in Eq. 29. Specifically, we employ the FastDTW [63] for computing DTW. FastDTW is able to find an accurate approximation of the optimal warp path between two time series. The FastDTW algorithm avoids the brute-force dynamic programming [30] approach of the standard DTW algorithm by using a multilevel approach. The time series are initially sampled down to a very low resolution. A warp path is found for the lowest resolution and "projected" onto an incrementally higher resolution time series. The projected warp path is refined and projected again to a higher resolution. The process of refining and projecting is continued until a warp path is found for the full resolution time series. For the DP, the distance between two CSs is calculated by the minimum number of substitutions, insertions and deletions required to transform a CS to another

one using their point features. In order to reduce the time complexity of traditional brute-force approach [30], we employ the solution proposed by Sellers [64]. Consequently, given two CSs with the same length, the time complexity for CS matching is reduced. This is because the employed method make a complete list of all pairs of intervals using given CSs, so that each pair displays a maximum local degree of similarity. Then the matching complexity is reduced by trading time for space. For the Hungarian algorithm, it was originally proposed by H. W. Kuhn in 1955 [29] for solving the assignment problem which consists of finding a maximum cardinality matching of maximum weight matching (or minimum weight perfect matching) in a weighted bipartite graph. With this approach, the correspondence between CS points is generated by minimising the global cost between CS point distances illustrated in Eq. 29.

The resulting distance values of the matched CS points can be denoted as $s_1, s_2, \cdots, s_N$ and the global similarity between C_1 and C_2 is calculated as follows:

$$D(C_1, C_2) = \frac{1}{N} \sum_{i=1}^{N} s_i \quad . \tag{30}$$

With vector distance computation, we directly calculate the CS distance using their feature vectors without considering any point matching. In this paper, we employ and evaluate the following measures to express the distance between $\boldsymbol{f}_{1,j}$, $\boldsymbol{f}_{2,j}$:

(1) Correlation [32]

$$D(C_1, C_2) = 1 - \frac{\sum_{m=1}^{N} (f_{1,j}^m - \frac{1}{N} \sum_{n=1}^{N} f_{1,j}^n)(f_{2,j}^m - \frac{1}{N} \sum_{n=1}^{N} f_{2,j}^n)}{\sqrt{\sum_{m=1}^{N} (f_{1,j}^m - \frac{1}{N} \sum_{n=1}^{N} f_{1,j}^n)^2 \sum_{m=1}^{N} (f_{2,j}^m - \frac{1}{N} \sum_{n=1}^{N} f_{2,j}^n)^2}} \quad . \tag{31}$$

(2) Histogram Intersection (HI) [33]

$$D(C_1, C_2) = 1 - \frac{\sum_{i=1}^{N} \min(f_{1,j}^i, f_{2,j}^i)}{\sum_{i=1}^{N} \max(f_{1,j}^i, f_{2,j}^i)} \quad . \tag{32}$$

(3) χ^2-Statistics [34]

$$D(C_1, C_2) = \sum_{i=1}^{N} \frac{(f_{1,j}^i - f_{2,j}^i)^2}{f_{1,j}^i + f_{2,j}^i} \quad . \tag{33}$$

(4) Hellinger (or Bhattacharyya Coefficient) [35]

$$D(C_1, C_2) = \sqrt{1 - \frac{\sum_{i=1}^{N} \sqrt{f_{1,j}^i f_{2,j}^i}}{\sqrt{\sum_{i=1}^{N} f_{1,j}^i \sum_{i=1}^{N} f_{2,j}^i}}} \quad . \tag{34}$$

The aforementioned distance measures are selected built on the concept of simpleness and applicability which are evaluated in [65]. However, there are more existing distance measures can be employed. In Section 6, we will discuss this extension as our future work.

3.3. CS Matching: Rich Descriptors

Different from simple and signature-based descriptors, feature vectors of rich descriptors $\boldsymbol{f}_{18}, \boldsymbol{f}_{19}, \cdots, \boldsymbol{f}_{27}$ are not uniformed. Different rich descriptors have their own way of matching CS points as well as calculating CS distances. The details are introduced as follows:

(1) For Height Function descriptor $\boldsymbol{f}_{18}$, to match two CSs C_1 and C_2, the distance between any pair of points should be computed. Let $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ denote CS points of C_1 and C_2, respectively, and F_1, F_2 denote their height features. The cost of matching $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is computed by comparing their height features:

$$d(\boldsymbol{p}_1, \boldsymbol{p}_2) = \sum_{i=1}^{N} \omega_i |f_1^i - f_2^i| \quad . \tag{35}$$

where ω_i is the weight coefficient for every component of the height feature. Here, high weights are put on CS points near to the center to tolerate CS deformations:

$$\omega_i = \frac{1}{\min\{i, N-i\}} \quad . \tag{36}$$

We compute the matching costs for every pair of points in C_1 and C_2 with Eq. 35 and generate a cost matrix similar to Eq. 29. We finally apply the matching algorithm like Hungarian [29], DP [30] or DTW [31] on the cost matrix to compute the optimal correspondence between sample points in two CSs. The distance between C_1 and C_2 is calculated by the method in Eq. 30.

(2) For Point Triangle $\boldsymbol{f}_{19}$, Contour Context $\boldsymbol{f}_{20}$, Beam Angle $\boldsymbol{f}_{21}$, Partial Contour $\boldsymbol{f}_{22}$, Opt Partial Contour $\boldsymbol{f}_{23}$ and Chord Distributions $\boldsymbol{f}_{24}$ descriptors, they are all constructed by the histograms of CS points. To match two CSs C_1 and C_2, the distance between any pair of points is computed by the standard histogram intersection and we get a cost matrix similar to Eq. 29. We apply the matching algorithm like Hungarian [29], DP [30] or DTW [31] on the cost matrix to compute the optimal correspondence between sample points in C_1 and C_2. Finally, the distance between C_1 and C_2 is calculated by the method in Eq. 30.

(3) For Length Direction descriptor $\boldsymbol{f}_{25}$, given two CSs C_1 and C_2 with the same length N, their distance and orientation matrices are $\{E^{C_1}, E^{C_2}\}$ and $\{\Theta^{C_1}, \Theta^{C_2}\}$, respectively. They define two affinity matrices that measure the similarity between C_1 and C_2. The first affinity matrix is based on comparison of distances between pairs of corresponding points:

$$A_E(C_1, C_2) = \exp\left(-\frac{(E^{C_1}(i,m) - E^{C_2}(i,m))^2}{(E^{C_1}(i,m)\sigma)^2}\right) \quad . \tag{37}$$

where σ represents the tolerance of distance differences (it is set to 0.2 in all our experiments). To make $A_E(C_1, C_2)$ invariant to scale, each distance is normalised by the distance between the first pair of points. The second affinity matrix is based on angle comparison of vectors connecting the corresponding pairs of points

$$A_\Theta(C_1, C_2) = \exp(-\frac{(\Theta^{C_1}(i,m) - \Theta^{C_2}(i,m))^2}{\delta^2} \quad . \tag{38}$$

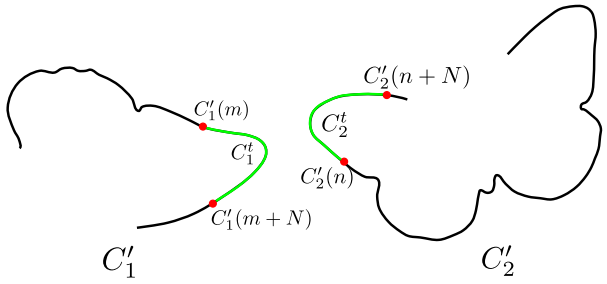
where δ represents the tolerance of angle differences (it is set to $\pi/4$ in all our experiments). Since both A_E and A_Θ are normalised, these are simply added to obtain the affinity matrix

$$A(C_1, C_2) = A_E(C_1, C_2) + A_\Theta(C_1, C_2) \quad . \tag{39}$$

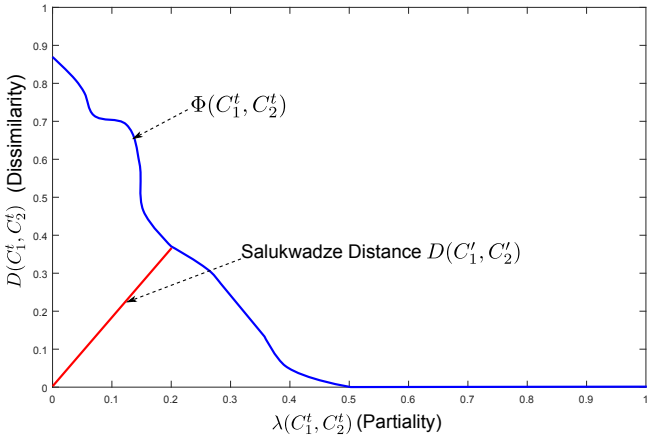
It can be observed that A is a $N \times N$ matrix representing the similarities of corresponding point pairs in C_1 and C_2. We finally apply the matching algorithm like Hungarian [29], DP [30] or DTW [31] on the cost matrix to compute an optimal correspondence between sample points in two CSs. The distance between C_1 and C_2 is calculated by the method in Eq. 30.

(4) For Line Segment $\boldsymbol{f}_{26}$ and Sub Box $\boldsymbol{f}_{27}$, both descriptors are organised by a multiple-dimensional feature vector. Similar to the signature-based descriptors, given two CSs C_1 and C_2, we can use the vector distance methods like Correlation (Eq. 31), Histogram Intersection (Eq. 32), χ^2-statistics (Eq. 33) and Hellinger (Eq. 34) for calculating their distance.

3.4. Open Curve Matching



(a) Open curve matching using CSs (green lines)



(b) Salukwadze distance (the red line)

Figure 11: Open curve matching and the open curve similarity method.

Given two open curves C'_1 and C'_2 with size N_1 and N_2, respectively, our aim is to find the similar parts among two open curves using CS descriptors and their matching algorithms. Specifically, one has to compare all sub-blocks of the open curve to find all matching possibilities and lengths. In each sub-block, CS descriptors and the matching

algorithms are employed for searching point correspondences as well as calculating the similarities. Then, the similarity between C_1' and C_2' is calculated based on their similar parts. For notational simplicity we assume that $N_1 \leq N_2$. In terms of searching similar parts between C_1' and C_2' this equals to finding two CSs C_1 and C_2 with size N starting at the curve point $C_1'(m)$ and $C_2'(n)$ which yield the smallest distance $D(C_1, C_2)$, $m \leq N_1$, $n \leq N_2$, $N \leq \min(N_1, N_2)$ (Figure 11(a)). Therefore, the problem is reduced to searing the proper (m, n, N) combination which produces a small distance $D(C_1, C_2)$.

Nevertheless, this brute-force method becomes inefficient for open curves with a larger number of points since all different matching possibilities and chain lengths have to be considered for finding such blocks. In order to overcome this limitation, some algorithmic optimisation methods are proposed [38, 66, 37]. One typical way is based on an adaption of the Summed-Area-Table (SAT) approach [67, 38]. A SAT is a data structure and algorithm for a quick and efficient generation of the sum of values in a rectangular subset of a grid. More specifically, for calculating the similarities for all possible configuration triplets m, n, N, N_2 integral images $Int^1, Int^2, \cdots, Int^{N_2}$ each of length N_1 are built for N_2 descriptor difference matrices M_D^i defined by

$$M_D^i = D(C_1'(1 : N_1), C_2'(i : i + N_1 - 1)) \tag{40}$$

where i denotes the length of CS, $i = 1, 2, \cdots, N_2$. However, it does not make any sense if i is too small to distinguish two CSs. In practice, we set the start value of i based on the length of open curves. The difference matrices M_D^i represent the N_2 possibilities to match two CSs. Based on these N_2 integral images $Int^1, Int^2, \cdots, Int^{N_2}$ the different values $D(C_1, C_2)$ can be calculated for every block of any size starting at any point on the diagonal in constant time. Finally, the similar parts $(C_1^\star, C_2^\star)$ between two open curves are identified by comparing those similarities to a fixed threshold:

$$(C_1^\star, C_2^\star) = \{(C_1^1, C_2^1), (C_1^2, C_2^2), \cdots, (C_1^H, C_2^H)\} \tag{41}$$

where H is the number of similar parts between C_1' and C_2'.

In order to calculate the similarity between open curves C_1' and C_2' for some tasks, e.g. open curve retrieval, we can directly use the mean similarities between their

matched parts. However, for two matched CSs C_1^t and C_2^t, $t = 1, 2, \cdots, H$, if their length N is becoming smaller, the similarity between C_1 and C_2 is becoming higher which seriously impacts the similarity between C_1' and C_2'. The main reason is the dropping of CS features along with the decreasing of N. Therefore, the Pareto-framework [68, 38] is employed for quantitative interpretation of partial similarity. Pareto-framework illustrates a way to select a multi-constrained path that can meet the optimal requirements. Specifically, as shown in Figure 11(b), we define two quantities: partiality $\lambda(C_1^t, C_2^t)$, which describes the length of the CS (the higher the value the smaller the CS), and the distance $D(C_1^t, C_2^t)$, which measures the dissimilarity. A Pareto optimum is defined by $\Phi(C_1^t, C_2^t)$ which is a pair of partiality and dissimilarity values that fulfils the criterion of lowest dissimilarity for the given partiality. Thus, $\Phi(C_1^t, C_2^t) = (\lambda(C_1^t, C_2^t), D(C_1^t, C_2^t))$. All Pareto optimums can be visualized as a curve, referred to as the set-valued Pareto frontier. Therefore, by focusing on the discretisation defined by our point sampling, we can calculate a global optimal Pareto frontier by returning the minimum descriptor difference for all partialities.

Finally, to achieve a similarity value between C_1' and C_2', the so-called Salukwadze distance is calculated based on the Pareto frontier by

$$D(C_1', C_2') = \inf_{(C_1^\star, C_2^\star)} |\Phi(C_1^\star, C_2^\star)|_1 \tag{42}$$

where $|\cdots|_1$ is the L1-norm of the vector. Therefore, $D(C_1', C_2')$ measures the distance from the coordinate $(0, 0)$ to the closet point on the Pareto frontier. The Salukwadze distance is then returned as the open curve similarity score.

4. Experimental Design

In this section, the experimental setup to evaluate the different CS descriptors is outlined. First, three datasets including CSs, open curves and sketches are described. Then, evaluation details in the settings of invariant analysis, matching performance, computational complexity in terms of time complexity and runtime are discussed. Finally, we briefly introduce the experimental environment.

4.1. Datasets

We aim to quantitatively evaluate the performance of CS descriptors for CS matching and their applications. However, to the best of our knowledge, there are no desirable datasets specifically for this task. This is because in most existing CS-related applications [54, 55, 56, 58, 6, 60, 8], researchers propose CS descriptors only for their own specific scenarios. Moreover, we cannot directly employ the existing datasets [38, 62, 39, 9] for CS evaluation since they contain only images or shapes rather than CS. Thus, we designed and collected three datasets for our experiments: (1) MPEG7 CS dataset for CS matching, (2) ETHZ CS dataset for open curve matching and (3) Sketching CS dataset for hand sketching matching.

MPEG7 CS: MPEG7 [69] dataset is a standard and commonly used shape dataset

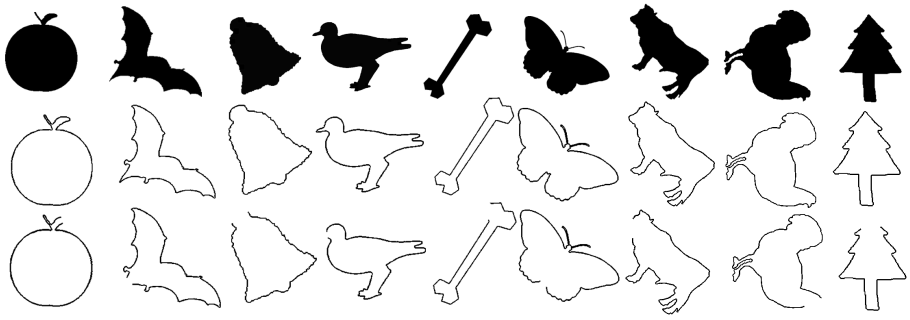


Figure 12: Sample shapes from MPEG7 database [69] for generating CSs. The correlated shape contours and CSs are shown in the second and third row.

for evaluating shape matching and classification. The total number of images in the MPEG7 database is 1400: 70 classes of various shapes, each class with 20 images. Built on this dataset, we manually generate the MPEG7 CS dataset using its shape contours (Figure 12). Specifically, for the shapes from the same class in MPEG7, we first extract their contours and then manually remove the same part from contours according to their classes (e.g, the mouth part of cattle and the tail part of bats, etc.). Finally, the MPEG7 CS dataset is generated with 1400 CSs. In order to easily and fairly evaluate the performance of CS descriptors, the CSs we generated for this dataset

have the same number of CS points. With this strategy, this dataset is also used for comparing the runtime between CS descriptors.

ETHZ CS: ETHZ [70] is a dataset for testing object class detection algorithms. It

Figure 13: Sample images from ETHZ database [70]. The correlated object contours and CSs are shown in the second and third rows.

contains 255 test images and features five diverse shape-based classes (apple logos, bottles, giraffes, mugs and swans). As shown in Figure 13, based on the shapes from ground truth, we manually generate the open curves from shape contours. Since some images have more than one object being detected, the total number of open curves is higher than the number of images. More specifically, there are 44 open curves for apple logos (the right half part), 55 open curves for bottles (the right half part), 91 open curves for giraffes (the upper neck part), 48 open curves for mugs (the right half part with handle) and 32 open curves for swans (the upper half part with head). In total, there are 271 open curves in the ETHZ CS dataset. This dataset is used for evaluating the open curve matching performance.

Sketching CS: Sketching CS is a dataset collected by ourselves. Figure 14 shows 18 different sketches which are used for generating Sketching CS dataset. These sketches are commonly used for the application of sketch-based object retrieval [21]. We first

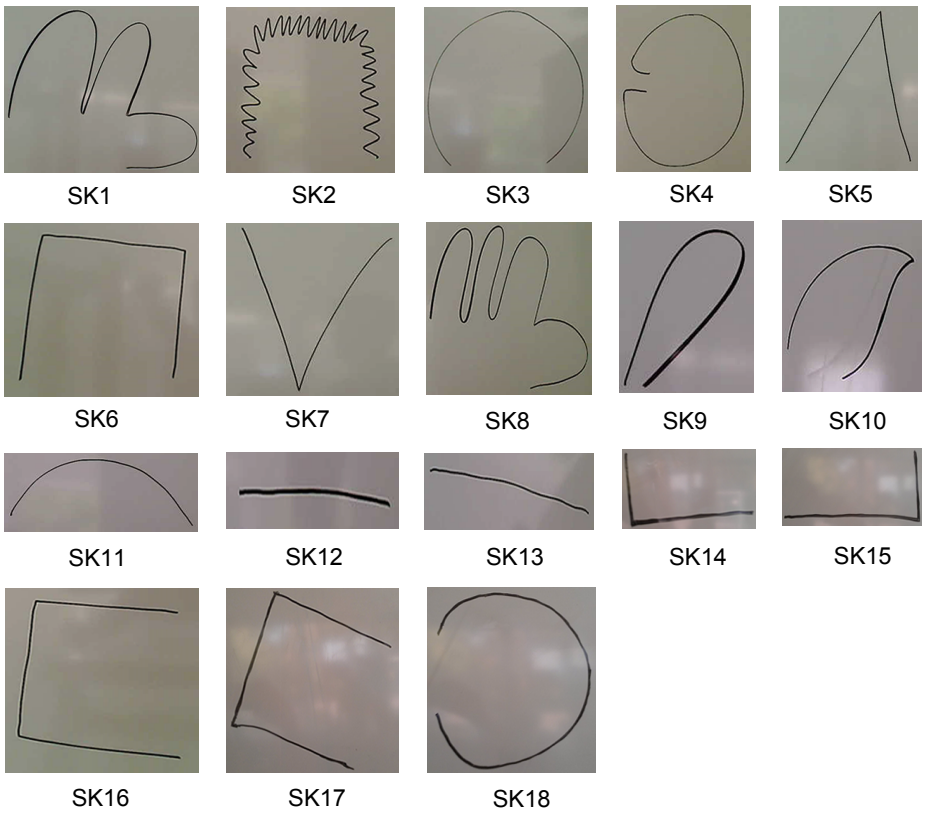


Figure 14: Collected sketches and their names for Sketching CS dataset.

collect the sketching images from a white board. After that, all the sketches are segmented and processed using the method introduced in Section 1. Finally, the Sketching CS dataset is generated with 18 open curves. This dataset is used for evaluating the difference between CS descriptors and human perception for sketching matching.

4.2. Experiment Types

Accurate matching requires CS descriptors which can effectively find perceptually similar curves from a database. Perceptually similar curves usually means rotated, translated and scaled curves. A desirable CS descriptor should be application independent rather than only performing well for certain type of CSs. Low computation complexity is also an important characteristic of a desirable CS descriptor. Depending

on these requirements, we establish three types of experiments in terms of invariance properties, matching performance as well as the computation complexity.

4.2.1. Experiment 1: Invariance Properties

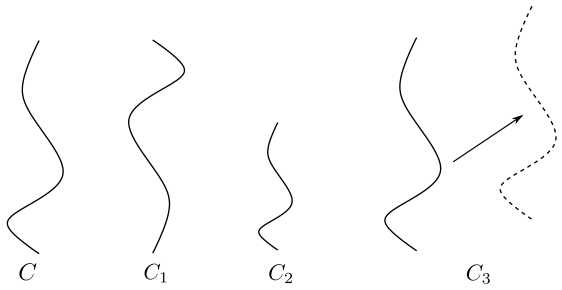


Figure 15: Invariant properties of a CS C: rotation (C_1), scaling (C_2) and translation C_3 invariants.

Descriptors are often classified according to their level of invariance to certain geometrical transformations. For example, descriptors invariant to Euclidean transforms remain unchanged after applying translation and rotation. In this paper, we mainly assess three invariance properties of each CS descriptor: rotation invariance, scaling invariance and translation invariance. As shown in Figure 15, rotation and scaling invariants mean that the CS features remain the same even when the CS is rotated and rescaled. Translation invariant means that two CSs can still be correctly matched even when every CS point is moved into a constant distance in a specified direction.

4.2.2. Experiment 2: Matching Performance

As illustrated in Table 2, we evaluate the CS descriptors with multiple settings. For the simple CS descriptors, since the distance between two CSs can be directly calculated using their single feature values, there are no matching processes involved (Section 3.1). For the signature-based CS descriptors, each descriptor is organised by a single multi-dimensional vector that represents characteristics of CS points. Thus, there are two methods for calculating the distance between CSs: Point matching and vector distance (Section 3.2). Therefore, we split the evaluation into two scenarios: One uses point matching methods and another one uses vector distance methods. For the rich

CS Descriptors	Distance Functions		Matching Algorithms	Lengths	Datasets
Simple ($\boldsymbol{f}_1$ - $\boldsymbol{f}_9$)	Difference Value		NULL	L1, L2, L3, L4	(1) ETHZ CS (2) Sketching CS (3) MPEG7 CS
Signature-based ($\boldsymbol{f}_{10}$ - $\boldsymbol{f}_{17}$)	Difference Value		(1) DTW (2) DP (3) Hungarian	L1, L2, L3, L4	(1) ETHZ CS (2) Sketching CS (3) MPEG7 CS
	(1) Correlation (2) Histogram Intersection (3) χ^2-Statistics (4) Hellinger		NULL		
Rich ($\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{27}$)	$\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{25}$	Proposed Distances	(1) DTW (2) DP (3) Hungarian	L1, L2, L3, L4	(1) ETHZ CS (2) Sketching CS (3) MPEG7 CS
	$\boldsymbol{f}_{26}$, $\boldsymbol{f}_{27}$	(1) Correlation (2) Histogram Intersection (3) χ^2-Statistics (4) Hellinger	NULL		

Table 2: Experiment settings for evaluating the matching performance of CS descriptors. L1, L2, L3 and L4 denote the sampling densities that are used for generating CS descriptors. In our experiments, L1 = 25%, L2 = 50%, L3 = 75% and L4 = 100%. DTW, DP and Hungarian denote the matching algorithms Dynamic Time Warping [31], Dynamic Programming [30] and Hungarian [29], respectively. NULL means this part is not considered in our experiments.

descriptors, since the feature structures of $\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{25}$ are the histograms that represent the features of each CS point (Section 3.3), we can use the same histogram intersection method for calculating the point distance and then applying different matching algorithms for finding point correspondences. However, for the rich descriptor $\boldsymbol{f}_{26}$ and $\boldsymbol{f}_{27}$, rather than the feature vectors for each CS point, both descriptors are organised by a feature vector representing the CS global features. Therefore, we apply and compare the different vector distance methods (Eq. 31, Eq. 32, Eq. 33 and Eq. 34) for these two rich CS descriptors.

For all descriptors, as shown in Table 2, we generate their features using four sampling densities for selecting sample points from a CS: L1 = 25%, L2 = 50%, L3 = 75% and L4 = 100%. Depending on this strategy, we want to check the influence of sample point density to the the CS matching performance. With the combinations of different

distance functions, matching algorithms and CS lengths, the experiments are applied on the proposed three datasets:

MPEG7 CS: We use this dataset to evaluate the CS matching performance by applying a CS retrieval scenario. In order to quantitatively evaluate the matching performance, we employ the so-called bulls-eye score [69] for evaluation. Given a query CS, we retrieve the 40 most similar CSs from the database and count the number of CSs belonging to the same class as the query. The bulls-eye score is the ratio of the total number of correctly matched CSs to the number of all the possible matches (which is 20×1400). Thus, the best score is 100 percent. However, as discussed in [69], the 100% bulls-eye score is impossible to obtain since some classes contain objects whose shapes are significantly different so that it is not possible to group them into the same class using only their shapes.

ETHZ CS: The experiment on this dataset aims to evaluate the open curve matching performances using different CS descriptors. To do so, we conduct an open curve retrieval scenario using the method introduced in Section 3.4. Specifically, we use each 271 curve as a query and retrieve the 100 most similar curves among the whole dataset. The final evaluation is composed of two parts: recall and precision. Both parts are calculated by the mean value of recall and precision from all classes [71]. For example, since there are 44 open curves in the Apple logo class, the precision (or recall) for this class is calculated by taking the average of precisions (or recalls) that are obtained using each of 44 open curve as a query.

Sketching CS: Sketching CS dataset is used for evaluating the sketching matching using CS descriptors. In order to collect the ground truth, we first carried out an on-line questionnaire for selecting the most similar sketches to the query by human perception. Specifically, we organise 10 groups of sketches in which five different sketches are randomly combined. In each group, we use one sketch as a query and the other four as candidates. However, among four candidates in each group, we need to ensure that at least one candidate is perceptually similar to the query since this makes it easier for us to capture the meaningful and pertinent results from both humans and machines [72]. In total, we received 71 effective results in each group. Table 3 illustrates the sketch groups and voting results. With the same query and candidate settings in Table 3, we

	Query	Candidates				Human Perception Results (%)			
		1	2	3	4	1	2	3	4
1	SK6	SK3	SK2	SK11	SK18	44.44	**55.56**	0	0
2	SK13	SK11	SK14	SK5	SK16	**77.78**	5.56	5.56	11.11
3	SK14	SK16	SK17	SK3	SK15	27.78	5.56	0	**66.67**
4	SK12	SK13	SK14	SK11	SK5	**88.89**	0	11.11	0
5	SK11	SK16	SK3	SK12	SK5	0	**61.11**	22.22	16.67
6	SK16	SK18	SK4	SK10	SK17	5.56	5.56	0	**88.89**
7	SK10	SK18	SK7	SK9	SK5	11.11	0	**88.89**	0
8	SK8	SK5	SK1	SK10	SK9	0	**88.89**	11.11	0
9	SK18	SK11	SK4	SK3	SK10	0	**55.56**	38.89	5.56
10	SK5	SK9	SK7	SK10	SK11	11.11	**77.78**	0	11.11

Table 3: Ten groups of sketches for comparing the difference of sketching matching between human perception and CS descriptors. Results of human perception are illustrated in the last four columns which are calculated by the number of votes on each candidate divided by the total number of votes (71 in our case) in their groups. The candidates with the maximum votes (marked with bold) in each group will be used as the ground truth.

apply the sketching retrieval using the open curve matching method with different CS descriptors. Finally, in each group, we compare the most voted sketching (marked with bold in Table 3) to the retrieved sketching which has the highest similarity value to the query and see whether they are the same sketching or not. We count the number of these coincidences among ten groups and divide it by 10. This value can be considered as the performance of sketching matching with different CS descriptors since it shows the similarity between a CS descriptor and the human perception.

4.2.3. Experiment 3: Computation Complexity

In this part, we examine the computation complexity of each CS descriptor with two parts: Time complexity analysis and the real runtime. More concretely, in the first part, we will theoretically analyse the computation complexity for CS descriptor generating and matching [73]. In the second part, we will use the full retrieval time of 1400 queries on the MPEG7 CS dataset as the runtime. The main reason is that there are abundant CSs in this dataset which can differ the retrieval time with different CS descriptors distinctly. In order to avoid the overlapping time for CS feature generation,

we generate all features before the retrieval process. Therefore, the final runtime is the summed time for CS feature generation and retrieval.

4.3. Experimental Environment

The experiments in this paper are delivered on two platforms: Cluster and laptop. Feature generation and full object retrieval experiments are accomplished on *Horus*, a cluster provided by the University of Siegen, which includes 136 nodes, each consisting of 2 Intel Xeon X5650 with 2,66 GHz and 48 GB memory. With this cluster, our massive experiments can be finished efficiently. In order to fairly compare the runtime of each CS descriptor, the runtime estimation experiments are finished on a laptop with Inter Core i7 2.2GHz CPU, 8.00GB memory and 64-bit Windows 8.1 OS. We do not use cluster for runtime estimation mainly because the task distribution in *Horus* may cause an unfair comparison since various processes in each node are different.

All methods in our experiment are implemented in Matlab. In the *Horus*, codes are executed in Matlab R2014b 64-bit, while Matlab R2015a 64-bit is used on the laptop.

5. Results

In this section, the results referring to the experimental design in Section 4 are reported. We first illustrate the theoretical invariance properties of each CS descriptor. Then, the matching performance on three datasets is reported. We also report the computation complexity of CS descriptors in terms of feature generation and matching. Finally, some observations and recommendations formed on the results are concluded and discussed.

5.1. Experiment 1: Invariant Properties

Table 4 illustrates the theoretical invariance properties of different CS descriptors. Firstly, by observing the results with respect to the CS translation, all the CS descriptors except Comcoor ($\boldsymbol{f}_{10}$) are invariant to this property. Different from other CS descriptors which are generated by distances and angles, Comcoor ($\boldsymbol{f}_{10}$) is represented by the CS coordinate. Therefore, Comcoor is not invariant to rescaling, rotation and translation. Secondly, we can clearly observe that all the simple CS descriptors are also

Simple CS Descriptors				Signature-based CS Descriptors				Rich CS Descriptors			
Descriptors	SI	RI	TI	Descriptors	SI	RI	TI	Descriptors	SI	RI	TI
$\boldsymbol{f}_1$	+	+	+	$\boldsymbol{f}_{10}$	-	-	-	$\boldsymbol{f}_{18}$	+	+	+
$\boldsymbol{f}_2$	+	+	+	$\boldsymbol{f}_{11}$	-	+	+	$\boldsymbol{f}_{19}$	+	+	+
$\boldsymbol{f}_3$	+	+	+	$\boldsymbol{f}_{12}$	+	+	+	$\boldsymbol{f}_{20}$	+	-	+
$\boldsymbol{f}_4$	+	+	+	$\boldsymbol{f}_{13}$	+	+	+	$\boldsymbol{f}_{21}$	+	+	+
$\boldsymbol{f}_5$	+	+	+	$\boldsymbol{f}_{14}$	-	$+^*$	+	$\boldsymbol{f}_{22}$	+	+	+
$\boldsymbol{f}_6$	+	+	+	$\boldsymbol{f}_{15}$	-	+	+	$\boldsymbol{f}_{23}$	+	+	+
$\boldsymbol{f}_7$	+	+	+	$\boldsymbol{f}_{16}$	+	+	+	$\boldsymbol{f}_{24}$	+	+	+
$\boldsymbol{f}_8$	+	+	+	$\boldsymbol{f}_{17}$	+	+	+	$\boldsymbol{f}_{25}$	+	+	+
$\boldsymbol{f}_9$	+	+	+					$\boldsymbol{f}_{26}$	+	+	+
								$\boldsymbol{f}_{27}$	+	+	+

Table 4: Invariant properties of different CS descriptors. SI, RI and TI denote the scale invariance, rotation invariance and translation invariance. The existence and lack of an invariance are indicated with '+' and '-', respectively. Regarding $+^*$, if the sampling is dense enough, then $\boldsymbol{f}_{14}$ is rotation invariant, and vice versa.

invariant to scaling and rotation. This is because these descriptors are coarse-grained and generated by only considering global properties of CSs. Thirdly, some signature-based descriptors, such as the Area Function ($\boldsymbol{f}_{14}$) and Triangle Area ($\boldsymbol{f}_{15}$), do not perform well for scaled CS. In practice, we can normalise them by the CS length or other CS simple descriptors. These descriptors thereby become scale-invariant. Lastly, except Contour Context ($\boldsymbol{f}_{20}$), all the rich descriptors comply with the three invariant properties. The main reason is that the shape context [59] feature is not rotation invariant. As discussed in [59], instead of using the absolute frame for computing the shape context at each point, one can use a relative frame, by treating the tangent vector at each point as the positive $x-$axis. This way, the reference frame turns into the tangent angle, and the result is completely rotation invariant. However, in a CS, many points are not well-defined or reliable tangents. Moreover, many local appearance features lost their discriminative power if they were not measured in the same coordinate system. Therefore, Belongie et al. [59] suggested that the rotation invariance can be applied if it is desirable for an application. Essentially, increasing the degree of invariance generally decreases the distinctiveness and thus weakens the distinctiveness property [8]. In many applications, complete invariance degrades the recognition performance, e.g.

when distinguishing "6" from "9" rotation invariance would be completely inappropriate. Therefore, invariance operation should be applied according to applications and datasets.

5.2. Experiment 2: Matching Performance

We report the matching performance on different datasets: MPEG7 CS is designed for evaluating the CS matching performance using bulls-eye scores, ETHZ CS is employed for evaluating open curve matching using recall and precision values, Sketching CS is collected for evaluating the similarity between human perception and CS descriptors using similarity values.

5.2.1. Evaluation on MPEG7 CS dataset

MPEG7 CS Dataset - Simple CS Descriptors				
Descriptors	L1	L2	L3	L4
$\boldsymbol{f}_1$	3.7	3.7	3.7	3.7
$\boldsymbol{f}_2$	3.7	3.7	3.7	3.7
$\boldsymbol{f}_3$	**22.3**	**22.8**	**22.8**	**22.8**
$\boldsymbol{f}_4$	**22.9**	**24.0**	**24.1**	**23.4**
$\boldsymbol{f}_5$	**22.3**	**22.8**	**22.8**	**22.8**
$\boldsymbol{f}_6$	18.7	18.9	18.9	18.9
$\boldsymbol{f}_7$	16.6	15.9	15.7	16.0
$\boldsymbol{f}_8$	3.7	3.7	3.7	3.7
$\boldsymbol{f}_9$	18.7	18.9	18.9	18.9

Table 5: CS matching results using Simple CS descriptors on MPEG7 CS Dataset. Eccentricity ($\boldsymbol{f}_3$), Bending ($\boldsymbol{f}_4$) and Rectangularity ($\boldsymbol{f}_5$) outperform the other descriptors (marked with bold).

Table 5 illustrates the CS retrieval results on MPEG7 CS dataset using simple CS descriptors. We can see that Eccentricity ($\boldsymbol{f}_3$), Bending ($\boldsymbol{f}_4$) and Rectangularity ($\boldsymbol{f}_5$) outperform the other descriptors in which Bending ($\boldsymbol{f}_4$) achieves the best performance on MPEG7 CS dataset. Moreover, considering the performance on different lengths, we can see that for each simple descriptor, the scores are almost the same. Therefore, all simple CS descriptors are robust to the CS length changing. The rationale behind

this is that simple CS descriptors are calculated by considering only global coarse-grained CS features. Even some fine-grained features get lost because of small number of sample points, the global features remain the same.

MPEG7 CS Dataset - Signature-based CS Descriptors (Point Matching Methods)												
Descriptors	DTW [31]				DP [30]				Hungarian [29]			
	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{10}$	62.1	62.3	62.3	62.4	50.0	49.9	50.6	50.9	**63.3**	**64.1**	**64.2**	**64.4**
$\boldsymbol{f}_{11}$	60.4	61.6	61.8	61.8	58.1	60.7	61.1	61.4	48.0	48.3	48.3	48.3
$\boldsymbol{f}_{12}$	54.5	55.9	60.0	61.4	54.9	55.7	59.6	60.8	51.1	51.4	50.0	49.8
$\boldsymbol{f}_{13}$	50.3	52.8	54.8	55.9	49.3	52.1	54.2	55.0	46.0	48.9	49.6	48.9
$\boldsymbol{f}_{14}$	55.0	57.7	57.7	59.4	50.8	54.5	53.7	56.4	41.9	43.6	38.6	42.4
$\boldsymbol{f}_{15}$	58.3	59.6	60.0	59.9	53.8	55.7	56.3	56.6	49.0	49.4	49.6	49.7
$\boldsymbol{f}_{16}$	3.2	2.9	2.9	2.9	3.3	3.0	2.9	2.9	3.3	2.9	2.9	2.9
$\boldsymbol{f}_{17}$	15.4	14.5	12.4	10.8	41.1	40.3	36.3	32.4	38.5	39.9	39.9	40.6

Table 6: CS matching results with Signature-based CS descriptors using point matching methods on MPEG7 CS Dataset. CS descriptor Comcoor ($\boldsymbol{f}_{10}$) with Hungarian matching method achieves the best performance (marked with bold).

MPEG7 CS Dataset - Signature-based CS Descriptors (Vector Distance Methods)																
Descriptors	Correlation				HI				χ^2-Statistics				Hellinger			
	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{10}$	8.2	7.1	7.1	6.9	13.5	14.0	14.5	14.3	55.7	55.8	56.5	56.6	**63.7**	**63.6**	**63.7**	**63.5**
$\boldsymbol{f}_{11}$	5.6	5.6	5.6	5.5	5.5	5.4	5.4	5.4	63.0	62.7	62.8	62.6	**63.7**	**63.6**	**63.7**	**63.5**
$\boldsymbol{f}_{12}$	5.9	5.8	5.9	5.9	7.1	7.0	6.8	6.6	44.2	47.4	50.0	49.4	42.7	41.8	44.8	43.7
$\boldsymbol{f}_{13}$	6.0	6.3	6.3	6.2	6.0	8.7	14.0	19.0	35.1	20.6	16.3	12.2	55.3	52.6	49.6	45.7
$\boldsymbol{f}_{14}$	5.6	5.6	6.2	5.6	5.1	5.1	5.1	5.1	54.4	54.8	54.8	54.3	49.7	49.2	47.7	46.3
$\boldsymbol{f}_{15}$	5.8	5.9	5.7	5.6	5.5	5.5	5.5	5.7	36.2	34.5	35.3	32.1	51.1	51.8	51.9	51.3
$\boldsymbol{f}_{16}$	3.3	3.0	2.9	2.9	2.9	2.9	2.9	2.9	3.2	2.9	2.9	2.9	2.9	2.9	2.9	2.9
$\boldsymbol{f}_{17}$	44.6	39.5	33.0	27.3	48.9	37.5	28.0	24.7	0.07	0.15	0.17	0.18	42.0	30.4	22.3	18.1

Table 7: CS matching results with Signature-based CS descriptors using vector distance methods on MPEG7 CS dataset. Both Comcoor ($\boldsymbol{f}_{10}$) and Cendistance ($\boldsymbol{f}_{11}$) with Hellinger distance method achieve the best performance (marked with bold).

For the signature-based CS descriptors on MPEG7 CS dataset, in Table 6, Comcoor ($\boldsymbol{f}_{10}$) with Hungarian matching method achieves the best bulls-eye score. In Table 7,

both Comcoor ($\boldsymbol{f}_{10}$) and Cendistance ($\boldsymbol{f}_{11}$) with Hellinger distance method obtain the best score. Among all results, Comcoor ($\boldsymbol{f}_{10}$) with Hungarian matching method achieves the best performance (64.4% bulls-eye score). Similar to the ETHZ dataset, we can observe that the CS matching performance is enhanced and damaged dramatically if the matching algorithms or vector distance methods are changed. Moreover, for most signature-based CS descriptors, CS retrieval using point matching methods performs much better than the vector distance methods.

MPEG7 CS Dataset - Rich CS Descriptors ($\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{25}$)												
Descriptors	DTW [31]				DP [30]				Hungarian [29]			
	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{18}$	64.9	64.7	63.7	63.4	70.8	74.8	76.2	76.9	62.9	70.7	72.7	74.0
$\boldsymbol{f}_{19}$	40.9	39.0	36.9	35.8	68.5	69.5	70.3	70.5	69.8	70.1	70.4	70.3
$\boldsymbol{f}_{20}$	64.2	64.5	64.5	64.5	62.4	63.9	64.2	64.4	66.2	67.7	68.1	68.4
$\boldsymbol{f}_{21}$	72.8	75.1	74.6	73.6	73.1	75.1	75.3	73.8	**79.6**	**79.6**	**77.4**	**73.9**
$\boldsymbol{f}_{22}$	64.6	64.6	64.8	64.4	69.2	73.3	76.6	77.4	64.8	66.4	67.2	66.5
$\boldsymbol{f}_{23}$	57.6	60.1	61.0	61.2	48.8	52.6	54.2	54.2	47.4	51.5	53.4	53.5
$\boldsymbol{f}_{24}$	66.3	66.0	65.9	65.3	70.6	73.8	76.5	77.3	69.4	72.8	74.3	75.0
$\boldsymbol{f}_{25}$	64.2	64.6	64.7	64.8	64.1	64.3	64.3	64.4	64.1	64.3	64.4	64.4

Table 8: CS matching results using Rich CS descriptors ($\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{25}$) on MPEG7 CS Dataset.

MPEG7 CS Dataset - Rich CS Descriptors ($\boldsymbol{f}_{26}$ and $\boldsymbol{f}_{27}$)																
Descriptors	Correlation				HI				χ^2-Statistics				Hellinger			
	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{26}$	70.8	70.7	70.5	70.3	70.0	71.0	71.2	71.2	72.7	73.2	73.3	73.3	74.5	74.8	74.8	74.8
$\boldsymbol{f}_{27}$	5.8	5.9	5.7	5.6	5.5	5.5	5.5	5.7	36.2	34.5	35.3	32.1	51.1	51.8	51.9	51.3

Table 9: CS matching results using Rich Descriptors ($\boldsymbol{f}_{26}$ and $\boldsymbol{f}_{27}$) on MPEG7 CS Dataset.

Table 8 and Table 9 illustrate the CS retrieval performance using rich CS descriptors. Among all rich descriptors, Beam Angle ($\boldsymbol{f}_{21}$) achieves promising results in all three matching methods, in which Beam Angle ($\boldsymbol{f}_{21}$) with Hungarian matching method achieve the best score (79.6% bulls-eye score). For most rich descriptors ($\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{26}$), they are relatively robust to the CS length and matching algorithm changing. However, for the Sub Box ($\boldsymbol{f}_{27}$), its performance is high related to the vector distance methods in

which using the Hellinger method outperforms the other vector distance methods. The main reason is that the Sub Box descriptor is composed by a 11-dimensional feature vector which can only capture the coarse-grained features. Therefore, its performance highly depends on the vector distance methods.

Comparing the performance of CS retrieval using three types of CS descriptors in MPEG7 CS dataset, we can draw the following observations: (1) Most rich descriptors have a better performance than the signature-based descriptors. (2) Most signature-based CS descriptors outperform the simple CS descriptors. (3) For most signature-based CS descriptors, point matching strategy performs better than the vector distance strategy for CS matching.

5.2.2. Evaluation on ETHZ CS dataset

ETHZ CS Dataset - Simple CS Descriptors (%)								
Descriptors	L1		L2		L3		L4	
	P	R	P	R	P	R	P	R
$\boldsymbol{f}_1$	33.9	56.8	34.0	56.9	34.1	57.1	34.0	57.0
$\boldsymbol{f}_2$	42.5	75.1	42.8	75.5	42.7	75.5	42.7	75.4
$\boldsymbol{f}_3$	**55.0**	**90.0**	**55.1**	**90.1**	**55.1**	**90.1**	**55.1**	**90.1**
$\boldsymbol{f}_4$	48.1	80.7	48.8	81.7	48.9	81.9	49.3	82.4
$\boldsymbol{f}_5$	**55.0**	**90.0**	**55.1**	**90.1**	**55.1**	**90.1**	**55.1**	**90.1**
$\boldsymbol{f}_6$	52.5	87.9	52.5	87.8	52.5	87.9	52.5	87.9
$\boldsymbol{f}_7$	37.1	61.1	37.1	60.8	36.1	58.8	36.4	59.7
$\boldsymbol{f}_8$	38.1	65.7	38.1	65.6	38.2	65.8	38.2	65.8
$\boldsymbol{f}_9$	52.5	87.9	52.5	87.8	52.5	87.9	52.5	87.9

Table 10: Open curve matching results using Simple CS descriptors on ETHZ CS Dataset.

Table 10 shows Precisions (P) and Recalls (R) of simple CS descriptors. As seen from this figure, Eccentricity ($\boldsymbol{f}_3$) and Rectangularity ($\boldsymbol{f}_5$) outperform the other descriptors (90.1% recall). Moreover, considering the performance on different lengths, like MPEG7 CS dataset, all simple CS descriptors are robust to CS length changing.

For the signature-based CS descriptors, as shown in Table 11 and Table 12, we can clearly observe that for both point matching and vector distance strategies, matching performances are highly related to the matching methods. For two strategies, point

ETHZ CS Dataset - Signature-based CS Descriptors (Point Matching Methods, %)													
Descriptors		DTW [31]				DP [30]				Hungarian [29]			
		L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{10}$	P	52.6	52.8	52.8	52.8	34.7	34.6	34.6	34.6	46.7	49.6	49.6	49.6
	R	86.0	86.2	86.3	86.4	58.3	58.0	58.1	58.0	81.3	81.3	81.2	81.2
$\boldsymbol{f}_{11}$	P	32.4	32.4	32.5	32.5	32.3	32.3	32.4	32.5	30.4	30.2	30.3	30.2
	R	53.3	53.2	53.4	53.4	52.8	52.7	52.9	53.0	50.3	49.9	50.1	49.8
$\boldsymbol{f}_{12}$	P	52.6	53.9	53.3	53.9	**54.4**	**53.0**	**52.8**	**53.9**	53.0	52.6	53.1	53.7
	R	89.0	90.4	89.4	90.1	**90.9**	**89.2**	**88.8**	**90.0**	85.8	85.7	86.8	87.8
$\boldsymbol{f}_{13}$	P	48.0	49.8	46.8	44.1	46.6	44.9	42.9	41.0	46.7	44.4	41.4	40.3
	R	79.3	81.7	78.2	74.7	77.5	74.9	72.4	70.1	77.0	73.4	40.3	67.8
$\boldsymbol{f}_{14}$	P	33.1	32.2	33.8	31.7	31.4	30.8	32.4	30.7	32.7	32.9	30.8	32.3
	R	54.2	52.7	55.0	52.0	50.9	49.9	52.5	49.9	53.6	53.9	50.7	53.2
$\boldsymbol{f}_{15}$	P	38.6	38.3	38.3	38.1	32.9	32.6	32.7	32.3	34.4	34.5	34.6	34.6
	R	62.8	62.4	62.4	61.7	53.7	53.1	53.0	52.6	57.2	57.4	57.4	57.4
$\boldsymbol{f}_{16}$	P	22.4	21.8	19.3	19.5	22.3	21.8	19.4	19.5	22.3	21.7	19.3	19.5
	R	38.1	37.7	37.1	37.0	38.0	37.6	37.1	37.0	38.0	37.6	37.1	37.0
$\boldsymbol{f}_{17}$	P	23.1	21.3	22.5	22.3	27.3	24.8	25.5	26.6	30.9	28.4	29.0	29.7
	R	43.4	38.9	40.8	38.8	52.2	50.0	48.9	49.4	57.0	51.6	53.2	54.9

Table 11: Open curve matching results with Signature-based CS descriptors using point matching methods on ETHZ CS Dataset. CS descriptor Tangent ($\boldsymbol{f}_{12}$) with dynamic programming matching method achieves the best performance (marked with bold).

matching methods perform better than the vector distance methods. Specifically, Tangent ($\boldsymbol{f}_{12}$) with dynamic programming and Comcoor ($\boldsymbol{f}_{10}$) with χ^2-Statistics achieve the best performance in two strategies, respectively. However, considering their best recall and precision scores, $\boldsymbol{f}_{12}$ and $\boldsymbol{f}_{10}$ are very close to each other though $\boldsymbol{f}_{12}$ with dynamic programming is slightly better.

For the rich CS descriptors, similar to signature-based descriptors, matching algorithms have big influences on recall and precision scores (Table 13 and Table 14). Moreover, for most descriptors, the more sample points we select, the less recall and precision we get. This is mainly because for those CS descriptors, more fine-granted features will be lost if we select less sample points. Thus, the global similarities between CSs become more similar. Compared to other CS rich descriptors, Point Triangle ($\boldsymbol{f}_{19}$), Contour Context ($\boldsymbol{f}_{20}$) and Length Direction ($\boldsymbol{f}_{25}$) are stable for CS

ETHZ CS Dataset - Signature-based CS Descriptors (Vector Distance Methods, %)																	
Descriptors		Correlation				HI				χ^2-Statistics				Hellinger			
		L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{10}$	P	49.0	49.6	49.5	49.7	29.7	27.8	28.5	27.0	**53.3**	**53.3**	**52.5**	**52.4**	49.4	49.4	49.2	48.8
	R	83.6	84.3	84.2	84.4	48.9	45.6	46.8	44.9	**90.4**	**90.4**	**89.5**	**89.4**	84.3	84.1	84.0	83.6
$\boldsymbol{f}_{11}$	P	5.8	6.0	6.0	6.0	17.0	17.2	17.1	17.1	35.1	34.9	35.0	35.0	49.4	49.4	49.2	48.8
	R	9.9	10.2	10.2	10.3	27.9	28.2	28.0	28.1	57.6	57.3	57.5	57.5	84.3	84.2	84.0	83.5
$\boldsymbol{f}_{12}$	P	12.7	15.1	14.1	14.0	4.5	5.7	5.9	6.2	51.4	49.9	49.8	49.8	43.4	39.0	38.9	38.0
	R	17.2	20.5	19.3	19.3	6.6	8.3	8.6	9.1	84.4	82.8	82.7	82.9	74.8	69.3	69.1	67.8
$\boldsymbol{f}_{13}$	P	4.7	5.1	6.7	7.9	45.4	29.2	11.5	9.7	47.4	42.0	34.1	30.2	53.2	55.2	52.1	50.0
	R	8.2	9.2	11.5	13.0	76.7	52.5	23.3	20.3	77.4	67.9	56.4	50.7	87.8	88.8	84.3	81.2
$\boldsymbol{f}_{14}$	P	4.3	4.6	3.6	5.4	14.0	14.0	14.1	14.3	38.3	38.1	37.8	37.5	49.1	47.3	47.2	47.2
	R	7.7	7.8	5.6	9.5	23.0	23.1	23.2	23.4	62.6	62.4	62.0	61.6	82.9	81.2	81.0	80.6
$\boldsymbol{f}_{15}$	P	9.0	9.3	9.0	8.4	27.5	26.8	27.1	26.6	40.2	39.3	39.3	39.2	43.5	43.9	43.9	44.8
	R	15.3	15.7	15.3	14.3	47.6	46.7	47.0	45.9	66.6	65.4	65.4	65.0	73.2	73.7	73.7	74.9
$\boldsymbol{f}_{16}$	P	22.2	21.4	19.3	19.6	18.5	18.5	18.6	18.6	22.0	21.8	19.3	19.5	18.0	18.5	18.7	18.7
	R	37.8	37.0	37.1	37.1	36.7	36.6	36.9	36.9	37.2	37.3	37.0	37.0	35.4	36.1	36.9	36.9
$\boldsymbol{f}_{17}$	P	33.7	26.0	25.8	29.1	34.6	28.6	29.7	29.9	19.5	15.8	16.0	21.1	33.3	26.5	24.4	24.3
	R	59.7	47.0	48.0	54.7	65.6	55.6	57.7	57.9	36.0	27.2	28.4	39.6	63.0	52.1	47.8	47.0

Table 12: Open curve matching results with Signature-based CS descriptors using vector distance methods on ETHZ CS Dataset. CS descriptor Comcoor ($\boldsymbol{f}_{10}$) with χ^2-Statistics vector distance method achieves the best performance (marked with bold).

length changing and also have good matching performance (more that 90% precision). Among all these descriptors, Point Triangle ($\boldsymbol{f}_{19}$) with dynamic programming achieves the best performance (95.4% recall). For Line Segment ($\boldsymbol{f}_{26}$) and Sub Box ($\boldsymbol{f}_{27}$) descriptors (Table 14) which are using the vector distance methods, both of their matching performances are not as good as Point Triangle ($\boldsymbol{f}_{19}$) with dynamic programming.

Comparing the performance of open curves matching using simple, signature-based and rich CS descriptors in this dataset, we can draw the following observations: (1) Most of rich CS descriptors outperform signature-based CS descriptors. (2) Most signature-based CS descriptors performs better than the simple CS descriptors. (3) For most signature-based CS descriptors, point matching strategy performs better than the vector distance strategy for open curve matching. (4) For open curve matching on ETHZ dataset, Point Triangle ($\boldsymbol{f}_{19}$) with dynamic programming achieves the best per-

ETHZ CS Dataset - Rich CS Descriptors ($\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{25}$, %)													
Descriptors		DTW [31]				DP [30]				Hungarian [29]			
		L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{18}$	P	40.8	40.3	40.0	39.3	46.0	46.5	46.5	46.3	46.8	47.2	47.2	47.2
	R	77.6	76.8	76.0	75.1	87.5	88.6	88.5	88.1	88.3	89.0	89.2	89.0
$\boldsymbol{f}_{19}$	P	43.0	41.1	41.0	41.0	**51.2**	**50.4**	**50.2**	**50.0**	51.0	50.1	50.0	50.0
	R	83.5	81.0	80.2	80.0	**95.4**	**94.2**	**94.0**	**94.0**	94.5	94.0	93.6	94.0
$\boldsymbol{f}_{20}$	P	41.5	41.5	41.5	41.5	50.0	50.0	49.5	49.5	49.0	49.0	49.0	49.0
	R	79.0	78.6	78.5	78.5	94.0	94.0	94.1	94.1	93.0	93.0	93.0	93.0
$\boldsymbol{f}_{21}$	P	45.3	41.2	37.3	35.3	41.0	37.0	34.0	32.4	43.5	40.1	37.3	35.3
	R	86.2	79.0	72.0	67.8	78.5	71.1	65.5	62.3	83.0	76.4	71.3	67.1
$\boldsymbol{f}_{22}$	P	41.1	40.2	40.0	39.6	46.1	46.3	46.2	46.1	44.6	43.6	43.1	42.9
	R	76.6	75.3	74.6	74.2	87.4	88.1	87.9	87.8	83.9	82.3	81.5	81.3
$\boldsymbol{f}_{23}$	P	48.8	48.7	48.7	48.8	30.1	28.4	27.8	27.7	30.8	29.0	28.5	28.3
	R	90.1	90.1	90.1	90.2	59.5	57.1	56.5	56.5	60.8	58.3	57.5	57.5
$\boldsymbol{f}_{24}$	P	41.5	40.5	40.0	39.7	46.7	46.9	47.0	47.0	46.4	46.7	46.8	46.9
	R	77.3	75.8	75.1	74.4	88.6	89.1	89.3	89.4	87.7	88.1	88.3	88.5
$\boldsymbol{f}_{25}$	P	40.6	40.6	40.6	40.6	48.7	48.7	48.7	48.7	40.4	40.4	40.5	40.4
	R	76.5	76.6	76.6	76.4	90.9	91.0	91.0	91.0	76.3	76.3	76.3	76.2

Table 13: Open curve matching results with Rich CS descriptors ($\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{25}$) on ETHZ CS Dataset. Point Triangle ($\boldsymbol{f}_{19}$) with dynamic programming achieves the best performance (marked with bold).

ETHZ CS Dataset - Rich CS Descriptors ($\boldsymbol{f}_{26}$ and $\boldsymbol{f}_{27}$, %)																	
Descriptors		Correlation				HI				χ^2-Statistics				Hellinger			
		L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
$\boldsymbol{f}_{26}$	P	39.3	39.0	39.2	39.1	33.2	33.2	33.4	33.2	29.2	29.2	29.3	29.2	43.7	43.5	43.7	43.5
	R	73.9	73.8	74.3	74.2	61.9	61.8	62.3	61.9	55.2	55.2	55.4	55.3	80.7	80.3	80.8	80.4
$\boldsymbol{f}_{27}$	P	5.6	3.7	3.3	3.4	2.6	2.2	2.2	2.3	51.6	53.7	54.1	53.7	52.3	53.5	53.6	53.4
	R	10.0	7.6	6.9	6.9	5.2	4.7	4.6	4.8	83.1	85.9	86.3	85.6	84.1	85.6	85.9	85.6

Table 14: Open curve matching results with Rich CS descriptors ($\boldsymbol{f}_{26}$ and $\boldsymbol{f}_{27}$) on ETHZ CS Dataset.

formance (95.4% recall). (5) Considering CS descriptors with the best performance in Table 5- 14, we can observe that for the most CS descriptors, if one has good performance for CS matching in the MPEG7 CS dataset, it also achieves promising results for open curve matching in the ETHZ CS dataset.

5.2.3. Evaluation on Sketching CS dataset

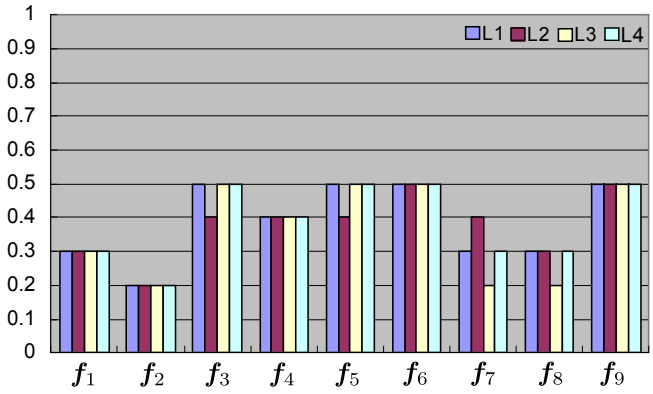
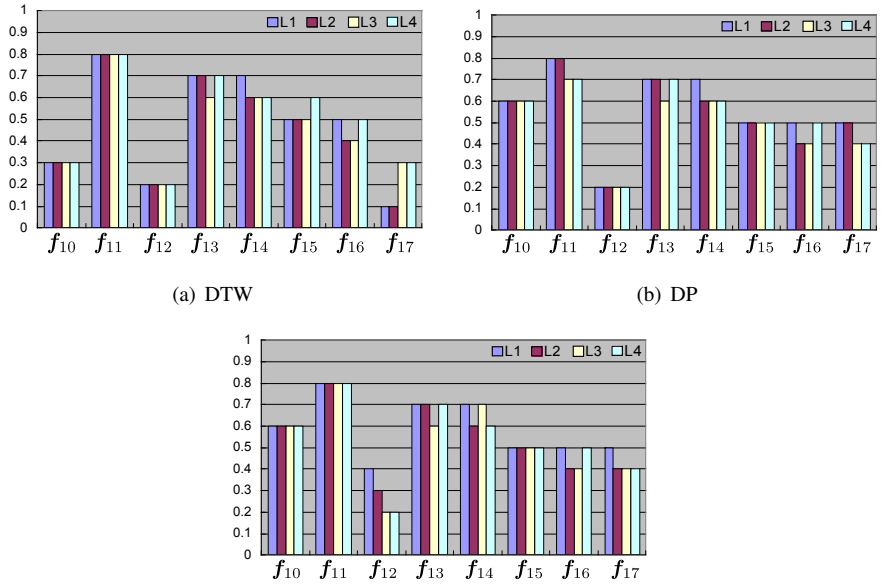


Figure 16: Similarity between simple CS descriptors and human perception on the Sketching CS dataset.

In this part, we employ the bar chart to intuitively evaluate and show the similarity between CS descriptors and human perception. Figure 16 illustrates the similarity comparison between simple descriptors with different CS lengths. It can be noted that simple descriptors are relatively robust to length changing. Moreover, Eccentricity ($\boldsymbol{f}_3$), Rectangularity ($\boldsymbol{f}_5$), LineRatio ($\boldsymbol{f}_6$) and Dislength ($\boldsymbol{f}_9$) have the same and highest similarity value (50% similarity) to human perception.

Compared to simple descriptors, Figure 17 and 18 show that most signature-based descriptors are not robust to length changing on the Sketching CS dataset. More specifically, the more sample points we select, the less similarity values we get. The main reason is that, similar to the signature-based descriptors, more fine-granted features will be lost if we select less sample points. Among all signature-based descriptors, Cendistance ($\boldsymbol{f}_{11}$) achieves the same and the highest similarity value (80%) in all point matching algorithms (Dynamic Time Warping [31], Dynamic Programming [30] and Hungarian [29]) and the χ^2-Statistics vector distance method.

As illustrated in Figure 19 and 20, most rich descriptors are robust to the CS length changing. Moreover, most of them have the same similarity values as simple and signature-based descriptors in which the Line Segment ($\boldsymbol{f}_{26}$) achieves the same and highest similarity value (80%) in both Histogram Intersection and χ^2-Statistics vector distance methods.



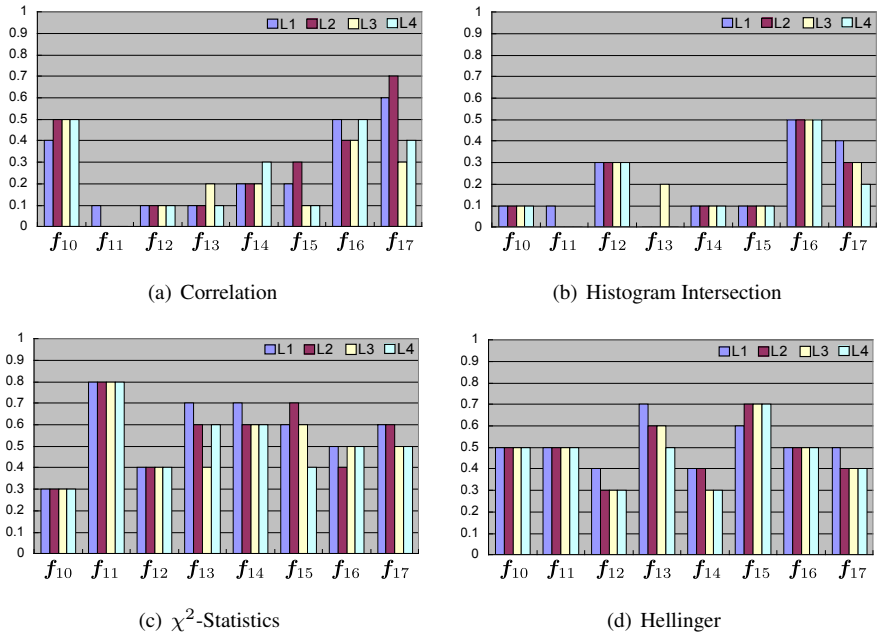
(a) DTW

(b) DP

(c) Hungarian

Figure 17: Similarity between signature-based CS descriptors (point matching methods) and human perception on the Sketching CS dataset.

The comparison in the above figures indicates that differences between simple, signature-based and rich CS descriptors are not so obvious as in the ETHZ CS dataset. However, in each type of CS descriptor, the distribution of similarity values is varying. The performance in cases of L3 and L4 is occasionally worse than the performance in cases of L1 and L2, one possible reason is the overfitting problem. Among three types of CS descriptors, Cendistance ($\boldsymbol{f}_{11}$) and Line Segment ($\boldsymbol{f}_{26}$) achieve the same and highest score (80%) using several point matching and the vector distance methods. Figure 21 illustrates the comparison of all similarity values among three types of CS descriptors. As the horizontal axis represents each value's ID which only used for visualisation, we focus on the vertical axis by analysing the similarity values. Considering the value distribution, signature-based CS descriptors have the biggest interval which ranged from 2.5% to 80%, followed by the rich CS descriptors which are in the range of $[10\%, 80\%]$. Simple CS descriptors have the smallest interval $[20\%, 50\%]$. One reason is that for each signature-based and rich CS descriptors, there are many algorithms



(a) Correlation

(b) Histogram Intersection

(c) χ^2-Statistics

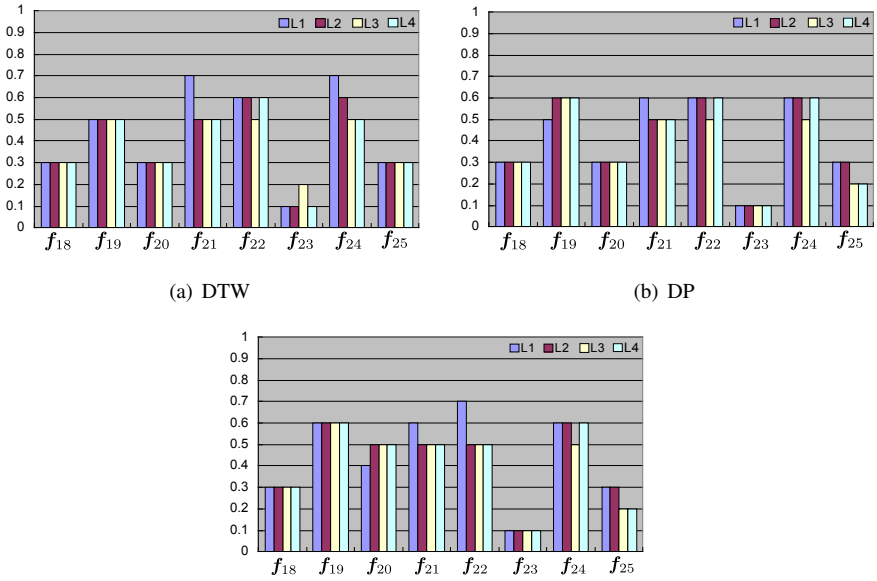
(d) Hellinger

Figure 18: Similarity between signature-based CS descriptors (vector distance methods) and human perception on the Sketching CS dataset.

that can be employed for calculating similarities between CSs. On the contrary, simple CS descriptors only have one. Therefore, rich and signature-based CS descriptors have more flexibility for handling different scenarios. Considering the mean similarity values among three types of descriptors, rich CS descriptors (52.68%) is closer to the human perception than signature-based (47.68%) while the simple CS descriptors (40%) have the lowest one. Overall, there is still a gap between the human perception and CS descriptors (mean value 46.79%) though only two descriptors obtain the same retrieval results (80%) as human perception.

5.3. Experiment 3: Computation Complexity

In this subsection, we first analyse the computational complexity theoretically. After that, we collect and compare the runtime for feature generation and matching of each CS descriptors. All the runtimes are collected by applying the full CS retrieval on MPEG7 CS dataset using full CS length (L4). For a fair comparison, all the runtimes



(a) DTW

(b) DP

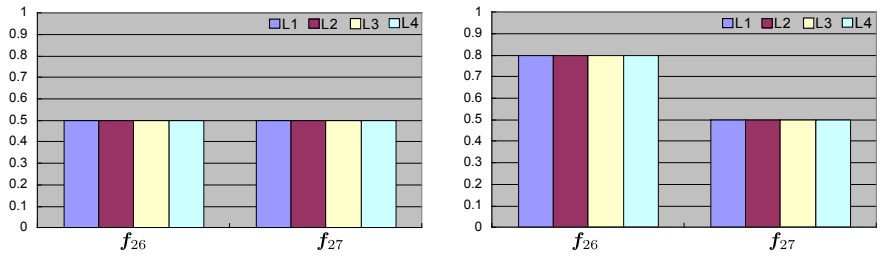
(c) Hungarian

Figure 19: Similarity between rich CS descriptors (point matching methods) and human perception on the Sketching CS dataset.

are collected on the same laptop computer as described in Section 4.3.

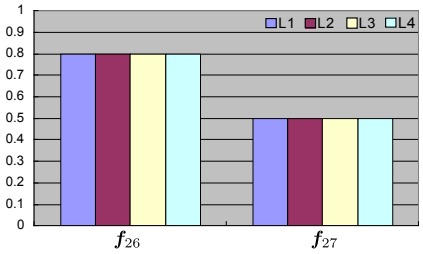
Table 15 illustrates the theoretical analysis of computational complexity for each CS descriptor. All simple CS descriptors have the same feature generation complexity $O(N)$ since they count the CS point numbers to calculate the CS length, area and the bounding box, etc. N denotes the number of CS points. Since CS descriptors are just scalar values, their distance can be computed by scalar subtraction, so its computation cost is $O(1)$.

For signature-based descriptors, except Triangle Area ($\boldsymbol{f}_{15}$) and Chord Length ($\boldsymbol{f}_{16}$), most descriptors take $O(N)$ complexity for feature generation since each element is calculated only using one CS point. $\boldsymbol{f}_{15}$ and $\boldsymbol{f}_{16}$ are both calculated by considering one target CS point and other reference points selected by searching the whole CS path. Therefore, their computation complexity is $O(N) * O(N) = O(N^2)$. For CS retrieval, two strategies are employed and compared independently to calculate the similarity between two signature-based CSs. With the vector distance strategy, all four

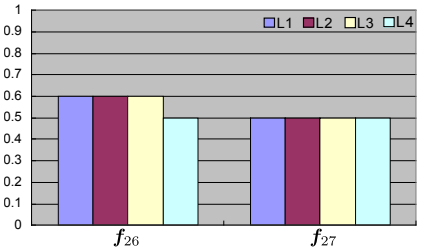


(a) Correlation

(b) Histogram Intersection



(c) χ^2-Statistics



(d) Hellinger

Figure 20: Similarity between rich CS descriptors (vector distance methods) and human perception on the Sketching CS dataset.

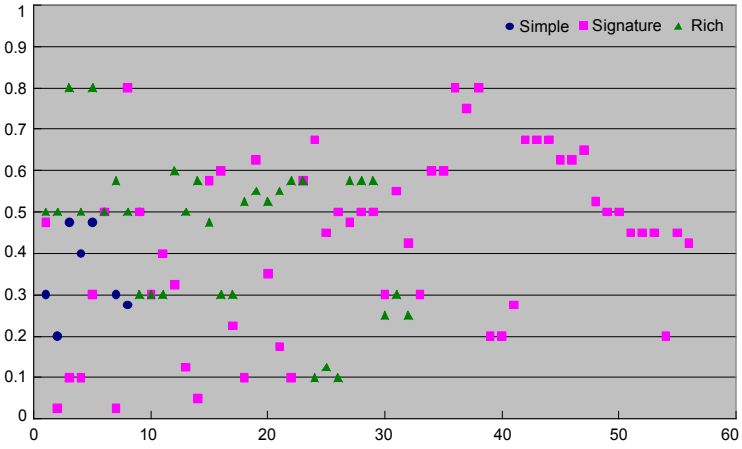


Figure 21: Comparison of similarity values between three types of CS descriptors. Each mark represents a similarity value from Figure 16 to 20, plotted by mark's ID on the horizontal axis and values on the vertical. Each value in this figure is calculated by the mean similarity value of four different lengths.

Simple CS Descriptors					
Name	Feature Generation	Difference Value			
$\boldsymbol{f}_1$ - $\boldsymbol{f}_9$	$O(N)$	$O(1)$			
Signature-Based CS Descriptors					
Name	Feature Generation	Matching (Point Matching)			Matching (Vector Distance)
		DTW [31]	DP [30]	Hungarian [29]	
$\boldsymbol{f}_{10}$ - $\boldsymbol{f}_{14}$, $\boldsymbol{f}_{17}$	$O(N)$	$O(N^2)$	$O(N^2)$	$O(N^3)$	$O(N)$
$\boldsymbol{f}_{15}$, $\boldsymbol{f}_{16}$	$O(N^2)$	$O(N^2)$	$O(N^2)$	$O(N^3)$	$O(N)$
Rich CS Descriptors					
Name	Feature Generation	Matching (Point Matching)			Matching (Vector Distance)
		DTW [31]	DP [30]	Hungarian [29]	
$\boldsymbol{f}_{18}$ - $\boldsymbol{f}_{20}$ $\boldsymbol{f}_{22}$ - $\boldsymbol{f}_{25}$	$O(N^2)$	$O(N^2)$	$O(N^2)$	$O(N^3)$	NULL
$\boldsymbol{f}_{21}$	$O(N)$	$O(N^2)$	$O(N^2)$	$O(N^3)$	NULL
$\boldsymbol{f}_{26}$	$O(N)$	NULL	NULL	NULL	$O(1)$
$\boldsymbol{f}_{27}$	$O(N^2)$	NULL	NULL	NULL	$O(1)$

Table 15: Theoretical analysis of computation complexity for each CS descriptor in terms of feature generation and matching. NULL means this part is not considered in our experiments.

distance methods have the same computation complexity $O(N)$.

With the point matching strategy, three matching algorithms have different computation complexity. More specifically, (i) the computational complexity of DTW is $O(M * N)$ where M and N are lengths of two time series, respectively. In our case, both time series have the same length N, so the time complexity of DTW is $O(N^2)$. (ii) For solving the sequence alignment problem using DP, as the full dynamic programming table is pre-constructed (Section 3.2), the time complexity for CS matching is reduced from $O(N^3)$ to $O(N^2)$ by observing that at any instant, the algorithm only requires two rows (or two columns) in memory. (iii) Hungarian algorithm is normally used as a classical solution to the assignment problem. As introduced in [74], it can solve our CS matching task in $O(N^3)$ time.

For rich CS descriptors, we can observe that most descriptors require (or more than) $O(N^2)$ complexity for feature generation. In contrast, Beam Angle ($\boldsymbol{f}_{21}$) is calculated

by the angles between CS points and its neighbouring points which can be captured directly. Line Segment ($\boldsymbol{f}_{26}$) is calculated by the statistics of a fixed range of straight-line scales n (n is set from 5% to 85% in all experiments). Since the n is a value independent of CS point number N and $n << N$, the computational complexity of this method is determined by the number of CS points. Therefore, the complexity of $\boldsymbol{f}_{21}$ and $\boldsymbol{f}_{26}$ are $O(N)$. For CS matching, according to the feature construction of different descriptors, the point matching strategy is applied to descriptors ranging from $\boldsymbol{f}_{18}$ to $\boldsymbol{f}_{25}$, in which the point matching algorithms have the same complexity as signature-based descriptors. However, with the vector distance strategy, Line Segment ($\boldsymbol{f}_{26}$) and Sub Box ($\boldsymbol{f}_{27}$) have the complexity $O(n)$ where n is the feature dimension. Since their feature dimensions are fixed and the distance between two vectors can be calculated in a constant time, their computational complexity is $O(1)$.

Computation Complexity - Runtime (Simple CS Descriptors; Hours)					
Name	Feature Generation	Matching	Name	Feature Generation	Matching
$\boldsymbol{f}_1$	0.00191	0.0086	$\boldsymbol{f}_6$	0.00004	0.0081
$\boldsymbol{f}_2$	0.00199	0.0085	$\boldsymbol{f}_7$	0.00007	0.0081
$\boldsymbol{f}_3$	0.00025	0.0087	$\boldsymbol{f}_8$	0.00219	0.0083
$\boldsymbol{f}_4$	0.00035	0.0087	$\boldsymbol{f}_9$	0.00004	0.0084
$\boldsymbol{f}_5$	0.00024	0.0085			

Table 16: Runtime comparison of simple CS descriptors.

Table 16, 17 and 18 clarify runtimes for full CS retrieval on the MPEG7 CS dataset using different CS descriptors. We can clearly observe that the matching strategy of the vector distance method takes much less time than the point matching strategy in both signature-based and rich descriptors. Moreover, observing the runtime among four vector distance methods, there is no obvious difference between them. However, considering the point matching strategy, the Hungarian [29] method takes the highest runtime followed by Dynamic Time Warping [31]. Dynamic Programming [30] takes the shortest runtime among the three point matching methods. For feature generation, Point Triangle ($\boldsymbol{f}_{19}$) takes the longest time compared to other descriptors. This is also correlated to the theoretical analysis of computation complexity in Table 15, in which

Computation Complexity - Runtime (Signature-based CS Descriptors; Hours)								
Descriptors	Feature Generation	Matching (Point Matching)			Matching (Vector Distance)			
		DTW [31]	DP [30]	Hungarian [29]	Correlation	HI	χ^2-Statistics	Hellinger
$\boldsymbol{f}_{10}$	0.0001	42.81	3.95	213.74	0.1102	0.0506	0.0542	0.0531
$\boldsymbol{f}_{11}$	0.0001	34.86	4.37	157.76	0.1042	0.0404	0.0460	0.0462
$\boldsymbol{f}_{12}$	0.0001	33.68	3.86	126.22	0.1032	0.0386	0.0441	0.0445
$\boldsymbol{f}_{13}$	0.0002	26.83	5.77	197.31	0.1044	0.0400	0.0454	0.0456
$\boldsymbol{f}_{14}$	0.0034	37.71	5.25	190.71	0.1043	0.0401	0.0452	0.0457
$\boldsymbol{f}_{15}$	0.0001	13.16	1.61	89.33	0.0985	0.0350	0.0392	0.0412
$\boldsymbol{f}_{16}$	0.0449	31.24	2.76	16.20	0.0772	0.0217	0.0236	0.0235
$\boldsymbol{f}_{17}$	0.0021	0.01	4.93	72.18	0.12	0.05	0.05	0.05

Table 17: Runtime comparison of signature-based CS descriptors.

Computation Complexity - Runtime (Rich CS Descriptors; Hours)								
Descriptors	Feature Generation	Matching (Point Matching)			Matching (Vector Distance)			
		DTW [31]	DP [30]	Hungarian [29]	Correlation	HI	χ^2-Statistics	Hellinger
$\boldsymbol{f}_{18}$	0.02	1.6	4.7	95.7	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{19}$	6.82	102.7	121.8	148.9	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{20}$	0.04	21.1	6.7	89.7	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{21}$	0.01	21.9	3.6	147.1	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{22}$	0.18	23.2	5.3	145.6	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{23}$	0.17	22.1	5.6	214.2	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{24}$	0.18	24.5	5.4	95.7	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{25}$	0.02	224.0	212.1	276.6	NULL	NULL	NULL	NULL
$\boldsymbol{f}_{26}$	0.005	NULL	NULL	NULL	0.016	0.009	0.010	0.014
$\boldsymbol{f}_{27}$	0.018	NULL	NULL	NULL	0.013	0.013	0.012	0.014

Table 18: Runtime comparison of Rich CS descriptors. NULL means this part is not considered in our experiments.

760 $\boldsymbol{f}_{19}$ has the highest complexity $O(N^3)$. Please notice that our code is not optimised, and its faster implementation is possible by optimising loops, settings and programming language, etc. Thus, there are still plenty of opportunities to reduce the running time.

5.4. Discussion

Name	Notation	Type	Matching Method	FG Time	Matching Time	Results* ETHZ CS	Results* MPEG7 CS
Comcoor	$\boldsymbol{f}_{10}$	Signature	χ^2-Statistics	0.0001	0.0542	P: 52.9, R: 89.9	56.2
Comcoor	$\boldsymbol{f}_{10}$	Signature	Hungarian	0.0001	213.74	P: 48.9, R: 81.3	64.0
Comcoor	$\boldsymbol{f}_{10}$	Signature	Hellinger	0.0001	0.0531	P: 49.2, R: 84.0	63.6
Cendistance	$\boldsymbol{f}_{11}$	Signature	Hellinger	0.0001	0.0462	P: 49.2, R: 84.0	63.6
Tangent	$\boldsymbol{f}_{12}$	Signature	DP	0.0001	3.86	P: 53.5, R: 89.7	57.8
Point Triangle	$\boldsymbol{f}_{19}$	Rich	DP	6.82	121.8	P: 50.5, R: 94.4	69.7
Contour Context	$\boldsymbol{f}_{20}$	Rich	DP	0.04	6.7	P: 49.6, R: 94.1	63.7
Beam Angle	$\boldsymbol{f}_{21}$	Rich	Hungarian	0.01	147.1	P: 39.1, R: 74.5	77.6
Partial Contour	$\boldsymbol{f}_{22}$	Rich	DP	0.18	5.3	P: 46.2, R: 87.8	74.1
Opt Partial Contour	$\boldsymbol{f}_{23}$	Rich	DTW	0.17	22.1	P: 48.8, R: 90.1	60.0
Chord Distribution	$\boldsymbol{f}_{24}$	Rich	DP	0.18	5.4	P: 46.9, R: 89.1	74.6
Length Direction	$\boldsymbol{f}_{25}$	Rich	DP	0.02	212.1	P: 48.7, R: 91.0	64.3
* As there are four values for four different lengths, we take the mean value for comparison. P: Precision, R: Recall, D-Value: Difference Value, FG: Feature Generation							

Table 19: Matching performance and time (hour) comparison between selected CS descriptors which have outstanding matching performance in ETHZ and MPEG7 CS datasets.

In the previous subsections, the theoretical invariance properties and computation complexity of CS descriptors were verified experimentally. However, possessing those properties independently is not sufficient to address open curve matching: A desirable CS descriptor should have both characteristics of good matching performance and relatively low computation complexity. Therefore, in this part, we will conduct the comparison of CS descriptors by taking both matching performance and computation complexity into account. In order to simplify the content, we only consider the descriptors which have outstanding matching performance on ETHZ and MPEG7 CS dataset.

Table 19 illustrate the comparison between the selected descriptors which have outstanding matching performance in ETHZ and MPEG7 CS datasets. We can observe that the selected signature-based and rich descriptors have robust performances in which Beam Angle ($\boldsymbol{f}_{21}$) with Hungarian [29] achieves the best bulls-eye score (77.6%). Tangent ($\boldsymbol{f}_{12}$) and Point Triangle ($\boldsymbol{f}_{19}$) with Dynamic Programming [30] obtain the best precision (53.5%) and recall (94.4%). However, considering the runtime

and retrieval performance of Partial Contour ($\boldsymbol{f}_{22}$) and Chord Distribution ($\boldsymbol{f}_{24}$) are close to the best on both datasets while taking up a lot less time for feature generation and matching.

Based on the observations above, we can draw the following recommendations: (i) When choosing a CS descriptor for open curve matching with time complexity not being of primary importance, the best choice is Point Triangle ($\boldsymbol{f}_{19}$) with Dynamic Programming [30] since it is scale, rotation and translation invariant (Table 4) and robust to CS length changing (Table 13 and Table 8). Moreover, it achieves promising results on both ETHZ and MPEG7 CS datasets (Table 19). (ii) To obtain a stable and promising performance while using less time for feature generation and matching, Partial Contour ($\boldsymbol{f}_{22}$) and Chord Distribution ($\boldsymbol{f}_{24}$) with Dynamic Programming [30] are the best choices since compared to Point Triangle ($\boldsymbol{f}_{19}$), both descriptors achieve close and promising results on two datasets but only use 2.6% time for feature generation and 4.4% time for CS retrieval (Table 19). (iii) If we want to apply a fast open curve matching and obtain a relatively promising results, Cendistance ($\boldsymbol{f}_{11}$) with Hellinger [35] vector distance method is the best choice, for which the fast runtime for feature generation and matching can be ensured while the matching performance on both datasets is not the best but still promising.

To obtain a state-of-the-art performance for open curve matching on a real-world dataset, multiple CS descriptors should be chosen and fused [75]. As discussed in [9], even a simple combination of shape descriptors improves the overall performance of individual descriptors. According to the introduction in Section 2.1, simple CS descriptors are not suitable to be standalones for open curve matching. According to the matching performance on ETHZ and MPEG7 CS datasets, we fuse Eccentricity (f_3) and Rectangularity (f_5) with other signature-based and rich CS descriptors. To compute proper fusing weights, we first divide the dataset into two equal parts, one being used for weight estimation and the other for testing. For fusing weight estimation, we employ a supervised optimisation scheme [76] in which two heuristic approaches are combined: Gradient Hill Climbing [77] integrated with Simulated Annealing [78].

The Gradient Hill Climbing method starts with randomly selected parameters. Then, it changes single parameters iteratively to form a new set of parameters. A fitness func-

Descriptors	Matching Algorithm	ETHZ CS Dataset		MPEG7 CS Dataset
		Precision	Recall	
Eccentricity ($\boldsymbol{f}_3$)	D-Value	55.1	90.1	22.7
Point Triangle ($\boldsymbol{f}_{19}$)	DP	50.5	94.4	69.7
Fused	DP + D-Value	56.1	97.6	74.6

Table 20: Retrieval Results on two datasets using the fused descriptors.

tion then evaluates whether the new set of parameters performs better or worse. In our case, the fitness function could be the recall, precision or the bull-eye score, depending on different scenarios. The Simulated Annealing strategy impacts the degree of the changes. In later iterations, the changes to the parameters are becoming smaller. This strategy can efficiently reduce the computation complexity of our optimisation method. With this method, we experimentally assess the matching performance by fusing the Point Triangle ($\boldsymbol{f}_{19}$) and Eccentricity ($\boldsymbol{f}_3$) descriptors. Our experiments shows that, comparing to Point Triangle ($\boldsymbol{f}_{19}$), the fused descriptor improves the matching accuracy by 4.4% on the ETHZ CS dataset and by 4.9% on the MPEG7 CS dataset (Table 20).

6. Conclusion and Future Work

In this paper, we made a comprehensive comparison of 27 CS descriptors by correlating them with distance and matching methods for CS matching. We also studied and evaluated the invariant properties, matching performance and computation complexity of CS descriptor. In order to cover various configurations for CS matching, the evaluation is carried out with respect to different matching algorithms and CS lengths, see Table 2 for an overview. From the theoretical and experimental results, it can be

Best Accuracy	Promissing Accuracy and Less Runtime	Fast Speed and Relatively Promissing Accuracy
Point Triangle ($\boldsymbol{f}_{19}$) + DP	Partial Contour ($\boldsymbol{f}_{22}$) + DP	Cendistance ($\boldsymbol{f}_{11}$) + Hellinger
	Chord Distribution ($\boldsymbol{f}_{24}$) + DP	

Table 21: Recommended choice of descriptors for different requirements.

derived that the selection of the CS length and matching algorithm affects the matching performance while matching algorithms have a higher significant affection. The results further reveal that signature-based and rich descriptors with proper matching algorithms can fulfil different requirements in terms of speed and accuracy for CS matching. The overall recommendations for choosing CS descriptors and their settings are illustrated in Table 21. In addition, a proper combination of rich and simple CS descriptors can improve the matching accuracy over the individual descriptors without adding too much computational complexity.

Acknowledgements: Research activities leading to this work have been supported by the China Scholarship Council (CSC) and the German Research Foundation (DFG) within the Research Training Group 1564 (GRK 1564).

References

[1] B. Hariharan, P. Arbelaez, R. Girshick, J. Malik, Hypercolumns for object segmentation and fine-grained localization, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 447–456.

[2] F. Wang, L. Kang, Y. Li, Sketch-based 3d shape retrieval using convolutional neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1875–1883.

[3] I. Biederman, G. Ju, Surface versus edge-based determinants of visual recognition, Cognitive Psychology 20 (1) (1988) 38–64.

[4] J. De Winter, J. Wagemans, Contour-based object identification and segmentation: Stimuli, norms and data, and software tools, Behavior Research Methods, Instruments and Computers 36 (4) (2004) 604–624.

[5] J. Shotton, A. Blake, R. Cipolla, Multiscale categorical object recognition using contour fragments, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (7) (2008) 1270–1281.

[6] N. Payet, S. Todorovic, From a set of shapes to object discovery, in: European Conference on Computer Vision, 2010, pp. 57–70.

[7] H. Riemenschneider, M. Donoser, H. Bischof, Using partial edge contour matches for efficient object category localization, in: European Conference on Computer Vision, 2010, pp. 29–42.

[8] M. D. Peter Kontschieder, Hayko Riemenschneider, H. Bischof, Discriminative learning of contour fragments for object detection, in: British Machine Vision Conference, 2011, pp. 1–12.

[9] C. Yang, O. Tiebe, P. Pietsch, C. Feinen, U. Kelter, M. Grzegorzek, Shape-based object retrieval by contour segment matching, in: IEEE International Conference on Image Processing, 2014, pp. 2202–2206.

[10] Y. Liu, J. Gall, C. Stoll, Q. Dai, H.-P. Seidel, C. Theobalt, Markerless motion capture of multiple characters using multiview image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (11) (2013) 2720–2735.

[11] G. Bertasius, J. Shi, L. Torresani, Deepedge: A multi-scale bifurcated deep network for top-down contour detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 4380–4389.

[12] W. Shen, X. Wang, Y. Wang, X. Bai, Z. Zhang, Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3982–3991.

[13] H. Zhang, Y. Liu, B. Xie, J. Yu, Orientation contrast model for boundary detection, Journal of Visual Communication and Image Representation 25 (5) (2014) 774–784.

[14] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 8 (6) (1986) 679–698.

[15] R. M. Haralick, L. G. Shapiro, Computer and Robot Vision, 1st Edition, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.

[16] X. Zhang, M. Lei, D. Yang, Y. Wang, L. Ma, Multi-scale curvature product for robust image corner detection in curvature scale space, Pattern Recognition Letters 28 (5) (2007) 545–554.

[17] M. Awrangjeb, G. Lu, Robust image corner detection based on the chord-to-point distance accumulation technique, IEEE Transactions on Multimedia 10 (6) (2008) 1059–1072.

[18] M. Donoser, H. Riemenschneider, H. Bischof, Linked edges as stable region boundaries, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1665–1672.

[19] J. J. Kivinen, C. K. Williams, N. Heess, Visual boundary prediction: A deep neural prediction network and quality dissection, in: International Conference on Artificial Intelligence and Statistics, 2014, pp. 512–521.

[20] M. Maire, S. Yu, P. Perona, Reconstructive sparse code transfer for contour detection and semantic labeling, in: Asian Conference on Computer Vision, 2014, pp. 273–287.

[21] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, M. Alexa, Sketch-based shape retrieval, ACM Transactions on Graphics 31 (4) (2012) 1–10.

[22] R. Gonzalez, R. Woods, Digital Image Processing, 2nd Edition, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[23] C. Berger, Strokes detection for skeletonisation of characters shapes, in: Advances in Visual Computing, 2014, pp. 510–520.

[24] X. Bai, L. Latecki, W. Liu, Skeleton pruning by contour partitioning with discrete curve evolution, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (3) (2007) 449–462.

[25] P. Soille, bwdistgeodesic uses the geodesic distance algorithm, in: Morphological Image Analysis: Principles and Applications, 2nd Edition, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003, pp. 219–221.

[26] A. Maheshwari, J.-R. Sack, K. Shahbaz, H. Zarrabi-Zadeh, Improved algorithms for partial curve matching, Algorithmica 69 (3) (2014) 641–657.

[27] O. Samanta, U. Bhattacharya, S. K. Parui, Smoothing of hmm parameters for efficient recognition of online handwriting, Pattern Recognition 47 (11) (2014) 3614–3629.

[28] M. Baust, L. Demaret, M. Storath, N. Navab, A. Weinmann, Total variation regularization of shape signals, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2075–2083.

[29] H. W. Kuhn, The hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1955) 83–97.

[30] R. Bellman, The theory of dynamic programming, Bulletin of the American Mathematical Society 60 (6) (1954) 503516.

[31] G. Al-Naymat, S. Chawla, J. Taheri, Sparsedtw: A novel approach to speed up dynamic time warping, in: Australasian Data Mining Conference, Australian Computer Society, Inc., 2009, pp. 117–127.

[32] G. Yule, M. Kendall, An Introduction to the Theory of Statistic, 14th Edition, Charles Griffin Co., 1968.

[33] Y. Rubner, C. Tomasi, L. J. Guibas, The earth mover's distance as a metric for image retrieval, International Journal of Computer Vision 40 (2) (2000) 99–121.

[34] R. L. Plackett, Karl pearson and the chi-squared test, International Statistical Review 51 (1) (1983) 5972.

[35] A. Bhattacharyya, On a measure of divergence between two multinomial populations, The Indian Journal of Statistics 7 (4) (1946) 401–406.

[36] D. Zhang, G. Lu, Review of shape representation and description techniques, Pattern Recognition 37 (1) (2004) 1–19.

[37] T. Ma, L. Latecki, From partial shape matching through local deformation to robust global shape similarity for object detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1441–1448.

[38] M. Donoser, H. Riemenschneider, H. Bischof, Efficient partial shape matching of outer contours, in: Asian Conference on Computer Vision, 2010, pp. 281–292.

[39] J. J. de Mesquita Sa Junior, A. R. Backes, Shape classification using line segment statistics, Information Sciences 305 (2015) 349–356.

[40] R. J. Yang Mingqiang, Kpalma K. Idiyo, A survey of shape feature extraction techniques, in: Pattern Recognition, 2008, pp. 43–90.

[41] M. Peura, J. Iivarinen, Efficiency of simple shape descriptors, in: Third International Workshop on Visual Form, 1997, pp. 443–451.

[42] I. T. Young, J. E. Walker, J. E. Bowie, An analysis technique for biological shape. i, Information and Control 25 (4) (1974) 357–370.

[43] A. Andrew, Another efficient algorithm for convex hulls in two dimensions, Information Processing Letters 9 (5) (1979) 216–219.

[44] P. J. Van Otterloo, A Contour-oriented Approach to Shape Analysis, Prentice Hall International Ltd., Hertfordshire, UK, 1991.

[45] R. Chellappa, R. Bagdazian, Fourier coding of image boundaries, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (1) (1984) 102–105.

[46] K. Arbter, W. E. Snyder, H. Burhardt, G. Hirzinger, Application of affine-invariant fourier descriptors to recognition of 3-d objects, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (7) (1990) 640–647.

[47] A. Krzyzak, S. Leung, C. Suen, Reconstruction of two-dimensional patterns from fourier descriptors, Machine Vision and Applications 2 (3) (1989) 123–140.

[48] H. Kauppinen, T. Seppanen, M. Pietikainen, An experimental comparison of autoregressive and fourier-based descriptors in 2d shape classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (2) (1995) 201–207.

[49] J.-R. Ohm, F. Bunjamin, W. Liebsch, B. Makai, K. Mller, A. Smolic, D. Zier, A set of visual feature descriptors and their combination in a low-level description scheme, Signal Processing: Image Communication 16 (12) (2000) 157–179.

[50] Q. M. Tieng, W. Boles, Recognition of 2d object contours using the wavelet transform zero-crossing representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (8) (1997) 910–916.

[51] H. S. Yang, S. U. Lee, K. M. Lee, Recognition of 2d object contours using starting-point-independent wavelet coefficient matching, Journal of Visual Communication and Image Representation 9 (2) (1998) 171–181.

[52] N. Alajlan, I. E. Rube, M. S. Kamel, G. Freeman, Shape retrieval using triangle-area representation and dynamic space warping, Pattern Recognition 40 (7) (2007) 1911–1920.

[53] J. W. Harris, H. Stocker, Segment of a circle, in: Handbook of Mathematics and Computational Science, New York: Springer-Verlag, 1998, pp. 92–93.

[54] L. Chen, R. Feris, M. Turk, Efficient partial shape matching using smith-waterman algorithm, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2008, pp. 1–6.

[55] J. Wang, X. Bai, X. You, W. Liu, L. J. Latecki, Shape matching and classification using height functions, Pattern Recognition Letters 33 (2) (2012) 134–143.

[56] C. Lu, L. Latecki, N. Adluru, X. Yang, H. Ling, Shape guided contour grouping with particle filters, in: IEEE International Conference on Computer Vision, 2009, pp. 2288–2295.

[57] J. Thureson, S. Carlsson, Appearance based qualitative image description for object class recognition, in: European Conference on Computer Vision, 2004, pp. 518–529.

[58] Q. Zhu, L. Wang, Y. Wu, J. Shi, Contour context selection for object detection: A set-to-set contour matching approach, in: European Conference on Computer Vision, 2008, pp. 774–787.

[59] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 509–522.

[60] H. Riemenschneider, M. Donoser, H. Bischof, Using partial edge contour matches for efficient object category localization., in: European Conference on Computer Vision, 2010, pp. 29–42.

[61] T. Cootes, D. Cooper, C. Taylor, J. Graham, Trainable method of parametric shape description, Image and Vision Computing 10 (5) (1992) 289–294.

[62] T. Ma, L. Latecki, From partial shape matching through local deformation to robust global shape similarity for object detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1441–1448.

[63] S. Salvador, P. Chan, Fastdtw: Toward accurate dynamic time warping in linear time and space, in: KDD Workshop on Mining Temporal and Sequential Data, 2004, pp. 70–80.

[64] P. H. Sellers, The theory and computation of evolutionary distances: Pattern recognition, Journal of Algorithms 1 (4) (1980) 359–373.

[65] P. Karczmarek, A. Kiersztyn, W. Pedrycz, P. Rutka, Chain code-based local descriptor for face recognition, in: International Conference on Computer Recognition Systems, 2015, pp. 10–20.

[66] R. Osada, T. Funkhouser, B. Chazelle, D. Dobkin, Shape distributions, ACM Trans. Graph. 21 (4) (2002) 807–832.

[67] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Conference on Computer Vision and Pattern Recognition, 2001, pp. 511–518.

[68] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, R. Kimmel, Partial similarity of objects, or how to compare a centaur to a horse, International Journal of Computer Vision 84 (2) (2009) 163–183.

[69] L. J. Latecki, R. Lakamper, T. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: IEEE Conference on Computer Vision and Pattern Recognition, 2000, pp. 424–429.

[70] V. Ferrari, T. Tuytelaars, L. V. Gool, Object detection by contour segment networks, in: European Conference on Computer Vision, 2006, pp. 14–28.

[71] T. Fawcett, An introduction to roc analysis, Pattern Recognition Letters 27 (8) (2006) 861–874.

[72] C. Yang, M. Grzegorzek, Object similarity by humans and machines, in: AAAI Fall Symposium on Modeling Changing Perspectives, AAAI Press, 2014.

[73] S. Homer, A. Selman, Introduction to complexity theory, in: Computability and Complexity Theory, Texts in Computer Science, Springer US, 2011, pp. 75–80.

[74] L. Liu, D. Shell, Assessing optimal assignment under uncertainty: An interval-based algorithm, in: Robotics: Science and Systems, 2010.

[75] K. van de Sande, T. Gevers, C. Snoek, Evaluating color descriptors for object and scene recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (9) (2010) 1582–1596.

[76] C. Yang, O. Tiebe, P. Pietsch, C. Feinen, U. Kelter, M. Grzegorzek, Shape-based object retrieval and classification with supervised optimisation, in: International Conference on Pattern Recognition Applications and Methods, 2015, pp. 204–211.

[77] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall Press, 2009.

[78] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science (1983) 671–680.