



HAL
open science

Recognition and interpretation of multi-touch gesture interaction

Zhaoxin Chen

► **To cite this version:**

Zhaoxin Chen. Recognition and interpretation of multi-touch gesture interaction. Human-Computer Interaction [cs.HC]. INSA de Rennes, 2017. English. NNT : 2017ISAR0005 . tel-01578068

HAL Id: tel-01578068

<https://theses.hal.science/tel-01578068>

Submitted on 28 Aug 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse

UNIVERSITE
BRETAGNE
LOIRE

THESE INSA Rennes
sous le sceau de l'Université Bretagne Loire
pour obtenir le titre de
DOCTEUR DE L'INSA RENNES
Spécialité : Informatique

présentée par

Zhaoxin CHEN

ECOLE DOCTORALE : MATISSE

LABORATOIRE : IRISA – UMR6074

Recognition and interpretation of multi-touch gesture interaction

Thèse soutenue le 28.04.2017
devant le jury composé de :

Hubert CARDOT

Professeur des universités, Université François-Rabelais de Tours / Président

Véronique EGLIN

Professeur des universités, INSA de Lyon / Rapporteur

Nicole VINCENT

Professeur des universités, Université Paris Descartes / Rapporteur

Harold MOUCHERE

Maître de conférences, Université de Nantes / Encadrant

Christian VIARD-GAUDIN

Professeur des universités, Université de Nantes / Co-directeur de thèse

Eric ANQUETIL

Professeur des universités, INSA de Rennes / Directeur de thèse

Recognition and interpretation of multi-touch gesture interaction

Zhaoxin CHEN



En partenariat avec



Avec le soutien de



Table of Content

| | | |
|----------|--|-----------|
| 1 | Introduction For Gesture Based Human Computer Interaction | 5 |
| 1.1 | Organization of the manuscript | 6 |
| 2 | State Of The Art | 9 |
| 2.1 | Preamble | 9 |
| 2.2 | Definition | 10 |
| 2.2.1 | Direct manipulation & Indirect command | 10 |
| 2.2.2 | Touch Gesture | 11 |
| 2.2.3 | Early Recognition | 12 |
| 2.3 | Handwritten Gesture Recognition | 13 |
| 2.3.1 | Single Touch Gesture | 13 |
| 2.3.2 | Multi-stroke Gesture | 16 |
| 2.3.2.1 | Trajectory based | 17 |
| 2.3.2.2 | Structure based | 19 |
| 2.3.2.3 | Feature based | 21 |
| 2.3.3 | Multi-touch Gesture | 22 |
| 2.4 | Early Recognition | 25 |
| 2.5 | Structured Document Composition and Recognition | 30 |
| 3 | Multi-touch Isolated Gesture Recognition | 33 |
| 3.1 | Introduction | 33 |
| 3.2 | Graph Modeling with Allen’s Relations | 34 |
| 3.2.1 | Graph Modeling | 36 |
| 3.2.2 | Graph Embedding | 40 |
| 3.2.3 | Global Shape Representation (GSR) | 40 |
| 3.2.4 | Experiments | 41 |
| 3.2.4.1 | Dataset | 41 |
| 3.2.4.2 | Results | 43 |
| 3.3 | Graph Modeling with Motion Based Features | 46 |
| 3.3.1 | Preprocessing and Stroke Segmentation | 47 |
| 3.3.2 | Gesture to Graph | 48 |

| | | |
|----------|--|------------|
| 3.3.2.1 | Geometry feature | 48 |
| 3.3.2.2 | Topology relation | 49 |
| 3.3.2.3 | Graph definition | 52 |
| 3.3.3 | Graph matching and classification | 54 |
| 3.3.3.1 | Subgraph matching for stroke comparison | 54 |
| 3.3.3.2 | Edge matching | 57 |
| 3.3.3.3 | Graph classification | 58 |
| 3.3.4 | Experiments | 59 |
| 3.3.4.1 | Dataset | 59 |
| 3.3.4.2 | Comparative results | 61 |
| 3.4 | Conclusion | 63 |
| 4 | Reject Option Based Early Recognition Algorithm | 65 |
| 4.1 | Introduction | 65 |
| 4.2 | Multi-classifier Early Recognition | 68 |
| 4.2.1 | Segment Classifier | 68 |
| 4.2.2 | Rejection Algorithm | 69 |
| 4.2.2.1 | Ambiguity rejection | 71 |
| 4.2.2.2 | Outlier rejection | 72 |
| 4.2.2.3 | Threshold optimization | 72 |
| 4.3 | Experimental Result | 74 |
| 4.4 | Conclusion | 76 |
| 5 | Structured Document Composition in Multi-user Context | 79 |
| 5.1 | Introduction | 79 |
| 5.2 | Multi-user diagram database | 80 |
| 5.2.1 | Diagram acquisition | 80 |
| 5.2.2 | Diversity of the content | 81 |
| 5.3 | Eager interpretation based recognition system | 83 |
| 5.4 | Experiments | 88 |
| 5.5 | Conclusion | 92 |
| 6 | Conclusion & Perspectives | 93 |
| 6.1 | Conclusion | 93 |
| 6.2 | Perspectives | 94 |
| 7 | Résumé en Français | 96 |
| | Publications of the author | 111 |
| | Bibliography | 118 |

Chapter 1

Introduction For Gesture Based Human Computer Interaction

The ease with which we type a text with keyboard, navigate the web with mouse, command smart devices by voice comes from the decades of study on Human-computer interaction (HCI). The current ubiquitous direct manipulation interface, where visible objects on the screen are directly manipulated with a pointing device, was first demonstrated by Ivan Sutherland in Sketchpad [Sut63] in 1963. It supported the manipulation of objects using a light-pen, including grabbing objects, moving them, changing size, and using constraints. Nowadays, mouse has been a standard device to replace the light-pen as a virtual human finger in the interface. Usually people use two fingers on mouse, by left and right click, to achieve the majority of operations.



Figure 1.1: Touch gesture based manipulation interface.

With the development of touch screen devices and techniques, touch gesture interaction becomes more prevalent in Human-computer interaction domain. Comparing to the mouse, touch gesture is more in line with the natural behavior of human beings. By our nature, people are used to use fingers to grab, drag, stretch and manipulate an object.

Multi-touch screen techniques open a way to let people extend these operations on virtual elements. It provides the users a 2D environment enabling people contacting virtual objects with full fingers instead of using two fingers clicking. This technology has penetrated into common electronic devices around us. The smart phone abandons physical keyboard and uses virtual keyboard on screen instead. Traditional notebook is replaced by Touchpad which provides a more convenient writing environment. Large interactive advertisement board appears on the street so that people can easily find the interest they want.

Since touch gesture has replaced most of the operations of mouse and keyboard on screen, the question then raises whether we can use touch gestures beyond direct manipulations. In reality, hand gesture can be used as a form of non-verbal communication between people through sending and receiving wordless clues. An example is American Sign Language(ASL), which is a natural language that chiefly uses manual gesture to convey meaning serves for deaf communities. So is it possible to teach computer the sign language and make people use symbolic gestures to execute commands? Actually, the current handwritten character recognition is one of these techniques. It teaches the computers to recognize the input finger's trajectories and transform the trajectories to corresponding characters. By the nature of character writing, current studies mostly focus on the mono-touch trajectories recognition, i.e. the trajectories are always written by a single finger. There are few research on the study of multi-touch gesture (like the ASL, with both hands, ten fingers) recognition for symbolic command.

In this thesis, we study the multi-touch gesture recognition problem. We explore the possibility of using multi-touch gesture for not only direct manipulation but also indirect symbolic command. The main contribution of this work is to provide tools and algorithms, mainly based from pattern recognition and machine learning theories to enrich User Interface Design solution. Of course, many other contributions that are not addressed in this work are required to design a global system. Specially, human factors and ergonomics, user experiences, user interface scenographies are beyond the scopes of this works. Our goal is to address the complexity of multi-touch gestures and develop a system to well analyze and recognize multi-touch gestures.

1.1 Organization of the manuscript

The manuscript of this thesis is organized as following:

Chapter 2: There have been years of studies for the usage of touch gestures. We introduce in this chapter the state-of-the-art research for both direct manipulation and indirect command. We discuss the different type of underlying recognition methods for both mono-touch gesture and multi-touch gesture.

Chapter 3: In this chapter, we propose a graph modeling strategy to characterize the temporal and motion features of multi-touch gesture. To verify our graph modeling and recognition strategy, we build a MTGSet which is a multi-touch gesture dataset as benchmark test.

Chapter 4: The current use of multi-touch gestures is mostly restricted to direct manipulations. We study the possibility of using multi-touch gesture for both direct manipulation and indirect command. We propose an early recognition strategy enabling the system to recognize a gesture by its beginning part in order to give a feedback to the user as soon as possible.

Chapter 5: The final goal of our research is to provide a real context that includes different types of gesture commands. As a first step towards this target, we develop a multi-user structured document composition environment where two users can simultaneously use gestures to compose a diagram. We build a multi-user gesture dataset and develop an eager recognition system to recognize the gesture on-the-fly in order to give a real time feedback to the users.

This project is co-funded by the region of Bretagne and Pay-de-la-Loire. We also thank `Excense` company for their support.

Chapter 2

State Of The Art

2.1 Preamble

This chapter presents an overview of the research and development status of the handwritten gesture recognition problem. To broaden the scope of the presentation and clarify some terms related to the domain, in section 2.2, we will first give a brief introduction of the basic concepts used in human computer interaction. Specifically, one important point is to make a distinction between direct manipulations and indirect commands, to introduce the notions of early recognition or lazy recognition. Each of these activities does not provide the same service and it is important to understand what can be expected as outcomes.

Then, in section 2.3, we focus more specifically on gesture interaction, and propose a classification of gestures related to the number of strokes, the number of simultaneous touch points, and the number of simultaneous users. The reason being that in this work, one original scope is to address multi-stroke and multi-touch gestures.

When developing a recognition system, the necessity of real-time classification based upon the principle that users must receive immediate and appropriate visual feedback about the effects of their actions has to be considered. In this context, early recognition methods are introduced in section 2.4. This strategy aims at recognizing gestures from their distinctive beginning part and achieves to recognize as soon as possible.

Finally, a global use case is proposed to conclude this chapter through the study of structured document composition systems. Such systems combine multi-touch interaction with recognition paradigms to facilitate digital document production. We will review the general handwritten document recognition methods and discuss the difficulties to associate them with multi-touch gesture.

2.2 Definition

Devices enabling touch input have been developed over the last decades providing a more convenience way for human-computer interaction. In recent years, touch gesture based applications have been growing considerably due to the prevalence of the smartphone and touch pad. In this section, we will illustrate some important concepts for handwritten gestures that help the readers to understand challenging work in this domain.

2.2.1 Direct manipulation & Indirect command

While as presented in the general introduction, many different modalities can be used to perform interaction, we will introduce the concepts of direct manipulation and indirect command using touch gesture examples. From the interaction point of view, touch gesture can be used for two aspects: direct manipulation for (virtual) elements and indirect symbolic gesture for triggering command. Common examples of direct manipulation are: using one finger holding on an element for selecting and dragging the element for moving. A more comprehensive context for direct manipulation can be found in a map view application, where users may scroll (pan) the map by dragging, change the zoom level by using two fingers for a pinch or stretch gesture, rotate the map by placing two fingers on the map and applying a rotate motion. Usually a direct manipulation system would give a continuous feedback, such as scroll the map, while the users move their fingers.

The indirect symbolic gesture is a one-shot operation commonly used in sketch-based interaction. Users may draw a certain gesture to trigger a corresponding pre-defined command. Such examples can be found in [AZ09], where the stroke gestures are used as shortcuts to draw predefined objects (shown in Fig. 2.1). Another widely used indirect command example is handwriting, where user input a character by drawing its shape instead of using keyboard.



Figure 2.1: Handwritten touch gesture for indirect command: (a) Users may input an icon from the menu. The menu also shows the mapping from strokes to icons. (b) Using stroke as the shortcut to input the icon instead of selecting from menu. [AZ09]

Since the two interactions offer different feedback to users, the underlying recognition

strategies have a clear difference. In the following section, we will review the recognition methods based on different gestures and different interactions.

2.2.2 Touch Gesture

Depending on the different interaction contexts, touch gesture varies from shape, number of strokes and number of simultaneous touches. In [SW13], authors summarize the different type of touch gestures based on the number of strokes and touches as shown in Fig. 2.2. Here, a **stroke** is regarded as the trajectory of a touch (finger, stylus, etc.) during uninterrupted contact on the sensing surface. The **touch number** indicates the number of concurrent contacts involved. For instance, the “**single-touch**” describes gestures only using one touch per stroke. This is the simplest touch gesture that can be used for both direct manipulation and indirect command. Common examples of single touch direct manipulation could be dragging on a virtual element or scrolling on a map application. The “**multi-stroke**” is the gesture which contains at least two strokes by one touch. This gesture is widely used for inputting sketches or characters which contains complex structures that can hardly be achieved by one stroke. Usually systems for “**multi-stroke**” give respond only after all the strokes are performed. Therefore “**multi-strokes**” gesture is only used for indirect command.

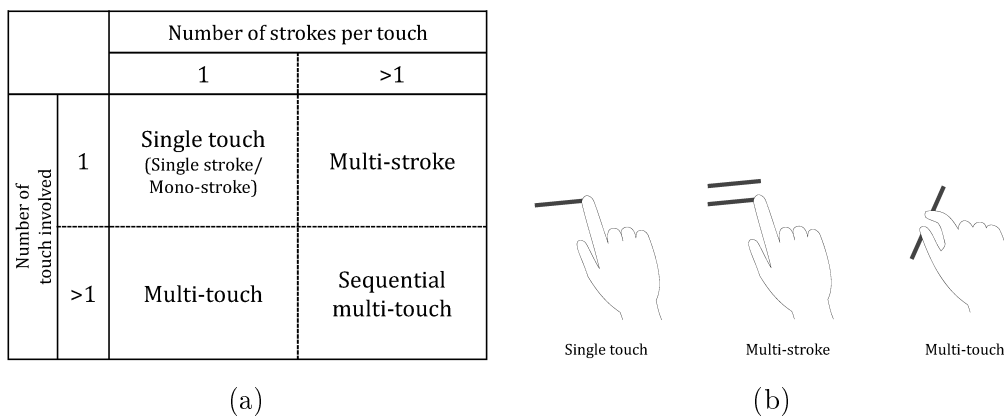


Figure 2.2: (a) A surface gesture taxonomy based on the number of strokes and touches. [SW13] (b) Examples of different type of gestures.

The term “**multi-touch**” is a more complex gesture such that more than one concurrent touch is involved. Such gestures are mostly used for direct manipulation such as zoom or rotation, where two or more simultaneous touches are performed on a sensing device. To the best of our knowledge, there is only a few studies and applications using “multi-touch” gesture for indirect command. For instance, a Mac’s multi-touch trackpad[Sup16] provide 8 direct and 7 indirect pre-defined gesture commands with different number of touches involved. Fig. 2.3 gives 4 examples of multi-touch indirect gesture commands, which are used as a shortcut to open Mission Control or show Notification Center. Most of

the indirect gestures are simply swipe based, which use 2 or more fingers moving towards a same direction. Only 'show desktop' and 'launchpad' adopt complex gesture, which require 4 fingers spreading or pinching, for indirect commands. Meanwhile, this trackpad does not support for user-defined gesture command so that it is not as convenient as shortcut from keyboard. Another indirect command use case can be found in [SBMI12], where 'multi-touch' gesture are considered as a remarkable biometric signal for user identification. In our view multi-touch gestures are supposed to be of special interest as they involve all degree of freedom. Aiming at the development of a general multi-touch gesture recognition system, our work focus on using multi-touch gesture as indirect command.

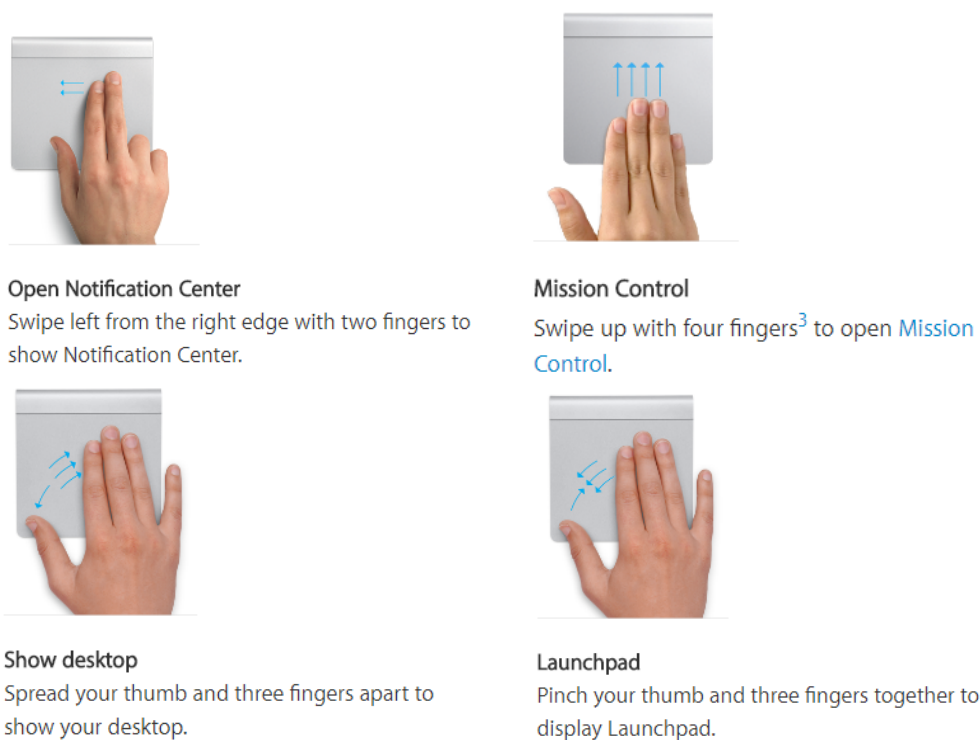


Figure 2.3: Indirect gesture commands supported by Mac's trackpad.

Lastly, the term “Sequential multi-touch” is not a common used gesture type. It refers to the gesture if it incorporates at least two subsequent strokes and simultaneous touches. It is defined and studied in [SW13], where authors aims to provide a classifier that covers all kinds of strokes and touches. Fig. 2.16 gives examples of sequential multi-touch.

In section 2.3, we will review the recognition strategies from the basic single touch gesture to the most complex multi-touch gesture.

2.2.3 Early Recognition

In general, an indirect gesture recognition system outputs the result after the end of a gesture. However, since touch trajectories are online data, gestures may have significant differences in their beginning part so that the recognition can be achieved before their end.

The underlying strategy is named as Early Recognition (ER), which aims at providing results before input gestures are completed. This strategy is important to develop practical and intelligent gesture or motion based man-machine interface. For example, motion based video games allow user to use motion gesture (such as jumping, squatting, hand waving) to control the game character. If the recognition process is executed after the gesture, there would be a significant delay between the user and game character. This delay decreases the usability and user experience of the system. In order to reduce the delay, early recognition has to be involved so that gestures can be recognized as soon as possible.

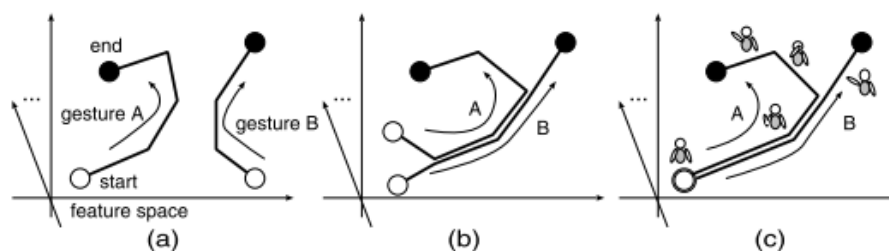


Figure 2.4: Gestures in feature space. (a) Two gestures have no common part. (b) Middle parts are common. (c) Beginning parts are common. [MUK⁺06]

Simply, early recognition can be achieved by partial recognition from the gesture's beginning parts. However, it would be ambiguous if several gestures have a common beginning part. Fig. 2.4 gives examples of possible relations between two gestures A and B in feature space. In Fig. 2.4(a) and (b), there is no common beginning part and thus we can easily expect correct early recognition results. In contrast, gestures in Fig. 2.4(c) have exactly same beginning part and can only be distinguished near the end. Therefore, an ideal early recognition algorithm should be able to detect these ambiguous common parts and find a balance between 'early' and 'accuracy'. In section 2.4, we will review the state-of-the-art methods for early recognition.

2.3 Handwritten Gesture Recognition

In this section, we discuss the state-of-the-art handwritten gesture recognition methods for different type of gestures.

2.3.1 Single Touch Gesture

As the simplest gesture, single touch gesture is widely used in many interaction system for both direct manipulation and indirect command. While used as direct manipulation, single touch gesture is mostly used for moving a selected virtual element. System need to measure the touch's movement direction, speed and displacement to give a corresponding

movement for the virtual element. Most of these movement information can be obtained directly from the underlying hardware and does not require complex recognition processing.

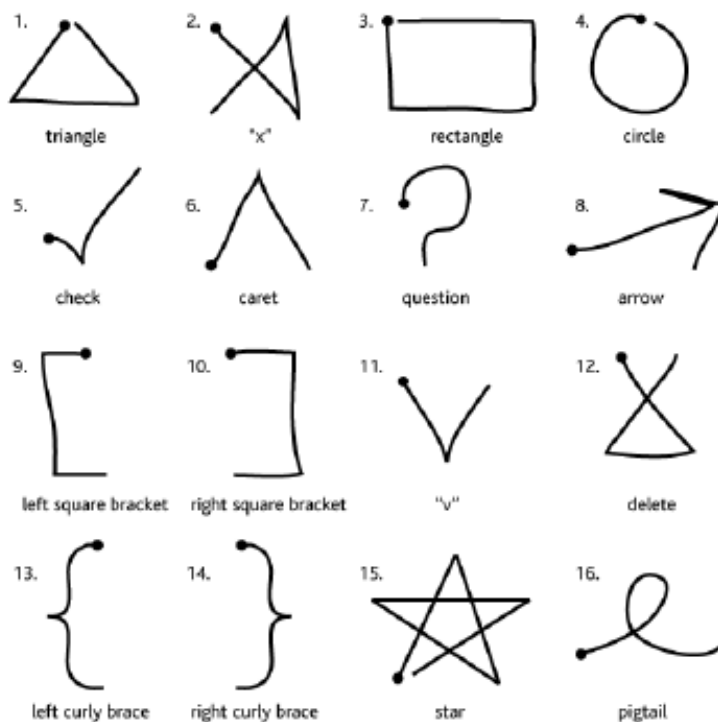


Figure 2.5: Single touch gesture set for executing commands. [WWL07]

It becomes a symbol recognition problem while gestures are used as indirect commands. As shown in Fig.2.5, which is a single touch gesture set studied in [WWL07]. Each gesture can be either linked to trigger a command or used as text input. Basically, there are two types of recognition strategies for single touch gesture (also named as single stroke, uni-stroke gesture): 1) sequence matching; 2) statistical recognition based on global feature. The online touch stroke data consists of a sequence of touch points from the touch down to the touch up. Dynamic programming algorithms such as dynamic time warping (DTW) can be employed for performing non-linear sequence matching. Fig. 2.6 shows an example of DTW matching studied in [NV05]. Each point is represented by its x and y coordinates as the local feature. In some other works, more elaborate local features are extracted for each point, such as velocity, pressure, curvature, etc. The DTW is then used to calculate a distance of the two sequences according to the points matching (as shown in Fig. 2.6(b)). For more details about DTW, the reader is referred to [VLOK01].

In [WWL07], authors present another competitive sequence matching algorithm named “\$1 recognizer”, for single stroke recognition. Instead of time warping, \$1 resamples each stroke into a fixed number of points. Then a candidate C is compared to each stored template T_i to find the average distance between corresponding points using a point-to-point matching equation,

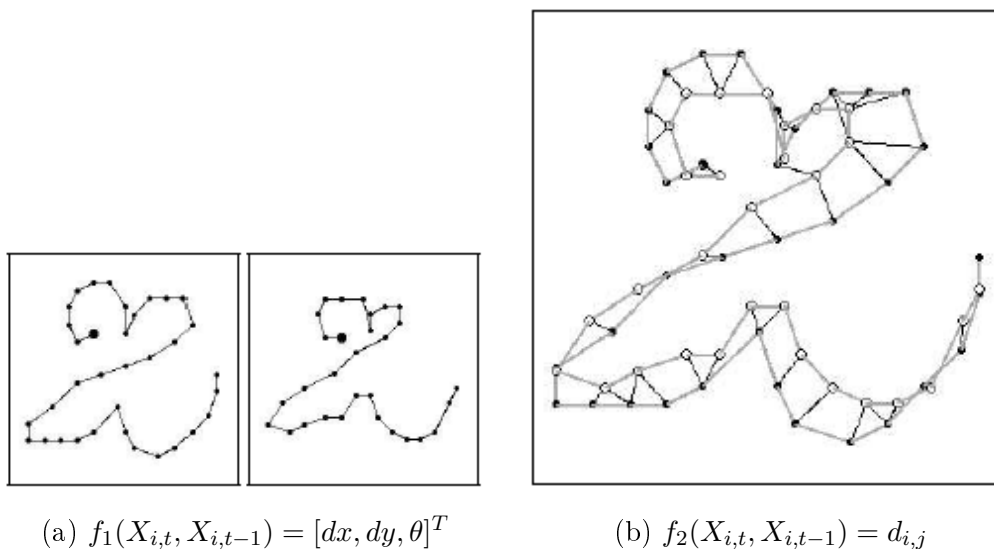


Figure 2.6: Example of a DTW matching. [NV05]

$$d_i = \frac{\sum_{k=1}^N \sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N} \quad (2.1)$$

where k is the point index for each symbol. x, y is the coordinate of each point. The template T_i with the least path-distance to C is the result of the recognition.

The sequence matching algorithm is easy, cheap and accurate with only a few loaded templates on a small dataset. It requires no complex mathematical procedures, yet competes with approaches that use statistical classification. Such algorithms are highly efficient for recognition of specific sets of simple gestures but do not tolerate much variation in the writing style or drawing process. To generalize the gesture representation and make use of the fast developing statistical recognition method, a tendency of using global features to characterize a gesture can be noticed for gesture recognition.

Generally global features are chosen according to the following criteria. Each feature should be meaningful so that it can be used in gesture semantics as well as for recognition. There should also be enough features to provide differentiation between all gestures and should not be too many for efficiency reasons. The Rubine's feature set is a typical global feature set which has been widely used for recognizing single stroke gestures [Rub91]. It employs 13 global features which are computed from a complete gesture shape. Fig. 2.7 shows a part of features used by Rubine, where (f_3) and (f_4) are the length and angle of the bounding box diagonal, (f_5) is the distance between the first and the last point. The full 13 features can be found in [Rub91].

After this feature extraction, each gesture is represented by a feature vector, $f = [f_1, \dots, f_{13}]$. Then classical statistical recognizer or linear/nonlinear machine can be implemented to achieve the classification.

Due to the fact that single touch gestures are chosen to be not ambiguous and simple to

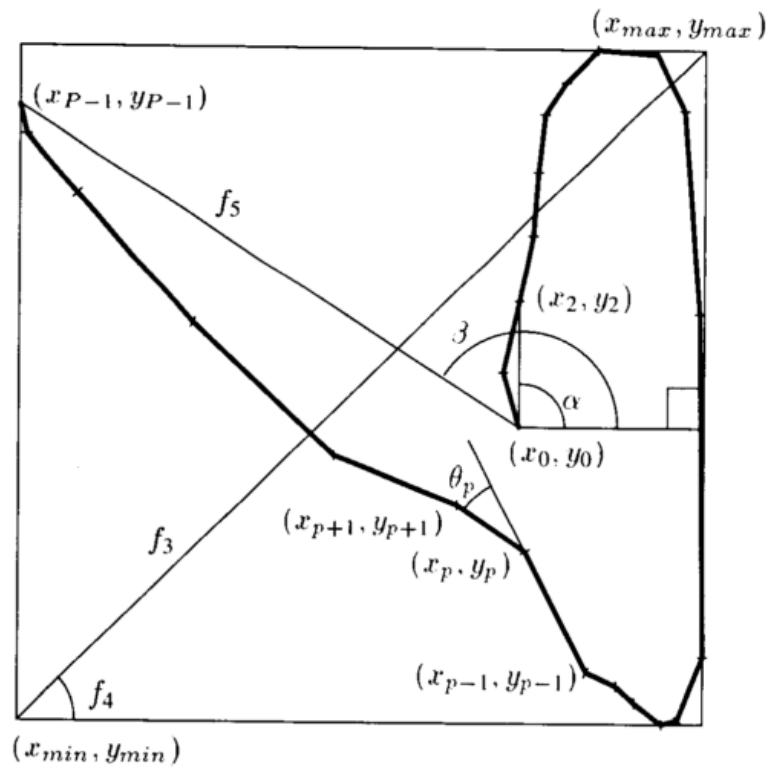


Figure 2.7: Features used to identify strokes. [Rub91]

be memorized has less variation and writing style, the methods above have been proved being efficient and accurate for single touch gesture recognition even though they are simple and cheap. Unfortunately, they are only moderately successful when applied to multi-touch/multi-stroke pen input. Shape variation, writing order, number of strokes have to be taken into account. We will provide a review of multi-stroke gesture recognition method in next section.

2.3.2 Multi-stroke Gesture

The term multi-stroke gesture here refer to the isolate symbols that contain two or more strokes, and all strokes are written in a sequence by a single contact (finger, stylus, etc.). Obviously multi-stroke gesture is not used for direct manipulation because users always expect an action after the end of the final stroke instead of during the writing process. Comparing to the single-stroke gesture, multi-stroke gesture contains more stroke variations and offer more freedom for the users. Therefore, it has a diversity of usage such as for drawing characters, pictograms, diagrams, etc. The diversity of patterns and high variation on stroke number and writing order make the recognition becomes a more challenging work. Large variety of methods have been proposed to solve this problem. This section propose to categorize these methods and provides a review of typical recognition methods for each category. Note that this thesis focus more on multi-

touch gesture recognition which is more complex but has less studies (shown in next section). We review the recognition methods for multi-stroke gestures in this section to reflect the difficulties for multi-touch gestures recognition and also enlighten us the way to solve that problem.

2.3.2.1 Trajectory based

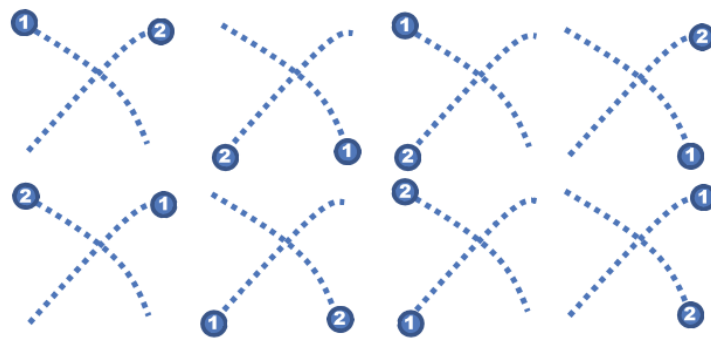
Since multi-stroke gesture is also written by a single contact, the trajectory can still be seen as a sequence of points but containing special pen up/down points. With an appropriate adjustment or constraint, the sequence matching strategy for single stroke gesture is still available for multi-stroke cases.

In [WWL07], authors add pseudo strokes to concatenate each two consecutive strokes. Each pseudo stroke starts at the end point of a previous stroke and ends at the start point of the following stroke. The multi-stroke gesture is then transformed as a single stroke so that any single stroke recognition method can be used. A similar strategy can be found in [NWW08], authors also add the pseudo strokes and implement the traditional DTW algorithm for gestures comparison. They give the constraints for DTW that each point in pseudo stroke can only be matched to a point in another pseudo stroke, same for the “real” point. Classification of a test sample is performed through nearest neighbor criteria with the DTW distance function.

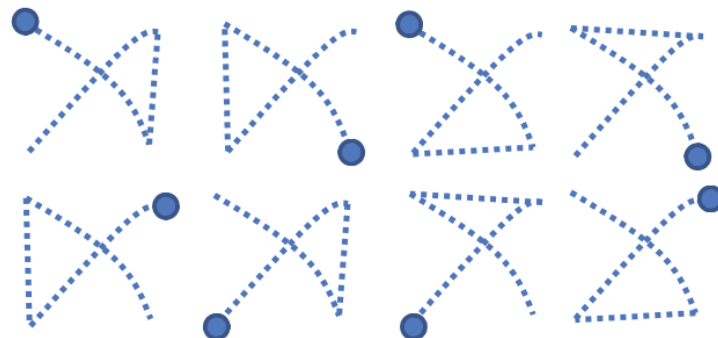
Even though these methods well transform the multi-stroke problem to the single stroke problem, the concatenation between two strokes limits the gesture to be written in a fixed order and direction. To ensure the different stroke orders and directions can be properly recognized, [AW10] present \$N recognizer which is a significant extension to the \$1 unistroke recognizer introduced in previous section. Basically, \$N goes further by recognizing gestures comprising multiple strokes and automatically generalizing from one multi-stroke to all possible multi-strokes using alternative stroke orders and directions. The main idea is to generate all permutations of the component strokes. Each permutation represents one possible combination of stroke order and direction. There are 2^N combinations for N strokes. Fig. 2.8(a) shows 8 possible permutations for a two-stroke “x”. The permutations of possible combinations are then converted to unistroke (shown in Fig. 2.8(b)) by simply connecting the endpoints of component strokes as presented in [WWL07] and stored in template set for comparison. At runtime, each candidate multi-stroke gesture is also connected in the drawn order to form a unistroke and compared to all unistroke permutation templates using the \$1 algorithm.

Obviously, the brute force of creating all permutations to represent a multi-stroke gesture results in a combinatoric explosion when the stroke number is large. This method is efficient for the gestures which contains a few strokes but not suited to recognition messy drawing such as Chinese characters or sketchy symbols.

These trajectory based methods are simple, require little processing resource, and



(a) The 8 permutations for a two-stroke “ x ”. The numbered dots indicate stroke order and beginnings.



(b) The 8 unistroke permutations for a two-stroke “ x ” generated from (a)

Figure 2.8: Example of all permutations for a two-stroke “ x ” and its unistroke representations. [AW10]

easily extensible to new class because most of them rely on nearest-neighbor classification paradigm. This simplicity enable easy incorporation of multi-stroke gesture recognition for user interface prototypes which do not require heavy and complex engine.

2.3.2.2 Structure based

Instead of transforming the multi-stroke into a single sequence, the structure based methods focus more on each individual stroke and the inner relations between each two strokes. These methods usually apply to the multi-stroke symbols which contain much more strokes and are insensitive for the writing order, such as handwritten sketch. Fig. 2.9 shows 2 examples of handwritten sketch from [MRLSL06] and [LLLW15]. Apparently, these sketch examples mainly differ from visual aspect and accord no importance to the online information. The main idea of structure based methods is to analyze the geometric relations between each two strokes and represent a symbol as a semantic network of strokes with their relations.

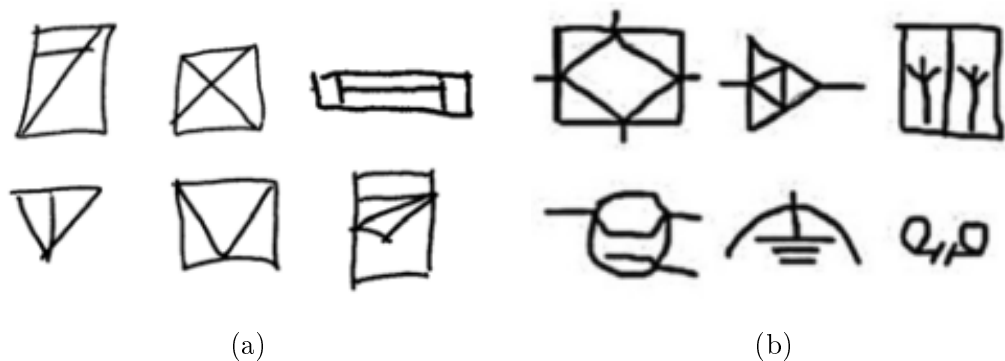


Figure 2.9: (a) Examples of architectural plans in [MRLSL06]. (b) Example of engineering drawing in [LLLW15].

In [LLLW15], authors present a typical state-of-the-art structure based sketch recognition method which exploits topology relations and graph representation for strokes. In sketch recognition domain, since the sketches are complex and contain large number of strokes, raw input of handwritten sketches usually consist of noisy and inaccurate strokes. As a standard techniques, the strokes are firstly refined and decomposed into a few basic primitives. Authors adopt the refinement approach in [XWJS02] which has four steps: polygonal approximation, agglomerate points filtering, endpoints refinement and convex hull calculation. Details can be referred in [XWJS02]. An example is given in Fig.2.10 where an open ended triangle becomes closed after processing. The refined strokes are then segmented into sub-strokes and fitted to a few primitive shapes. Note that the refinement and segmentation perform well on clean dataset, noisy or highly curved strokes are still hard to be segmented properly.

There are plenty of ways to define the relations for each pair of primitives. Mas et. al

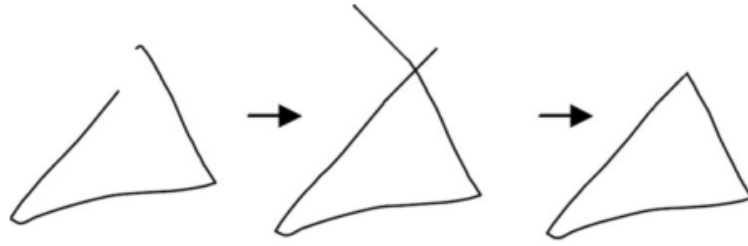


Figure 2.10: Example of stroke refinement.[XWJS02]

[MRLSL06] present an approach to generate a set of adjacency grammars based on five relations (Parallel, Perpendicular, Incident, Adjacent, Intersects). This set determines the final grammatical ruleset to characterize a symbol. [LLLW15] presents a more comprehensive topology definition according to the type of the involved primitives and the number of their intersections. Fig. 2.11 shows the examples of topology relations between primitives.

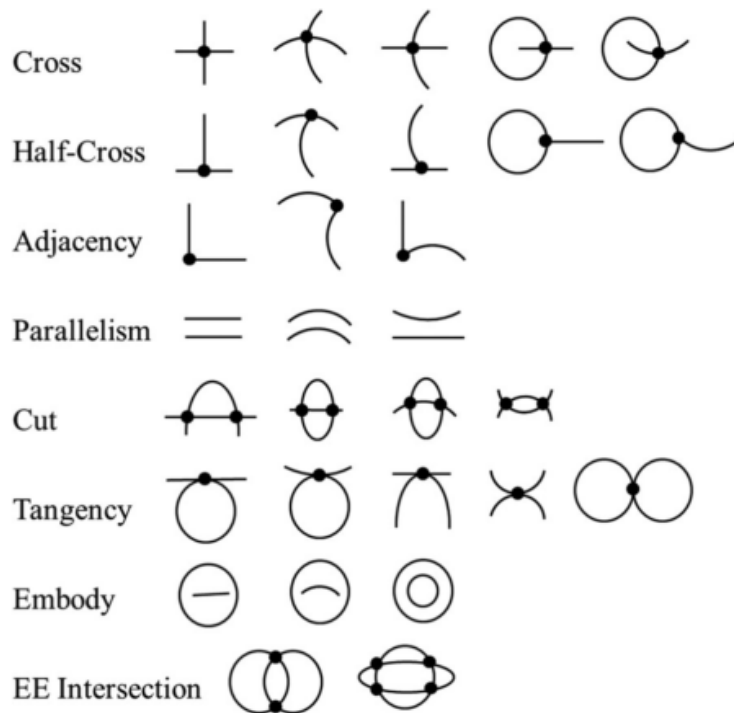


Figure 2.11: Examples of topology relations between primitives.[LLLW15]

Further more, in order to well illustrate the structure, in [LLLW15] authors use a topology graph representation to integrate both topology and geometry information as shown in Fig. 2.12. In graph representation, each vertex is a primitive in the sketch, each edge indicates a certain topology relation of a pair of primitives. To give a more precise description of the relations, authors also measure the spatial distance between two primitives as the complementary feature for the relation. It is shown as the weight on the edges.

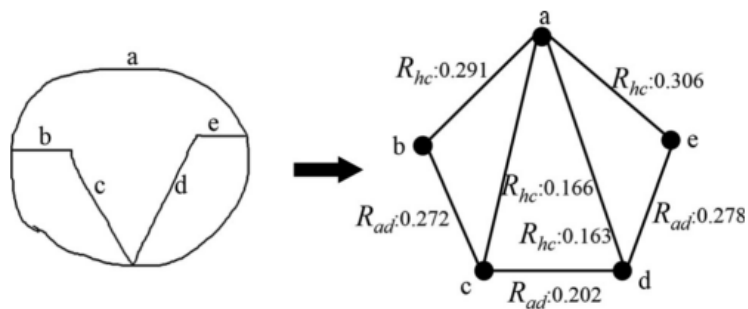


Figure 2.12: A sketch and its topology graph with the relations and geometry feature on edges. R_{ad} means adjacency relation for two primitives which have a common endpoint. R_{hc} means half-cross relation that one primitive has an endpoint joining some inner point of another primitive. [LLLW15]

By its nature, graph representation usually has a complex topology structure since it measures relations for each primitive pair. Due to the fact that different researchers define different topology relations and features for the primitives, it is difficult to give a general recognition strategy for graph representation. Normally, researchers design their specific graph matching and recognition method according to the graph structure and feature they used. Sousa et al. [SF10] use graph spectra to map graphs into vector descriptors. The spectrum of a graph G (which consists of n eigenvalues, where n is the number of nodes) is computed from the eigenvalues of its adjacency matrix. They compute the similarity between graphs by calculating a distance between the correspondent descriptors. [LLLW15] define a novel product operation to calculate the element-wise multiplication of two adjacency matrices. This operation is used to find the common structure parts of two graphs and measure the similarity between two sketches.

The graph based methods are slow because they perform graph matching for each sketch in the dataset, and their runtime increases with respect to the complexity of graphs. This makes it hard to directly apply to a large scale database. To accelerate the recognition process, Bunke et al. [RB10] present a general method of transforming any graph into vector descriptors. We will detail this method in later sections.

2.3.2.3 Feature based

Extracting features to train and feed a statistical classifier, such as kNN, neural networks, support vector machine (SVM), is a more popular solution in pattern recognition. A very close topic which has been widely studied is isolated character recognition [DLJZ07] where characters can be seen as isolated multi-stroke symbols. In our specific case of indirect multi-stroke gesture recognition, simple features inherited from Rubine's ones can be extended. A tendency is concerned to induce more and more complex features based on both static (stroke number, convex hull, area) and dynamic (average direction, velocity, curvature) information. In [JZ07a] authors design a 14 features set. In [NWV08] authors

employed Rubine’s features and an additional 15 other global features that make it to 28. In [WNvGV09] authors design 20 new global features and add it to those 28 features that results to a 48 features set. In [DA13] authors design their own 49 features set.

We will not detail each feature. Most of the features are geometry features as Rubine’s features which describe the appearance of the writing result. The role of a feature set is to numerically describe symbols and create boundaries between them, so one symbol can be discriminated from another in the corresponding feature space. It is difficult to tell which feature set or individual feature is most discriminative. The recognition result is also highly related to the recognition method and dataset they used. In [DA13] authors compare different feature sets on four datasets. There is no evidence to believe that a more complex feature set can always yield a better result. Some simple and universal feature sets can also outperform the systems that were designed, tuned and optimized for recognition of specific datasets.

2.3.3 Multi-touch Gesture

The multi-touch gesture interaction become popular with the development of touch screen display technique in recent years. A common sense for the usage of multi-touch gesture is to directly manipulate a virtual element on the touch interface. Usually a direct manipulation based system has limited gesture vocabulary such as click for selection, drag for moving, pinch for zoom, etc. System need to give the correct feedback to the user at very early stage of a gesture. The recognition for direct manipulation is achieved by analyzing the spatial displacement of fingers over time. For instance, in [OIL11] authors allow three motions for multi-touch direct manipulation: translation, scaling and rotation. Fig.2.13 shows the three motions according to the displacement of touch points.

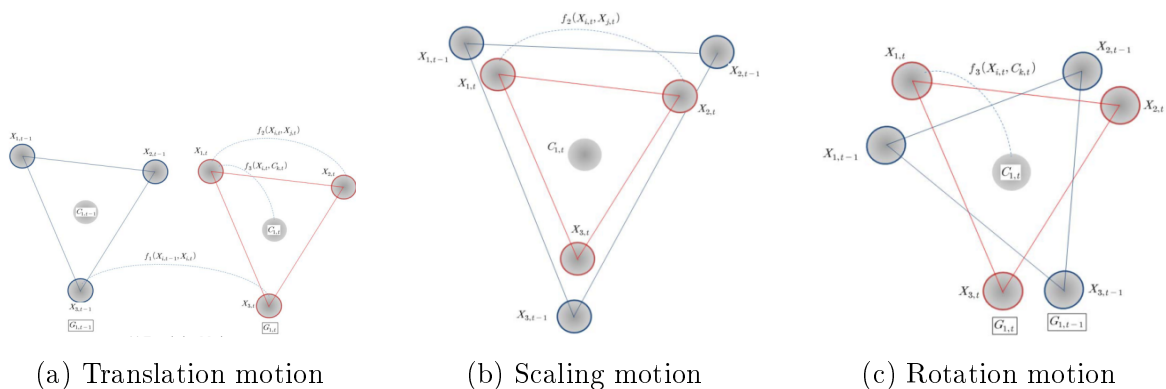


Figure 2.13: The displacement of three touch points from time $t-1$ to t can be translation, scaling and rotation simultaneously. The f_1 , f_2 and f_3 are feature functions which are respectively related to translation, scaling and rotation. [OIL11]

In order to detect the user’s intention from the trajectories, they define three motion parameters, respectively related to the three motions, that calculate the displacement of

touch points between each time $t - 1$ to t . The definition of three motion parameters are shown in Fig.2.14. The system triggers the corresponding motion operation when the variation of any motion parameter is larger than a certain threshold. Another similar method can be found in [ORB⁺15], where authors define three distance functions (swipe, rotate and zoom) to evaluate the displacement of fingers. The chosen motion is given by any of the three distance with the highest value.

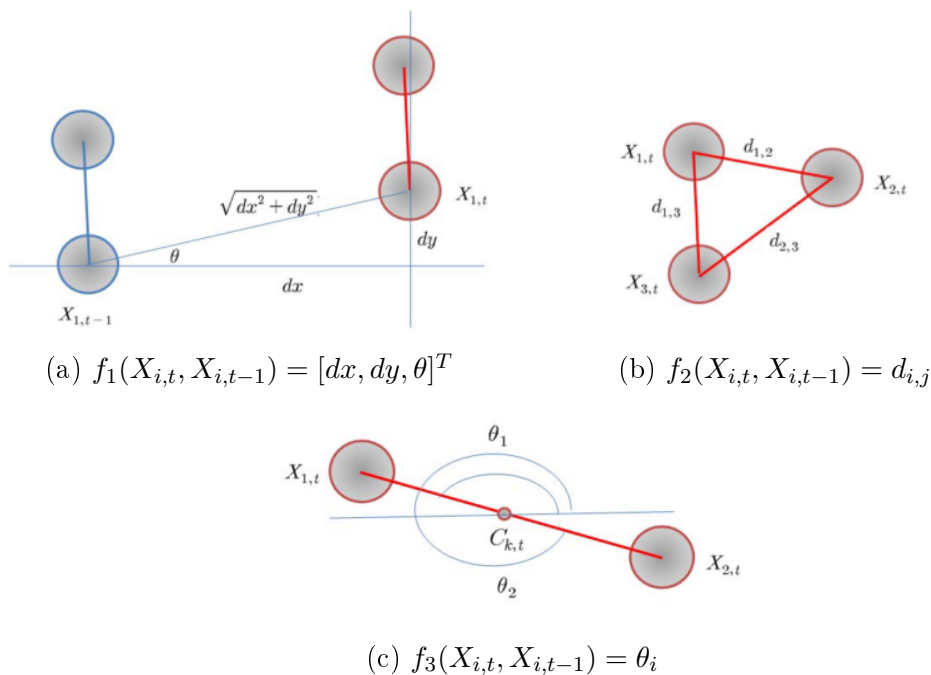


Figure 2.14: The motion feature functions: f_1 measures the translation vector of i^{th} point pair between time $t - 1$ and t . f_2 measures the distance $d_{i,j}$ between points in a certain time. f_3 measures the rotation angle θ_i of i^{th} node between time $t - 1$ and t . [OIL11]

To the best of our knowledge, there is few researches aiming at the development of a multi-touch gesture recognition system for indirect command as well as the multi-stroke gesture interaction. Unlike the multi-stroke gesture where strokes are always written in sequence, the strokes in multi-touch gesture may have complex synchronization or intersection relations. Two gestures may have the same appearance but contain different inner-stroke relations. A fundamental issue is the modeling of these relations between strokes as the key feature for multi-touch gesture recognition.

Some context dependent works use syntactic approaches, i.e. a textual description for strokes, to describe the movements and temporal progressions of a multi-touch gesture. For example, Kammer et al. [KWK⁺10] present the GeForMT (Gesture Formalization for Multi-touch) where the finger traces are abstracted to atomic gestures (POINT, LINE, CIRCLE, SEMICIRCLE, etc.). Then they use respectively symbolic operator and prefix elements to denote the temporal progression and relative movement between traces. The resulting syntax looks like 1F (HOLD) * 1F (SEMICIRCLE), which means one finger is

holding on the screen while another finger is performing synchronously a SEMICIRCLE type gesture. An example is shown in Fig. 2.15. A similar approach can be found in [KHDA12], where authors specify each gesture as a regular expression over a stream of touch event. The recognition is involved by matching the event stream to a list of predefined regular expression in order to trigger the operation. These methods abstract each stroke to a predefined atomic gesture and focus more on the dynamic relations between strokes. Because of lacking of analyzing global geometry features, these methods are believed to apply to simple multi-touch gesture but fail to recognition gestures which have complex shapes.

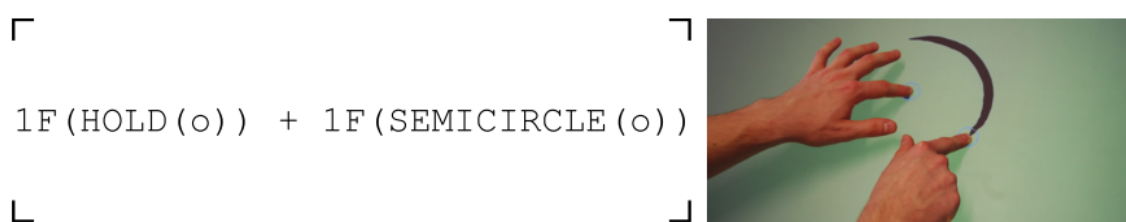


Figure 2.15: Rotate gesture described by GeForMT, where $1F$ means number of finger, *HOLD* and *SEMICIRCLE* are predefined atomic gestures, o means the finger is focusing on an object. [KWK⁺10]

In [SBMI12] [SBMIA14], Sae-Bae et al. consider multi-touch gestures as a remarkable biometric signal for user identification. The recognition is achieved by comparing the shape of two gestures stroke by stroke and calculating a distance to measure the similarity. This system analyzes the geometry feature for each stroke and compares two gestures from the global point of view. However, it assumes that each gesture has a fixed number of strokes and all strokes are performed synchronously. This strategy is not general enough to deal with the various types of multi-touch gestures.

The most related study can be found in [SW13]. They aim at the development of a multi-touch gesture recognition system for self-defined gestures as well as sketch-based interaction techniques. Fig. 2.16 shows some examples of their gestures set.

Fig. 2.17 provides an overview of their recognition procedure. The main contribution of the system is the feature extraction of each stroke and a pairwise stroke matching based classifier. In feature extraction, they extract not only the local shape features of each stroke but also the relative structural and temporal features within the gesture. Based on these features, they independently build a statistical model for each stroke in the gesture. A gesture template is shown as a set of statistical models of all its strokes. The classifier is achieved by comparing an input gesture with every gesture template regarding each stroke model. This is done by handling strokes separately and computing a matrix containing pairwise matching likelihoods of template strokes and input strokes. A best matching likelihood is computed by solving the maximum matching problem formulated by this matrix. However, a shortage of this research is that all the gestures in this

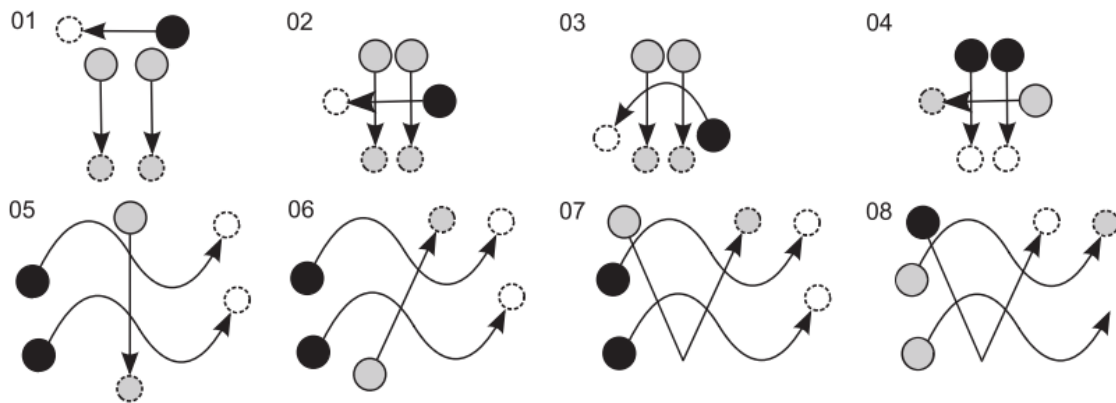


Figure 2.16: Examples of indirect command oriented multi-touch gestures presented in [SW13]. Larger dots are depicting the start of a trajectory (from one touch), the arrows their movement and dashed smaller dots symbolize their end. Different strokes are colored differently, black elements belong to the first stroke, gray ones to the second.

dataset consist of 3 strokes, authors did not study the robustness for varied number strokes. Moreover, this dataset does contains gestures which have same appearance but only differ from written order (e.g. in Fig. 2.16, gesture No. 07 and No. 08 have same two synchronized strokes, they differ from the written order of the third stroke). But there are no two gestures which have same appearance but differ from the synchronized strokes. These problems have not been addressed in this state-of-the-art work and they may limit to recognize the user's self-define gestures. The community still need a more general multi-touch gesture dataset and more study on multi-touch gesture recognition problem.

2.4 Early Recognition

As we illustrated in previous section, the recognition for direct manipulation gesture and indirect command gesture are completely different. The former one use a real-time strategy which analyze the fingers' trajectories during each time interval, while the latter one has to wait until the end of trajectories, and analyze the global structure or shape of the gesture. Normally, practical applications use either of them for human computer interaction. However we imagine a certain context which support both of these interaction, an underlying recognizer may be confused about whether it should interpret the input as direct manipulation or wait until the end. The co-existence of these two usages requires a feedback as soon as possible to be consistent with a direct manipulation. Hence, an Early Recognition (ER) strategy is desirable to cope with these two kinds of commands.

A basic idea of ER is to employ a partial matching method, where the recognition result of an input pattern is determined by the matching distance of its beginning part from reference patterns. To the best of our knowledge, Petit *et al.* [PM13] proposed for the first

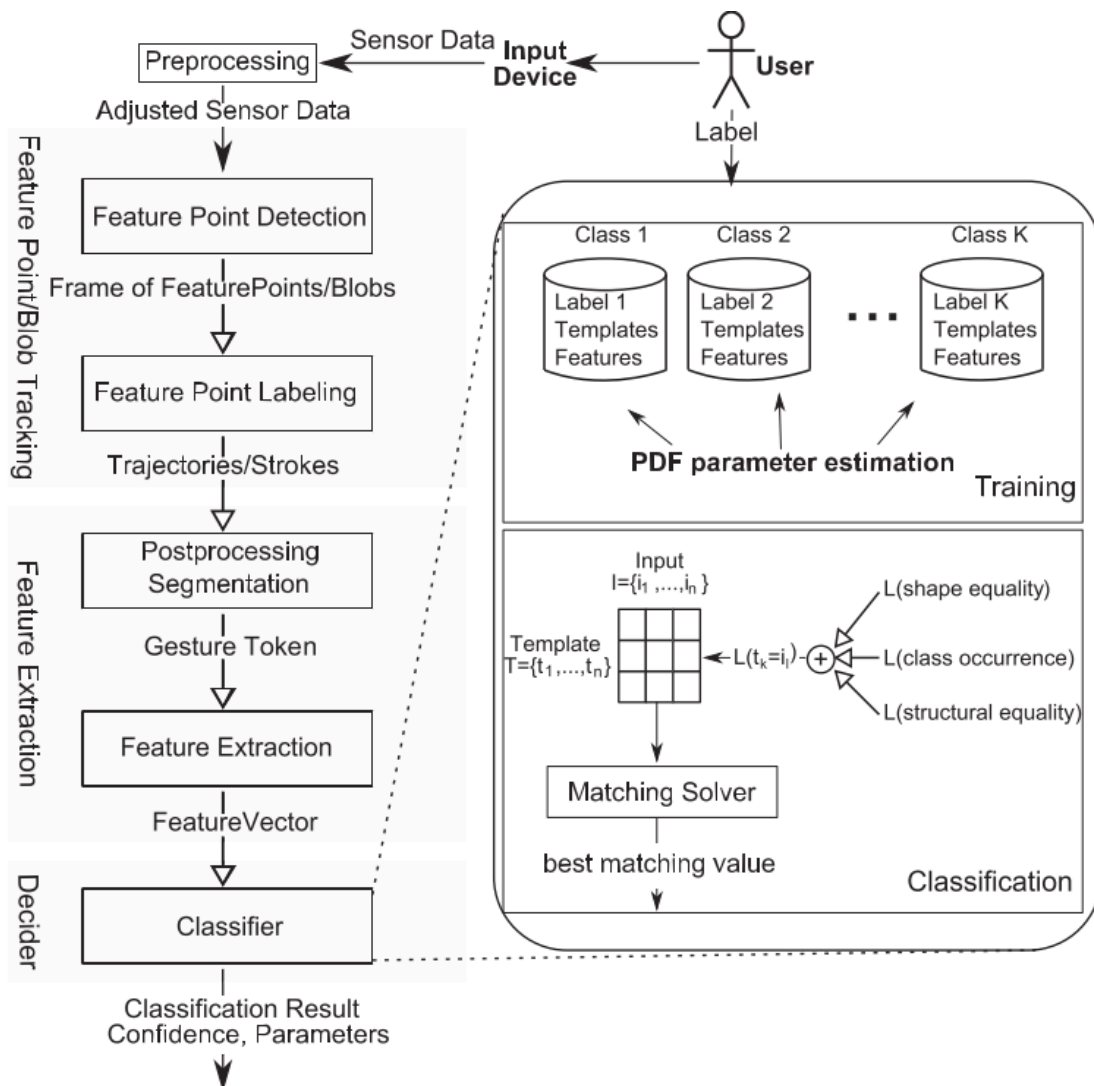


Figure 2.17: The overview of a multi-touch symbolic gesture classifier's architecture presented in [SW13].

time using mono-touch gesture for both command shortcut and direct manipulation in the same context. They consider that human gesture can be segmented into a sequence of motion units, based on velocity, duration, and shape. They provide a real-time description that interprets every piece of trajectory and either give continuous feedback during gesture articulation or at the end. Fig. 2.18 gives an example from their work. A “Heart” like gesture has different interpretations according to the progressing of its trajectory. The first two stage, “Press” and “Start-move” are default for each gesture. While in the stage 3, the trajectory is interpreted as “Drag”, which is a direct manipulation that triggers a continuous feedback to the interface. With the progressing of this trajectory, in stage 4 the “symbolic pattern” is detected. System cancels the drag effect and wait to the end of gesture to trigger a “Heart” command.

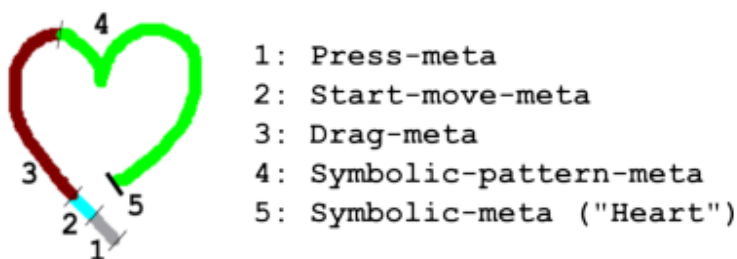


Figure 2.18: An example of early recognition for a gesture, the recognizer give different feedback according to the progressing of trajectory. [PM13].

This is a typical but imperfect example which well illustrates the strategy and difficulty for early recognition. Some gestures may contain very similar beginning part which confuse the system that gives a incorrect feedback. An ideal early recognition system should collect enough information before making a decision and avoid the ambiguous beginning part. Unfortunately, there are few ER works for on-line 2D handwritten gesture. We will review the early recognition method in related domain in the following part to explore the general idea for early recognition algorithm.

In [MUK⁺06], Mori *et al.* use early recognition for motion gesture prediction. The proposed early recognition algorithm is based on conventional dynamic programming (DP). Fig. 2.19 gives an overview of their recognition method. Let $I = I_1, I_2, \dots, I_\tau$ be an input gesture sequence, where I_τ represents its feature vector at frame τ . Similar, let $R = R_1, R_2, \dots, R_t$ be a registered reference gesture sequence, where R_t represents its feature vector at frame t . A conventional DP algorithm for recognizing the input gesture sequence I is considered as an optimal nonlinear matching algorithm between I and R . Their algorithm is described as following:

Step 1: For $\tau = 1, 2, \dots$, repeat Step 2-3.

Step 2: For $t = 1$, calculate the following DP-recurrence equation:

$$g_1(\tau) = 3d_1(\tau). \quad (2.2)$$

where $d_t(\tau)$ represents the distance between I_τ and R_t

Step 3: For $t = 2, \dots, T$, calculate the following DP-recurrence equation:

$$g_t(\tau) = \min \begin{bmatrix} g_{t-1}(\tau - 1) + 3d_t(\tau) \\ g_{t-1}(\tau - 2) + 2d_t(\tau - 1) + d_t(\tau) \\ g_{t-2}(\tau - 1) + 3d_t(\tau) + 3d_t(\tau) \end{bmatrix} \quad (2.3)$$

The non-linear matching distance at input gesture's frame τ is provided as

$$d(\tau) = g_T(\tau) \quad (2.4)$$

The above conventional DP algorithm works successfully to matching an input gesture sequence to entire part of a reference gesture. To achieve the early recognition, authors slightly modify the above algorithm. Specifically, the distance is re-defined as

$$d^*(\tau) = \arg \min_{t^*} g_t^*(\tau) \quad (2.5)$$

which means that the DP algorithm uses a local minimum distance at t^* th frame of the reference gesture as the partial matching distance (Shown in Fig. 2.19(b)). This method is simple and provide recognition results with far shorter recognition times than conventional algorithm. However, a simple distance based recognizer does not have the ability to identify the common beginning part of different gestures. This ambiguity or common beginning part strongly degrades the accuracy of the prediction results.

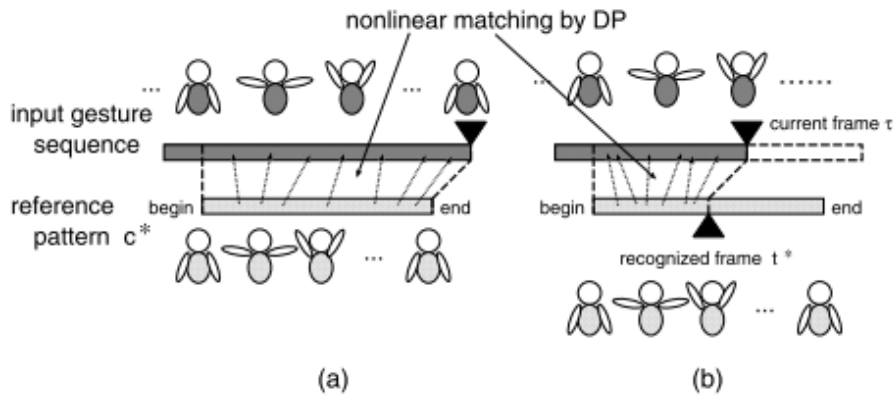


Figure 2.19: (a) Conventional gesture recognition. (b) Early recognition strategy. [MUK⁺06].

To deal with the common beginning part problem, in [KSNT11], authors propose to calculate the distance gap between the most and second most similar gestures. Denoting the nearest class c_1 and the second nearest class c_2 , the result is determined as the class c_1 when the difference of the two classes fulfills $d(c_2) - d(c_1) > th$. The threshold th is the parameter which determines the timing of early recognition. A large threshold would result a high recognition accuracy with the price of a high recognition time delay. In the

worst case, recognizer need to match the entire sequence of the input gesture to make the decision. In other words, there is a trade off between recognition time and accuracy, the earlier decision is made, the less accuracy results.

In [UA08], authors present a general early recognition method using multi-classifier strategy. They train a set of frame classifiers $h_1(x), \dots, h_t(x), \dots, h_T(x)$, where $h_t(x)$ is a frame classifier prepared at t th frame, x is a set of training pattern. The frame classifier $h_t(x)$ provides a recognition result by only using the feature vector of the t th frame. The recognition result at the t th frame will be determined by combining t recognition results provided by $h_1(x), \dots, h_t(x)$, i.e.,

$$H_t(x) = \text{sign}\left(\sum_{\tau=1}^t \alpha_{\tau} h_{\tau}(x)\right) \quad (2.6)$$

where α_{τ} is the weight of each individual classifier computed by the error rate of $h_{\tau}(x)$. One possible definition of α_{τ} is

$$\alpha_{\tau} = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right) \quad (2.7)$$

where ε_t is the error rate. Moreover, authors optimize this procedure using weight propagation. When training the frame classifiers from $t = 1$ to T , the patterns mis-recognized by the last classifier $h_{t-1}(x)$ are largely weighted not to be mis-recognized by $h_t(x)$. Fig. 2.20 illustrates this procedure. We refer readers to their paper for more details. This multiple frame classifier strategy achieves a better results than each individual frame classifier because the multi-classifier can form its discrimination boundary in a higher dimensional feature space (As shown in Fig. 2.20(c)).

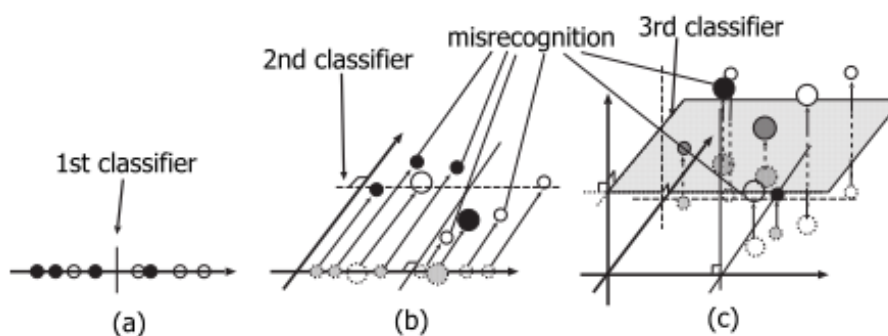


Figure 2.20: Three frame classifiers combined with weight propagation. [UA08].

In a short conclusion, the above methods try to solve the early recognition using different strategies, either partial matching between two sequences or multi-classifier at each frame. However, there are few works to explore the intrinsic problem of early recognition, i.e. how to well identify the ambiguous common beginning part of different gestures and give a optimal trade off between recognition time and accuracy. In chapter 4, we will present our reject option based method for this target.

2.5 Structured Document Composition and Recognition

By structured document, we refer to the documents that consists of two-dimensions arranged symbols (e.g. flowchart diagrams, musical scores, electrical circuits). Examples of a flowchart and a logic circuit are shown in Fig. 2.21. To realize the full diagram recognition, in general two subproblems have to be solved: 1) Detection of elementary units. 2) Retrieval the relations among elements. Unlike the isolated symbol recognition, where all strokes are known to be a part of one symbol, a stroke in a diagram can have many different interpretation depending on its context. A stroke could be a one-stroke symbol, a part of multi-stroke symbol or even contains several symbols. The complex stroke combination problem is so challenging that many existing recognition systems avoid it by placing constrains on the way user draws. For instance, some authors require users to draw symbol with only one stroke [LM01], some others require the user to provide explicit cues, such as making a pause between two symbols[HR07]. These constrains facilitate the recognition process but fail to match the way people naturally draw.

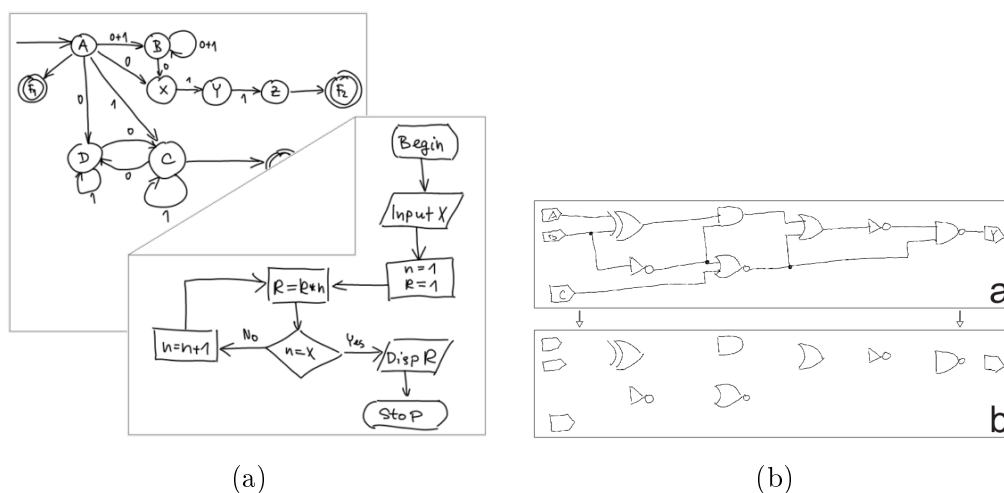


Figure 2.21: (a) Examples of diagrams with structure in [BPP⁺14]. (b) A digital logic sketch in [PSDA10].

From the interaction point of view, there are two recognition strategies for structured document recognition, shown in Fig. 2.22. The first is *lazy interpretation*, which uses the full document as the input and recognize all the symbols and their relations in one shot. The analysis process ignores the user's creative phase during the document composition and has full context to make the most sensible decision. Lazy interpretation offers the advantage that every stroke in a completed document is meaningful (despite the noise) for the nearby context. It means that a stroke is certainly either to be an isolated symbol, or supposed to be grouped with nearby strokes to form a multi-stroke symbol since full context is presented. The difficulty of detecting elementary units introduced above transforms

into a stroke grouping problem based on their spatial and temporal relations. Plenty of research give their effort in this orientation. [AFMVG11] proposed an online handwritten diagram dataset and generate most of possible segmentations filtered by some geometric constraints. The final segmentation is the one which maximize the probability of each symbol recognized by a classifier. [BPP⁺14] and [CLC13] use the same dataset but with different approaches. Bresler et al. [BPH13] estimate a general distance threshold based on the two closest points from two strokes to determine if they are enough spatially close to be a symbol candidate. Then the final decision is also seen as a max-sum optimization considering the symbol probabilities and their relations. In [CLC13] the recognition process is guided by a grammar base syntactic analysis which parses the full document structure. Peterson et al. [PSDA10] present a two step solution: first isolated strokes are classified and then a binary stroke grouping classifier (based one one temporal feature and 12 spatial features between two strokes) is used to group the strokes.

Another strategy is *eager interpretation*, which is a real-time recognition solution. It tries to interpret each stroke as well as their structure during the composition, more precisely after each input stroke. By eager interpretation, it incrementally updates the recognized document according to the coming stroke and offers a real-time visual feedback to the users. Comparing to the lazy interpretation, eager interpretation gives a more naturally way for human-computer interaction that allows immediately validation and correction from users instead of waiting to the end. However, the difficulties are also obvious: not every stroke is meaningful for the nearby strokes when it is completed. A stroke could be a beginning part of a symbol and the full symbol has not yet been finished. On the one hand, the system need to be intelligent enough to identify the unfinished strokes and wait for more information to make the decision. On the other hand, the analysis process must be efficient to keep the use'pace that gives feedback as soon as possible. Eager interpretation still remains a complex and open problem that is rarely exploited.

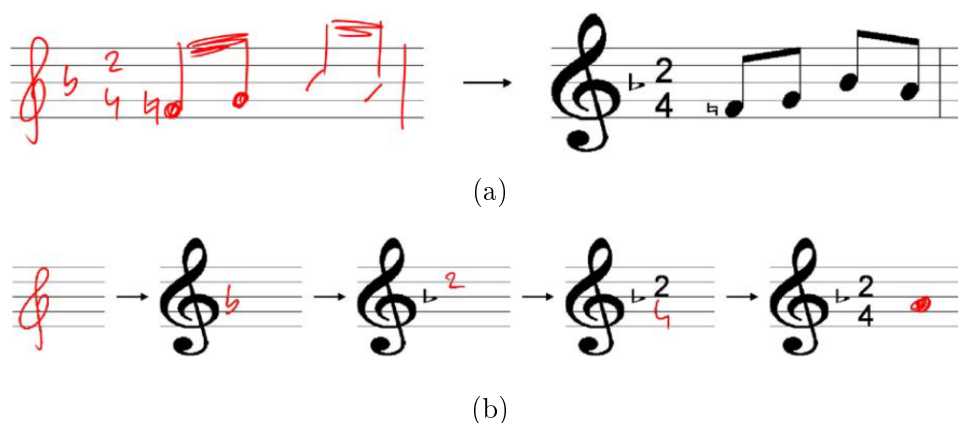


Figure 2.22: (a) Lazy interpretation of musical score. (b) Eager interpretation of musical score [MA09].

In our work, we go one step further on structure document composition. We are interesting in the diagram composed by multiple users, which has never been studied before. This project allow users to work together to complete a complex diagram by collaboration. We try to implement the eager interpretation to give a real-time feedback to the users. This effort will be introduced in chapter 5.

Chapter 3

Multi-touch Isolated Gesture Recognition

3.1 Introduction

Touch gesture interaction is considered as very natural in human computer interaction. It offers a more intuitive and convenient user experience than using a mouse and/or a keyboard. As a consequence, in recent years, an increasing number of efforts have been spent on enhancing computers' capabilities to better interpreting the user's gesture commands.

Generally, touch gesture is mainly used in two contexts when performing HCI in Human Computer Interaction (HCI): manipulation and command. The former is an online mode during which a system analyzes the spatial movements of fingers on the fly. Such gestures are mainly object oriented, i.e., two fingers pinching for zoom out, circling for rotation, etc. Users may receive a real-time feedback from the system during the movement of their fingers. The latter is an offline mode and the effect is evaluated after the gesture is completed. The recognition is achieved by analyzing the writing order and global shape of the trajectories as well as sketch-based interaction, e.g. text recognition. The system will then trigger a pre-defined command based on the recognized gesture. Currently, multi-touch gestures are mostly used as manipulation mode such as zooming, panning, rotating, etc. Usually, a local analysis of the gesture is performed based on the motion relationships between each finger during short time intervals [OIL11], as illustrated in previous chapter in Fig. 2.14. On the other hand, command interaction is generally achieved with mono-touch gestures. Such systems analyze the gesture's shape, speed, and writing direction to provide a global interpretation [MMM⁺12] [WNvGV09]

The problem is much more complex if a multi-touch gesture is used to execute a command operation. As stated in [SW13], where the term *sequential multitouch* is used to subsume multi-stroke and multi-touch gesture, a multi-touch gesture consists, by definition, of a variable number of finger trajectories and simultaneous touches. Spatial

and temporal relations between these trajectories, including synchronicity, are important properties to recognize different gestures. Such relations are useful to distinguish shapes with different writing orders. We described in chapter 2 some State-of-the-art methods [KHDA12] [LC02] [LLLW15] study these relations and their underlying semantics using individual segment instead of a global point of view. Some other multi-stroke sketch matching methods [FFJ11] [LLLW15] exploit the topology relations between sketch primitives and give a global interpretation but obviously the temporal relations are not concerned.

In this chapter, we present our effort for an online multi-touch gesture recognition system that effectively addresses the global interpretation issue. Two graph models will be introduced to extract the inner stroke spatial and temporal features. In section 3.2 we will detail our first graph model which is based on the features of static strokes relations. We combine these stroke relation features with classic geometric features, i.e. HBF49 [DA13], as a supplement to better characterize the multi-touch gesture. To test this graph model we have designed a new multi-touch gesture dataset with 18 gesture classes. Experimental results will be detailed and analyzed in section 3.2.4. Furthermore, to introduce more dynamic motion features into the graph model, we then present in section 3.3 the second graph model with motion based features. In this model we quantify the motion relations between strokes with numeric features instead of categorical features comparing to the first approach. A more sophisticated graph matching algorithm is proposed based on this new graph model in order to calculate the dissimilarity between graphs. To reflect the real challenges and complexities in multi-touch gesture domain, we update our gesture dataset and bring in more multi-touch gestures which have intricate inner stroke relations. We name this dataset as MTGSet (Multi-Touch Gesture dataset) and make this dataset freely available¹ to constitute a baseline benchmark for the multi-touch gesture recognition community. We present the experimental results in section 3.3.4.

3.2 Graph Modeling with Allen's Relations

We introduce here our first recognition system for multi-touch gestures. Fig. 3.1 shows the global framework of this recognition system. We will first discuss the graph modeling approach where the strokes and their spatial and temporal relations are respectively represented by vertices and edges in the graph. We then introduce a graph embedding strategy to encode the graph into a fixed size vector to feed a vector based classifier. An external feature extraction method is used as a complement for the global shape representation.

¹www-intuidoc.irisa.fr/category/bases-de-donnees

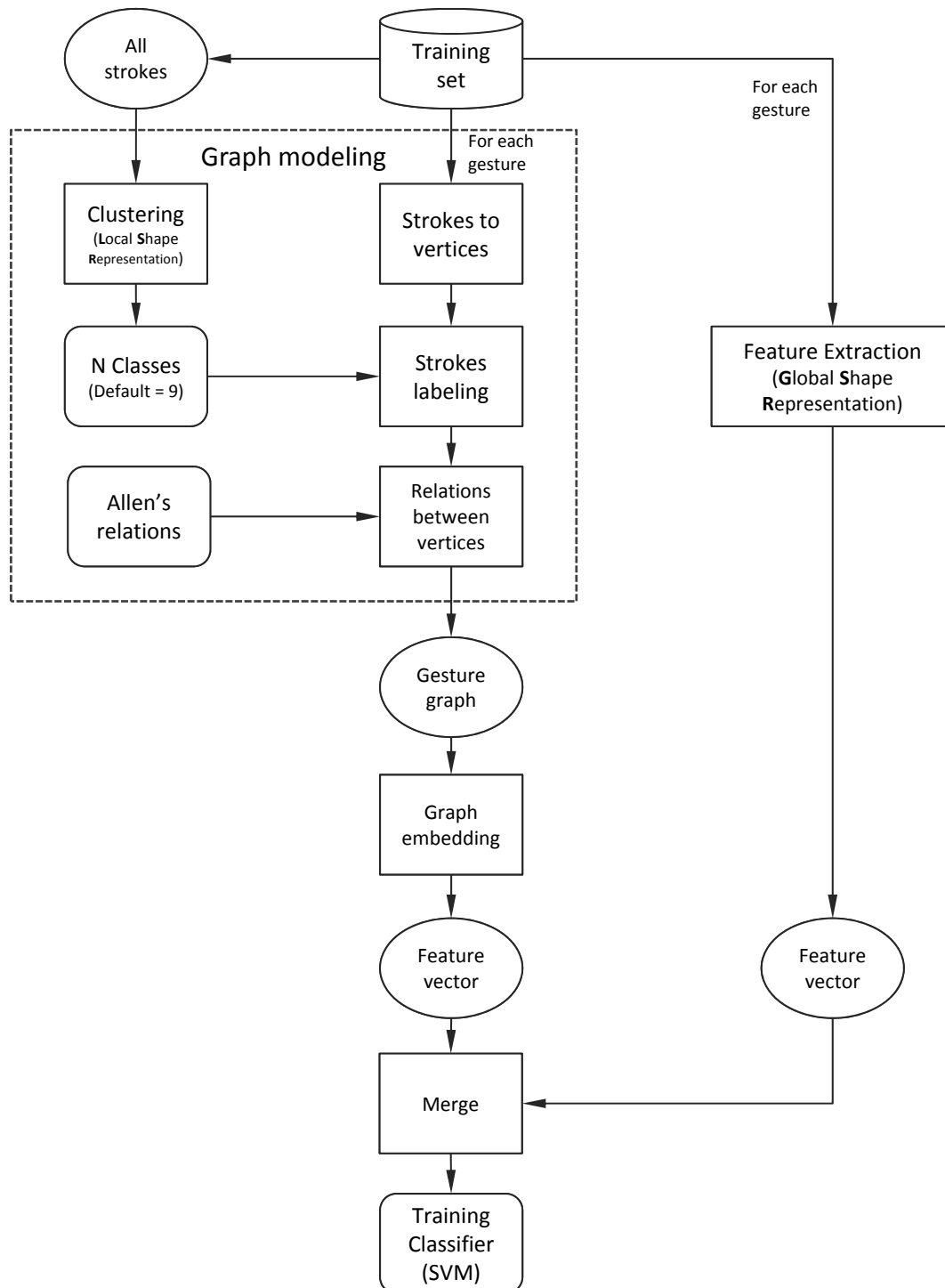


Figure 3.1: The graph modeling and classifier training architecture of gesture recognition system.

3.2.1 Graph Modeling

A multi-touch gesture modeling should consider three kinds of information: spatial, temporal and shape information. The spatial information shows the relative position of each single stroke with reference to the others inside the gesture, while a temporal information illustrates the written order or synchronization between the different strokes and the duration of each one. The last information should retain knowledge about the intrinsic shape of the stroke, allowing to distinguish for instance between a simple straight line from a more complex curve.

In a first step, each stroke is represented by three vertices as presented in Fig. 3.2, i.e. a begin vertex (V_b), a stroke vertex (V_s) and an end vertex (V_e). Note that the stroke is not explicitly segmented into three parts to match with the three vertices. These three vertices are used as reference points for the following stroke relation measurement.



Figure 3.2: A stroke is represented by three vertices in a graph. V_b : begin vertex; V_s : stroke vertex; V_e : end vertex.

Secondly, the shape information need to be integrated into the graph. With this first solution, to simplify the graph classification, we use a discrete codebook to encode each stroke as primitive. Here the primitive refers to the low level basic shape (e.g. line, arc, ellipse) which serves as the abstraction of a stroke. Unlike the other works [LLLW15] [AMG07] in which strokes are decomposed into pre-defined shape primitives, we implement a clustering method for all the strokes in the training set to define the code book. We extract the HBF49 features [DA13] to characterize each stroke and use the standard Euclidean distance based K-means method to achieve the clustering. Each stroke is then represented by the class of primitives. This class label is stored as an attribute on each stroke vertex V_s , denoted as $\Sigma_{V_s} = \{V_{c1}, \dots, V_{cn}\}$, where n is the number of classes. Since the shape information is measured on each individual stroke, this strategy is named as **Local Shape Representation (LSR)** in the following paragraph. Consequently, a more general denotation of the type of vertex is $\Sigma_V = \{V_b, V_e, V_{c1}, \dots, V_{cn}\}$, where V_b and V_e are specific for the extreme vertices.

Based on these vertex representation, we then measure the spatial and temporal relations between strokes. We make use of the Allen's relations [All83] which originally characterized the inferences about time by discrete labels. Fig. 3.3 shows the illustration and examples of Allen's relations. Such relations enumerate 7 relations between two events on one-dimension. Based on that, we respectively use Allen's relations on t -axis for measuring temporal relation and x, y -axis for spatial relation between two strokes, denoted

as $\Sigma_R = \{\Sigma_{R_x}, \Sigma_{R_y}, \Sigma_{R_t}\} = \{B_x, \dots, F_x, B_y, \dots, F_y, B_t, \dots, F_t\}$.

| Relation | Example | Relation | Example |
|-------------|--------------------------|------------|---------------------------|
| Before (B) | Str ₁ : _____ | During (D) | Str ₁ : _____ |
| | Str ₂ : _____ | | Str ₂ : _____ |
| Equal (E) | Str ₁ : _____ | Start (S) | Str ₁ : _____ |
| | Str ₂ : _____ | | Str ₂ : _____ |
| Meet (M) | Str ₁ : _____ | Finish (F) | Str ₁ : _____ |
| | Str ₂ : _____ | | Str ₂ : _____ |
| Overlap (O) | Str ₁ : _____ | | |
| | Str ₂ : _____ | | |

Figure 3.3: Examples of seven Allen's relations.

Considering a two strokes gesture as an example, Fig. 3.4 shows a general graph prototype with all potential relations that need to be measured. These relations are represented by edges as $E_s(x, y)$, $E_{st}(x, y, t)$ and $A_{st}(x, y, t)$ between two corresponding vertices in the graph. The definition of these notations is shown as follows.

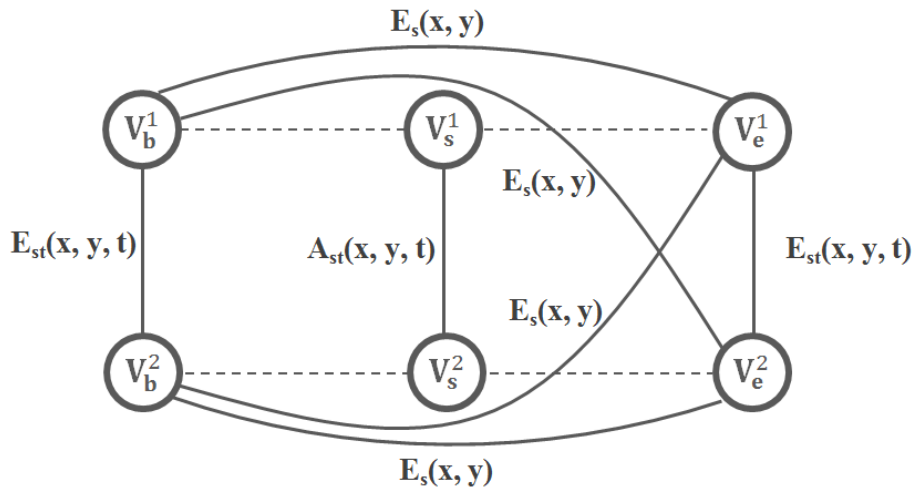


Figure 3.4: An example of a general graph modeling for a two strokes gesture.

1) **Edge between stroke vertices ($A_{st}(x, y, t)$):** $A_{st} \subseteq \Sigma_R$ means that the full set of Allen's relations are used to measure the relationships with respect to time, x-axis position and y-axis position between two stroke vertices V_s , where the subscript st means both spatial and temporal relation need to be measured. Note that since we measure the relation from time, x-axis and y-axis, the edge A_{st} has a set of three relations as its features.

2) **Edges between extreme vertices ($E_s(x, y)$ and $E_{st}(x, y, t)$):** Because the extreme vertices represent each of the finger-down or finger-up positions, only *Equal* time,

x-axis or y-axis relation is measured between extreme vertices. $E_{st}(x, y, t) \subseteq \{E_x, E_y, E_t\}$ is the edge which has a set of *Equal* time, x-axis or y-axis property as its features. It is used to represent the relation between begin vertices pair or end vertices pair. Unlike the edge between stroke vertices which has a fixed number of three relations, relations between extreme vertices may not satisfy any of an *Equal* property. Therefore, edge E_{st} can have a varied number of relations from 1 to 3. If none of an *Equal* property is satisfied, edge between corresponding extreme vertices would not be generated. The edge $E_s(x, y) \subseteq \{E_x, E_y\}$ is simplified from $E_{st}(x, y, t)$. It is used to represent the relation between a begin vertex and an end vertex. Obviously the *Equal* time is not satisfied in this case. It may also have a varied number of relations either 1 or 2, as defined for $E_{st}(x, y, t)$.

We note that the relations (edges) between extreme vertices service as a complementary measurement for Allen's relations. Even though some *Equal* relations between extreme vertices are also implied in Allen's relations between stroke vertices, we keep these redundant representation in the graph to ensure a rich structure layout. Otherwise, gestures which have a same number of strokes will always have a same graph structure.

In the following we will give two examples and their full graphs to intuitively demonstrate the graph modeling process.

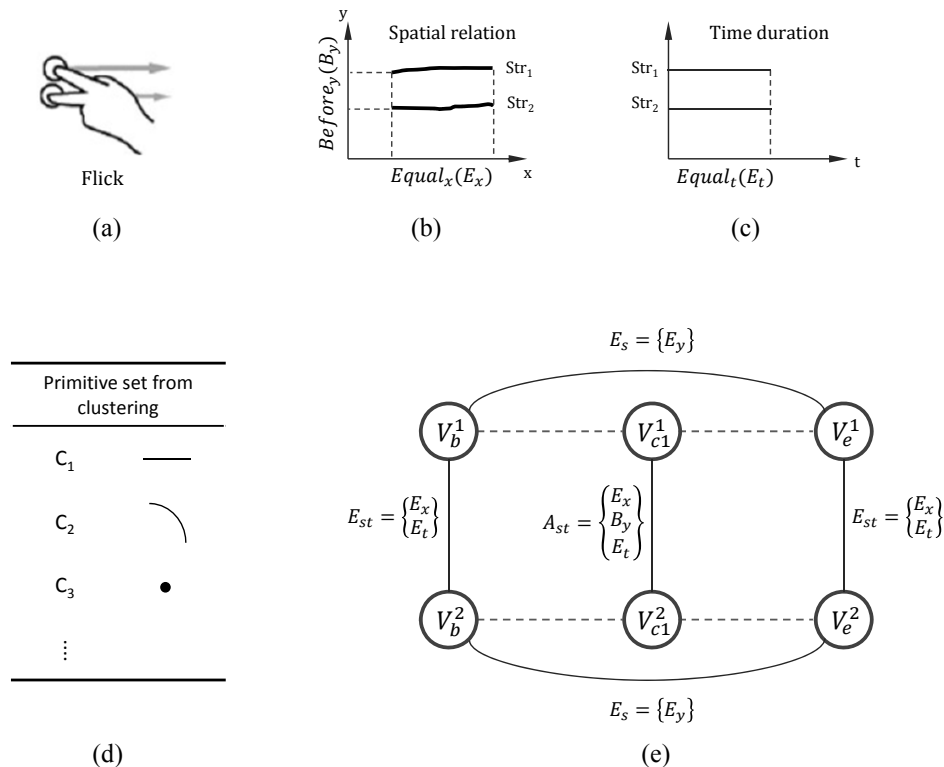


Figure 3.5: a) A flick gesture. b) Spatial relationship between strokes. c) Temporal relationship between strokes. d) Primitive set from clustering. e) Graph model with labels.

Fig. 3.5(a) shows a *Flick* gesture where two strokes are written simultaneously. The

spatial and temporal relations between strokes are depicted in 3.5(b) and (c). According to the Allen's relations, the attribute $Equal_x(E_x)$, $Before_y(B_y)$ and $Equal_t(E_t)$ are associated to the edge A_{st} as shown in Fig. 3.5(e). The edge $E_{st} = \{E_x, E_t\}$ between V_b^1 and V_b^2 indicates the two starting points are written in the same region on x-axis and in the same time. Same property will also be found between the two V_e vertices. The edge $E_s = \{E_y\}$ for $\{V_b^1, V_e^1\}$ pair and $\{V_b^2, V_e^2\}$ pair means that the extreme points are written in the same region on y-axis. Comparing to the general case in Fig. 3.4, the edges between V_b^1 and V_e^2 , V_b^2 and V_e^1 in Fig. 3.5(e) are removed because the corresponding two points are not located at the same position on neither x-axis nor y-axis. According to the primitive set shown in Fig. 3.5(d) (A real primitive set should be acquired from the clustering of all strokes. Here we use an artificial set to demonstrate the **LSR** process), both of the two strokes are classified to C_1 . Therefore, each stroke vertex V_s is set to V_{c1} .

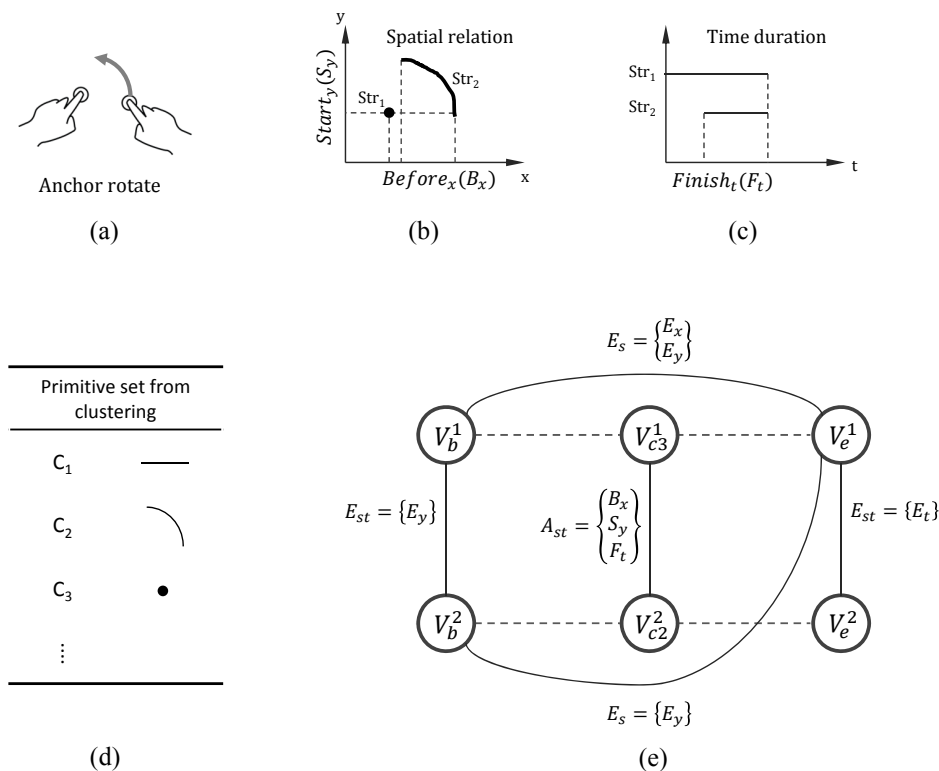


Figure 3.6: a) A anchor rotate gesture. b) Spatial relationship between strokes. c) Temporal relationship between strokes. d) Primitive set from clustering. e) Graph model with labels.

Fig. 3.6(a) gives another example of an *Anchor rotation* gesture. This gesture is written by two fingers following a semi-synchronous way in time domain (as shown in 3.6(c)). The corresponding full graph is shown in Fig. 3.6(e). Consequently, comparing to the *Flick* gesture above, the two graphs have different structure and attached attributes on edges and vertices. Therefore, our graph model retains rich structure and attribute information of multi-touch gesture and well capture the temporal relations between strokes.

It is discriminative and informative for multi-touch gesture recognition.

3.2.2 Graph Embedding

A graph embedding method aims to transform the graph structure into a feature vector for the benefit of using statistical classification method. In this paper, we adopt a graph embedding approach introduced by Sidere et al.[SHR09]. The basic idea of this approach is to build a matrix where each row is relative to a label of vertices or edges while each column corresponds to a sub structures P_j of the graph. The value of the matrix at $[L_i, P_j]$ is the number of occurrences of the label L_i in each sub graph P_k which is isomorphic to the sub structures P_j . The construction of the vectorial representation can then be performed by transforming the matrix into vector feature space. Fig. 3.7 gives a graph embedding example of the graph in 3.5(e).

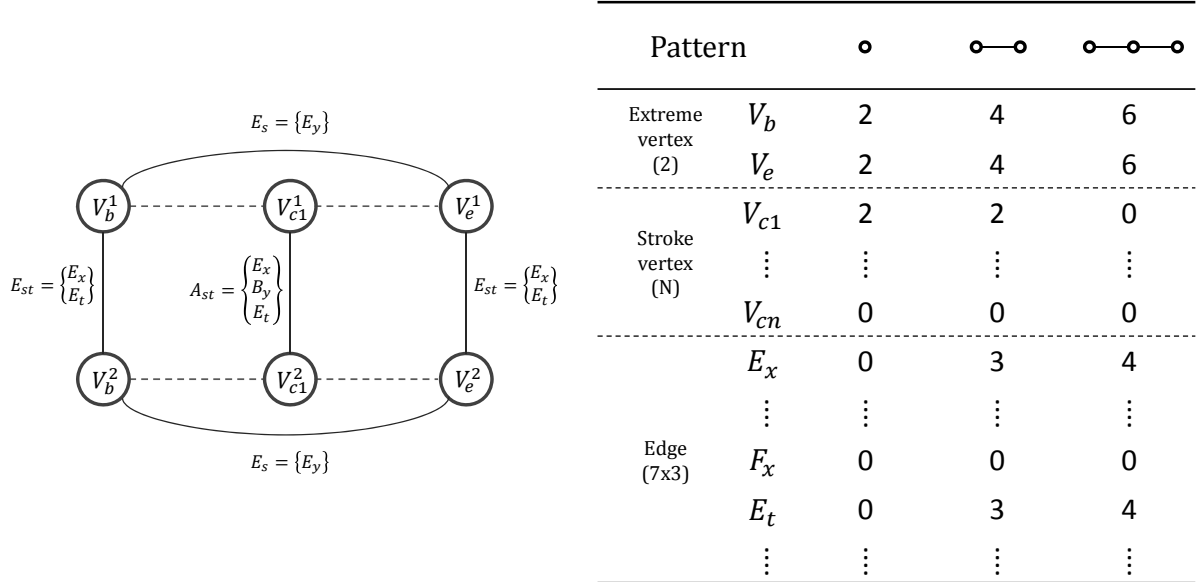


Figure 3.7: Vectorial representation of the graph in 3.5(e).

In our case, we empirically choose three sub structures which are one vertex, two vertices with one edge and three vertices with two edges for the column of the matrix. The row involves cluster number N (typically 9), labels of begin and end for the vertex, and the seven Allen's relations for three aspects (time, x-axis and y-axis). On the whole, 32 labels will be related to the rows. Accordingly, a graph feature vector with a length of 96 will be generated after the graph embedding.

3.2.3 Global Shape Representation (GSR)

The above **LSR** helps to integrate the stroke's shape information into the graph. It is generally believed that the global shape information of a full gesture is implied by the **LSR**

and graph topology. For our purpose, this global shape information need to be explicitly described and used for graph classification. To combine the global shape information and graph representation, we extract the HBF49 features on a full gesture and concatenate its feature vector to the graph feature vector achieved above. Note that the HBF49 is a static feature set for charactering mono-touch gesture. It has been proved to achieve a strong result on many mono-stroke or multi-stroke datasets. However, it doesn’t contain the temporal relation feature between strokes. Therefore, we make use of this feature set as a complementary description for our graph modeling and enhance the robustness of the recognition system. In the experimental section, we will compare the recognition result with and without this feature set.

3.2.4 Experiments

We conducted experimental evaluation of our proposed recognition system over a multi-touch gesture dataset. We will first introduce this multi-touch gesture dataset and then report the experimental results.

3.2.4.1 Dataset

There are few standard common dataset for multi-touch gesture recognition. This is because in a common sense multi-touch gestures are used for direct manipulation instead for inputing symbolic command. Therefore, we design a multi-touch gesture dataset which reflects the possibility of using multi-touch gestures for indirect symbolic commands. This dataset contains a total of 1,800 multi-touch gesture samples, written by 10 persons. There are 18 different multi-touch gestures which are initially designed for indirect command (Fig. 3.8). These gestures are composed of points, linear segments and arcs with a varying number of strokes. Note that most of them have an apparent distinction according to their shape except two pairs, *Command C-1* versus *Command C-2* and *Flick* versus *Flick-twice*. In gesture *C-1*, user performs firstly the ‘dot’ on the left side, then the ‘C’ shape on the right side. On the contrary, in gesture *C-2* the ‘dot’ is done after the ‘C’ shape. For the second pair of gesture, the difference is not in the sequence but in the synchronization : the *Flick* gesture is performed with two fingers flicking simultaneously, whereas the *Flick-twice* gesture is done by one finger but flicking twice. These two pairs of gestures have the same shape respectively but are different from a temporal point of view. We introduce them in the dataset to reflect the real challenge of multi-touch gesture characterize and evaluate the capability of our recognition method to discriminate gestures from temporal information.

Fig. 3.9 gives a screen shot of the data acquisition tool we used. Note that we record some basic information (Fig. 3.9(a)) of each user i.e. *sex*, *age* and *handedness* (The *name* is erased due to a law.....). Although these information are not used in our experiment,

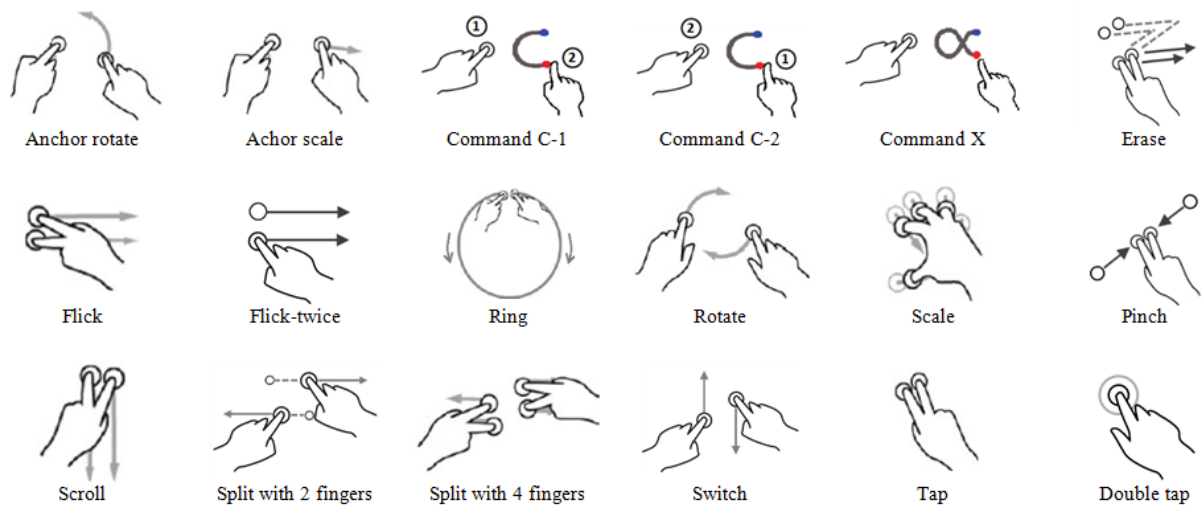


Figure 3.8: Multi-touch gestures prototypes in our experimental dataset.

we believe that they can be useful for a future work for human behavior analyzing. Since multi-touch gesture have complex dynamic variations, different gestures may have a same appearance but differ from writing order. Therefore, we give an animation of each gesture to explicitly show the writing order of touches instead of using a static image (Fig. 3.9(b)).

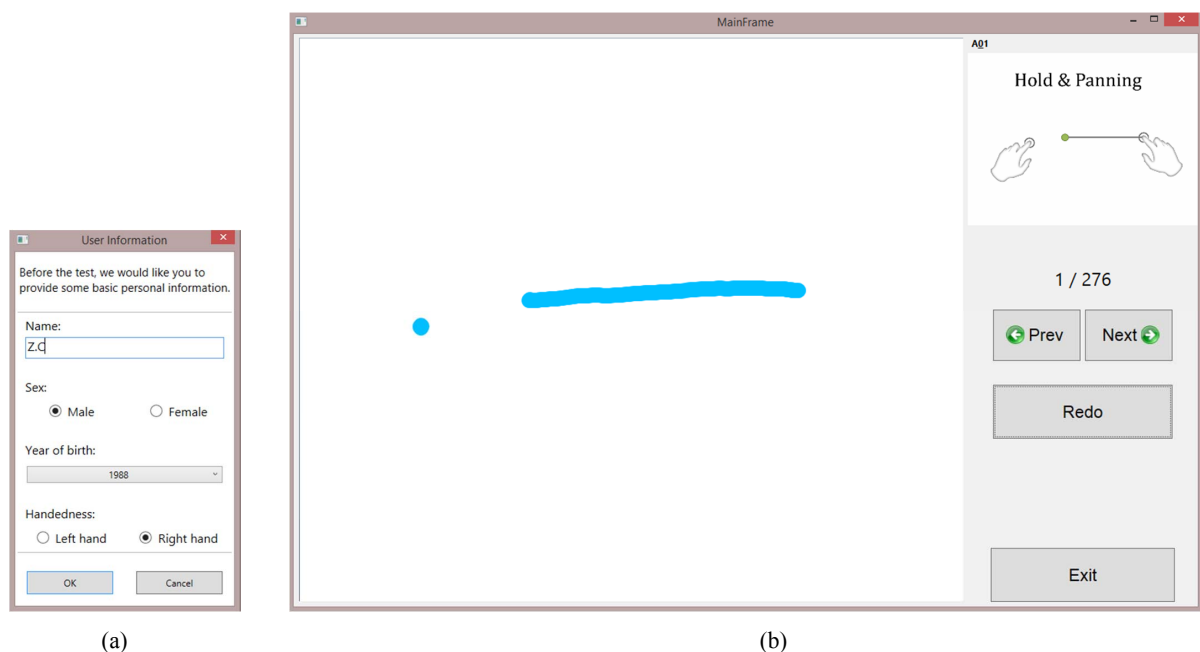


Figure 3.9: (a) We record some basic information from the users. (b) Data acquisition tool. In the top right, a animation is used to show the groundtruth of a multi-touch gesture.

3.2.4.2 Results

Since the graph features have been embedded in a vector features, we chose LIBSVM as the classifier. The LIBSVM with a Gaussian kernel and default parameters appeared to provide adequate capability for gesture classification.

From the architecture of our gesture recognition system shown in Fig. 4.4, the graph modeling procedure can be roughly segmented into three modules: 1. Basic graph (Strokes to vertices and Relations between vertices); 2. **LSR** for stroke labeling; 3. **GSR** by feature extraction. To fully evaluate the importance and influence of these modules, five experiments were conducted with different modules or module combinations respectively.

Experiment 1 (Basic Graph): this experiment uses basic graph to characterize the multi-touch gesture without **LSR** and **GSR**. Only spatial and temporal relations between strokes were integrated in the graph as features. It is regarded as a baseline system for comparison.

Experiment 2 (GSR): the HBF49 has been tested on different mono-touch gesture datasets and proven to be a powerful feature set. In this strategy, we directly use HBF49 on our multi-touch gesture dataset without graph modeling and embedding to test its competence of characterizing multi-touch gesture. In other words, this experiments takes into account only the shapes of gestures.

Experiment 3 (Graph+GSR): in this trial, we combine the above two representations. The feature vector of GSR is concatenated after the graph features from the first experiment to achieve a conjoint feature vector.

Experiment 4 (Graph(LSR)): the fourth experiment evaluates the performance of integrating the **LSR** into the graph. As explained in section 3.2.1, the **LSR** is based on a non supervised clustering to quantify the stroke shapes, the optimal number of cluster K is set experimentally to 9. The comparison of different values of K will be presented below.

Experiment 5 (Graph(LSR)+GSR): finally, we combine the graph with both global and local shape information. It was achieved by concatenating the feature vector of **GSR** after the vector obtained in the fourth experiment. This experiment correspond to the full recognition procedure as shown in Fig. 3.1.

All the experiments adopted a 5-cross-validation (CV) scheme for testing writer-independent (WI) performance.

Fig. 3.10 summarizes the recognition rates obtained by different methods or their combination. The results show that the graph strategy, containing only spatial and temporal information, obtains 87.50% recognition rate which is lower than the HBF49 based GSR method (90.44 %). With a deeper investigation from the confusion matrix in Fig. 3.11, most of the mis-classifications by graph strategy happens between the gestures *Command C-1*, *Command C-2* and *Command X* because they have similar spatial relationships between strokes (Fig. 3.11(a)). We can note that **GSR** is able to classify the majority of

| Method | Length of features | Accuracy rate (%) | Std. Deviation |
|----------------|--------------------|-------------------|----------------|
| Graph | 87 | 87.50 | 0.037 |
| GSR | 49 | 90.44 | 0.034 |
| Graph + GSR | 136 | 90.11 | 0.035 |
| Graph(LSR) | 96 | 92.56 | 0.013 |
| Graph(LSR)+GSR | 145 | 94.50 | 0.020 |

Figure 3.10: Recognition rate obtained by different modules.

multi-touch but fails to make a distinction between **Command C-1** versus **Command C-2** and **Flick** versus **Flick-twice** since they are similar in shape, respectively (Fig. 3.11(b)).

The Graph+GSR method results in 90.11% which is slightly lower than Experiment 2. However, when we integrate the shape information of each stroke inside the graph by using clustering method, the recognition rate of Graph (LSR) rises to 92.56%. In accordance to what can be expected, the final experiment, Graph (LSR) + GSR which integrates all the information together achieves the best recognition rate, 94.50%, that is significantly better than others. Meanwhile, by evaluating the standard deviation of 5-cross-validation, the two strategies with **LSR** inside produce smaller variations than the other three strategies in the writer independent situation.

The cluster number K used in the clustering process is also a factor of great concern for the recognition. The comparison results of the recognition rate under different values of K are illustrated in Fig. 3.12.

It shows that the same trend in the relationship between cluster number K and recognition rate can be observed from both methods. The peak appears when K is chosen to 9. Neither too large nor too small cluster number are able to well perform. The comparison also proves that the Graph (LSR) + method is always better than the Graph (LSR) method.

We have investigated a method to recognize the multi-touch gestures. Unlike many other works, we study this problem from a new perspective considering a multi-touch gesture as an indirect command. We believe that three kinds of information, spatial, temporal and shape information, of the gesture should be processed for the recognition. We first proposed a graph modeling method which measures the spatial and temporal relationships between strokes of the gesture. In order to integrate the shape information into the graph, a clustering method is employed to label the shape of the stroke inside the graph as a local shape feature. Another globe shape feature is extracted with our previous baseline method HBF49 features. Using global shape information extracted through HBF49 features allows to improve these results.

| | C1 | C2 | X | F | F-2 |
|-----|----|----|----|-----------|-----------|
| C1 | 34 | 17 | 35 | 0 | 0 |
| C2 | 6 | 77 | 15 | 0 | 0 |
| X | 16 | 19 | 60 | 0 | 0 |
| F | 0 | 0 | 0 | 99 | 1 |
| F-2 | 0 | 0 | 0 | 3 | 96 |

(a) Graph (Without shape feature)

| | C1 | C2 | X | F | F-2 |
|-----|-----------|----|-----------|----|-----|
| C1 | 99 | 1 | 0 | 0 | 0 |
| C2 | 36 | 63 | 0 | 0 | 1 |
| X | 0 | 2 | 98 | 0 | 0 |
| F | 0 | 0 | 0 | 82 | 18 |
| F-2 | 0 | 0 | 1 | 42 | 56 |

(b) GSF

| | C1 | C2 | X | F | F-2 |
|-----|----|----|----|-----------|-----------|
| C1 | 49 | 17 | 34 | 0 | 0 |
| C2 | 2 | 82 | 16 | 0 | 0 |
| X | 16 | 12 | 72 | 0 | 0 |
| F | 0 | 0 | 0 | 92 | 8 |
| F-2 | 1 | 0 | 0 | 3 | 95 |

(c) Graph + GSF

| | C1 | C2 | X | F | F-2 |
|-----|----|----|----|-----------|-----------|
| C1 | 64 | 24 | 10 | 0 | 0 |
| C2 | 29 | 68 | 0 | 0 | 1 |
| X | 2 | 4 | 81 | 0 | 0 |
| F | 0 | 0 | 0 | 99 | 1 |
| F-2 | 0 | 0 | 1 | 2 | 96 |

(d) Graph (LSF)

| | C1 | C2 | X | F | F-2 |
|-----|----|----|----|------------|-----------|
| C1 | 75 | 14 | 8 | 0 | 0 |
| C2 | 21 | 77 | 1 | 0 | 0 |
| X | 2 | 0 | 88 | 0 | 0 |
| F | 0 | 0 | 0 | 100 | 0 |
| F-2 | 1 | 0 | 0 | 3 | 95 |

(e) Graph (LSF) + GSF

C1: CommandC-1 C2: CommandC-2 X: CommandX
F: Flick F-2: Flick-twice

Figure 3.11: Confusion matrix of some typical misclassified gestures of different classification methods. The row relates to the ground truth.

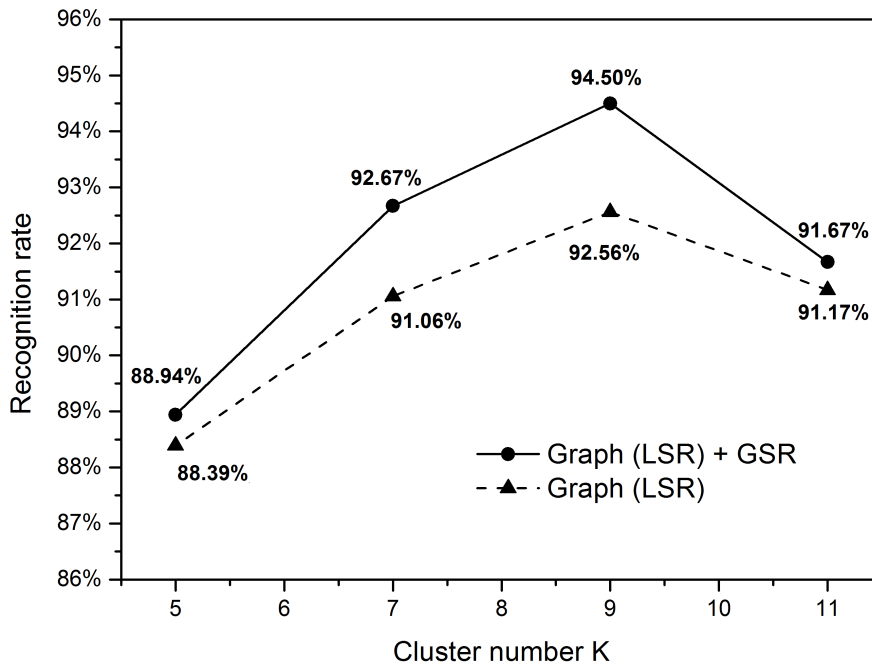


Figure 3.12: Performance evaluation of different values of cluster number K used for LSR.

However, in this model we characterize the inner stroke features by simply discrete and static relations. To give a more precise measurement for spatial and temporal inner stroke relations, we explore to quantify these relations and use numerical features to better describe a multi-touch gesture. A more complex graph model will be introduced in next section.

3.3 Graph Modeling with Motion Based Features

The previous graph modeling measures spatial and temporal relations mainly according to the strokes' pen-up and pen-down points. This modeling doesn't explicitly describe the relations on the middle part of the strokes. Meanwhile, the modeling uses only symbolic descriptions (Allen's relations and stroke shape labels) as features in the graph. These discrete descriptions are not precise enough to quantify the relation and shape features. To rectify the shortcoming of the modeling, in this section we present a new graph model embedding scalar attributes which allows a more smooth representation. Furthermore, we segment each stroke into small substrokes so that the relation measurement can be allowed on every part of a stroke. Fig. 3.13 gives an overview of our second multi-touch gesture analyzing system. Firstly the finger trajectories will be broken down into substrokes to decrease the redundancy and computation complexity for graph matching. This step will be described in section 3.3.1. We then introduce the new gesture graph in section 3.3.2 to characterize the multi-touch gesture. A set of static shape features is extracted for each

substroke. Furthermore, spatial and temporal relations between substrokes are identified to model their relative motion and position. All these information are integrated into the gesture graph. We then, in section 3.3.3 introduce the graph matching algorithm, based on graph edit distance involving dynamic time warping (DTW) algorithm, to measure the similarity between two graphs. A new distance based graph embedding algorithm is finally presented to transform the graph into feature vector for classification.

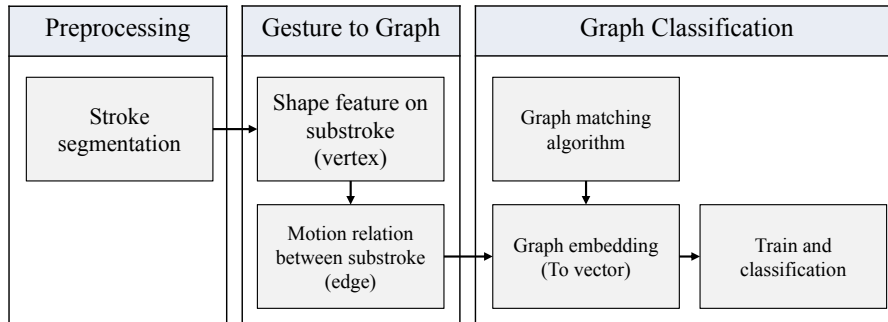


Figure 3.13: Overview of the graph based gesture recognition in three stages.

3.3.1 Preprocessing and Stroke Segmentation

An on-line handwriting signal is a sequence of points recording the coordinates (x, y) , and time stamp (t) for each input event. For the study of a multi-touch gesture, the most important feature is the dynamic motion relations between its strokes such as closing, spreading, etc. Because these dynamic motion relations changes over time during the writing, it is hard to describe the motion relations between two full strokes. On the other hand, extracting motion relation based on each point-pair is not necessary since it will generate much more redundant information and will significantly increase the computation cost. Therefore, one solution is to segment the strokes into substrokes and use substrokes as the basic elements to build the graph model. The segmentation is performed by point resampling with a pre-defined spatial step/length Δ . The term “substroke” is defined as the link between two consecutive re-sampled points. Instead of using a linear interpolation algorithm as in [WWL07] we choose to keep the original points available on the trajectory, as long as they fulfill the space displacement constraints. So we can preserve accurate time information, but the length of each segment may not exactly be equal to Δ . Note that we preserve at least the first and last point, a stroke will generate at least one substroke even if a stroke is just one point. The algorithm is detailed in **Algorithm 1** and depicted in 3.14.

At the end of this step, each original stroke is substitute with new re-sampled segments, which are regarded as its substrokes. We will study the spatial and temporal relation for every substroke pair in the graph model.

Algorithm 1 Stroke resampling in a length of Δ **Input:** original trace $P_{org}(p_0, \dots, p_n)$ **Output:** new trace $P_{new}(p_0, \dots, p_m)$

```

1:  $P_{new} \leftarrow p_0$ 
2:  $D \leftarrow 0$ 
3: for all  $p_i (i \geq 1)$  in  $P_{org}$  do
4:    $d \leftarrow \text{Distance}(p_{i-1}, p_i)$ 
5:   if  $(D + d) \geq \Delta$  then
6:      $(P_{new} \leftarrow p_i)$ 
7:      $D \leftarrow 0$ 
8:   else
9:      $D \leftarrow D + d$ 
10:  end if
11: end for
12:  $P_{new} \leftarrow p_n$ 
13: return  $P_{new}$ 

```

3.3.2 Gesture to Graph

Basically, a gesture symbol is represented as a set of substrokes computed by algorithm 1 described in the previous section. In this section we will extract two important information : the shape of each substroke and topology relations for each substroke pair. The information and the substroke set are represented by a graph.

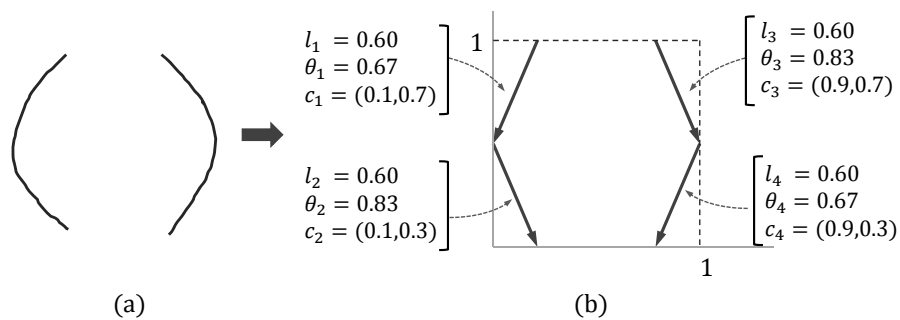
3.3.2.1 Geometry feature

Figure 3.14: Stroke segmentation and substroke representation. (a) a raw bracket like gesture; (b) the gesture is segmented into four substrokes and normalized inside a unit square bounding box. Each substroke has a feature vector composed of its length (l), angle (θ) and centroid (c).

We characterize each substroke by four geometric features including length (l), angle

(θ) and its centroid coordinates (c_x, c_y) .

$$F = (l, \theta, c_x, c_y) \quad (3.1)$$

The angle θ is measured using the starting and ending points of the stroke and normalized between 0 and 1, i.e. $\theta = \frac{1}{2\pi} \tan^{-1}((y_1 - y_0)/(x_1 - x_0))$. Fig. 3.14 illustrates an angle bracket like gesture. It composes of two trajectories decomposed into two substrokes for each. All features are equally weighted by being measured under the unity bounding box to have the same difference between their possible maximum and minimum values.

3.3.2.2 Topology relation

Unlike a mono-touch gesture, whose strokes are written in sequence and delimited by a finger (stylus)-up event, the strokes in a multi-touch gesture could have complex synchronization relations. Fig. 3.15 illustrates a possible temporal behavior of a set of 3 strokes.

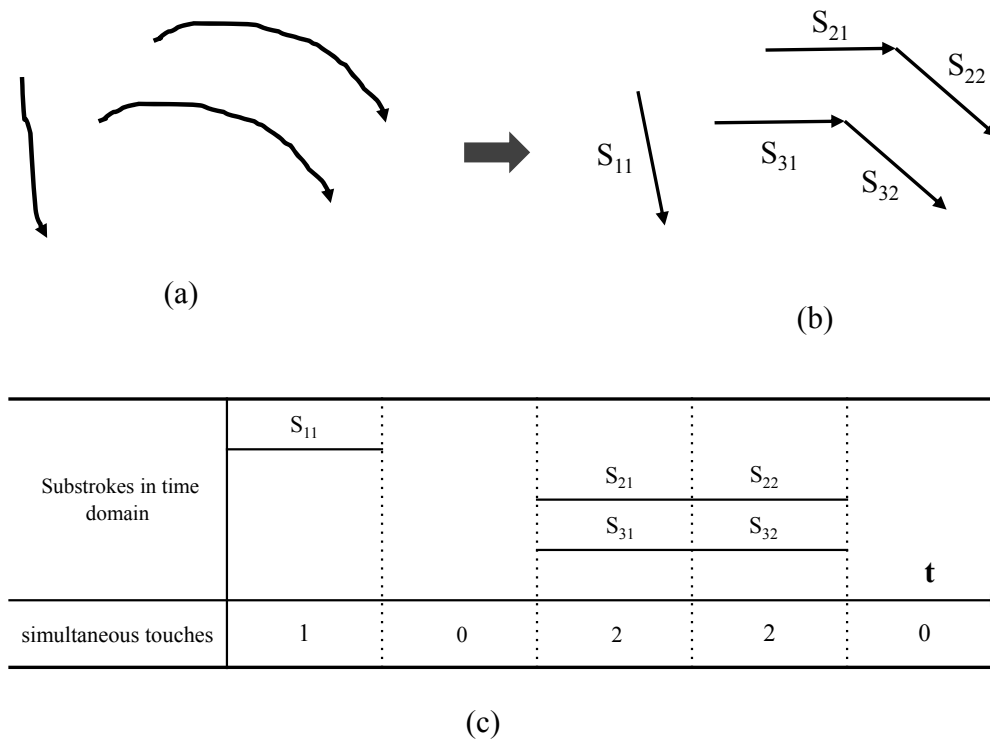


Figure 3.15: Example of temporal activity of substrokes in a multi-touch interaction. (a) A three strokes gesture. (b) The substroke representation, S_{ij} indicates the j th substroke of stroke i . (c) Temporal activity of substrokes.

To represent the comprehensive temporal progression and relative movement of sub-strokes, we define three types of relations between two sub-strokes: adjacent(a), synchronous(s) and consecutive(c), denoted as $\Sigma_R = \{R_a, R_s, R_c\}$.

Relation 1 (Adjacent, R_a) . Substrokes S_{ij} and S_{kl} are adjacent when they belong to the same stroke and are consecutive in time.

$$R_a(S_{ij}, S_{kl}) = 1 \quad \text{iff } (k = i \text{ and } l = j + 1) \quad (3.2)$$

With the example of Fig. 3.15 (c), we have: $R_a(S_{21}, S_{22}) = 1$; $R_a(S_{31}, S_{32}) = 1$; for all other pairs $R_a(S_{ij}, S_{kl}) = 0$. This relation will preserve the sequential information of the substrokes belonging to the same stroke in the model.

Relation 2 (Synchronous, R_s) . Two substrokes S_{ij} and S_{kl} are synchronous when they belong to different strokes and are written at the same time.

This relation indicates that the two substrokes are written by two fingers in a synchronous manner which is a typical property for a multi-touch gesture. Since two substrokes cannot have precisely the same starting and ending times, we compute the degree of synchronicity t_s to set the R_s relation:

$$R_s(S_{ij}, S_{kl}) = 1 \quad \text{iff } (t_s(S_{ij}, S_{kl}) > t_\lambda), \quad (3.3)$$

where

$$t_s(S_{ij}, S_{kl}) = \frac{t_o(S_{ij}, S_{kl})}{\min(t(S_{ij}), t(S_{kl}))}. \quad (3.4)$$

t_o is the overlapping time of two substrokes, and $t(S_{ij})$ is the duration of the corresponding substroke. With Fig. 3.15(c), we can evaluate $t_s(S_{21}, S_{31}) = t_s(S_{22}, S_{32}) = 1$;

For the synchronous relation, we associate a feature vector as the attribute to describe the relative motion between these substroke pair. A popular technique [OIL11] for encoding the motion of two fingers is based on the relative motion of the starting and ending points of each substrokes. We use here translation, scaling and rotation motions (m_t, m_s, m_r , respectively) to encode the relative movement of two synchronous substrokes. Fig. 3.16 illustrates the definition of these three motions.

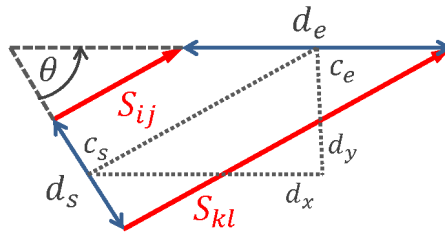


Figure 3.16: Motion features of two synchronous substrokes S_{ij}, S_{kl} . Translation motions (d_x, d_y) are based on c_s and c_e , the centroids of starting point pair and ending point pair, respectively. Scaling motions (d_s, d_e) are the distance of starting and ending point pairs, respectively. Rotation motion is the θ from the starting point pair to the ending point pair.

Based on the features defined in Fig. 3.16, a motion feature vector can be computed as follows:

$$M(S_{ij}, S_{kl}) = \begin{pmatrix} m_t \\ m_s \\ m_r \end{pmatrix} = \begin{pmatrix} \sqrt{dx^2 + dy^2} \\ d_e - d_s \\ \theta/2\pi \end{pmatrix} \quad (3.5)$$

This feature vector will be weighted with the degree of synchronicity t_s so that the attribute w_s of a synchronous relation is valued by

$$w_s(R_s) = w_s(S_{ij}, S_{kl}) = t_s(S_{ij}, S_{kl}) \cdot M(S_{ij}, S_{kl}) \quad (3.6)$$

Relation 3 (Consecutive, R_c) . Strokes S_i and S_k are written in sequence when there is no synchronous relation between any of their substrokes S_{ij} and S_{kl} .

This relation captures sequential dependencies between two strokes. Since we model the relations at the substroke level, this relation is built only from the last substroke of the first stroke to the first substroke of the successive one. To measure the time elapse between two substrokes, an attribute w_c is computed relating to the time delay t_d between the two substrokes

$$w_c(R_c) = w_c(S_{ij}, S_{kl}) = t_d/T \quad (3.7)$$

This feature is normalized by the duration of the gesture T so that it is scaled to $[0,1]$. Considering Fig. 3.15, the following substrokes are connected with a R_c relation-ships: (S_{11}, S_{21}) ; (S_{11}, S_{31}) .

Consequently, the full substroke relations of the example Fig. 3.15 are shown in 3.17 (a). The corresponding adjacency matrix is given in Fig. 3.17 (b).

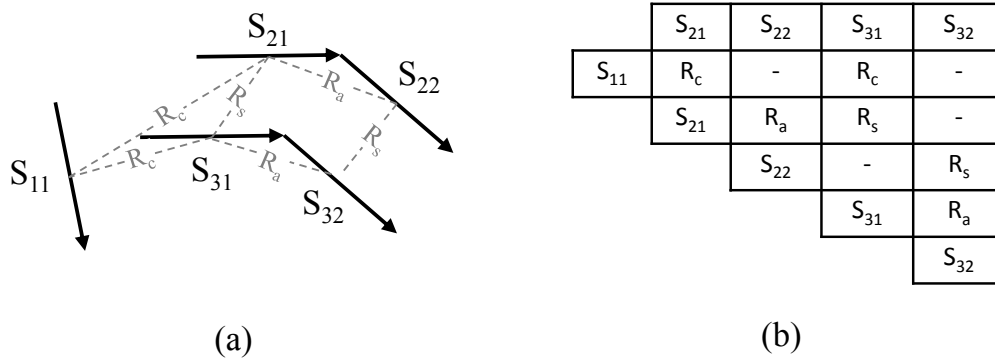


Figure 3.17: (a) Illustration of the substroke relationships from example of Fig. 3.15. (b) The corresponding adjacency matrix.

3.3.2.3 Graph definition

Based on the geometry feature and topology relation we introduced above, we propose a *Multi-Touch-Stroke Graph* (MTSG) representation to integrate all the information as follow,

Definition 1 (Multi-touch-stroke graph). A MTSG is a four-element tuple $G = (V, E, \mu, \omega)$.

- (1) vertex V : each substroke is represented by a vertex v .
- (2) edge $E \subseteq V \times V$: each edge $E(v_i, v_j)$ connects a pair of vertices (v_i, v_j) that has a certain substroke relation R_a, R_s, R_c .
- (3) attribute function μ on vertex: the attribute of a vertex is the geometry feature of the substroke $\mu(v_i) = F(v_i)$.
- (4) attribute function w on edge: according to the type of relationship that defines the edge we associate the following attribute:

$$w_a(v_i, v_j) = 1, \text{ when } R = R_a$$

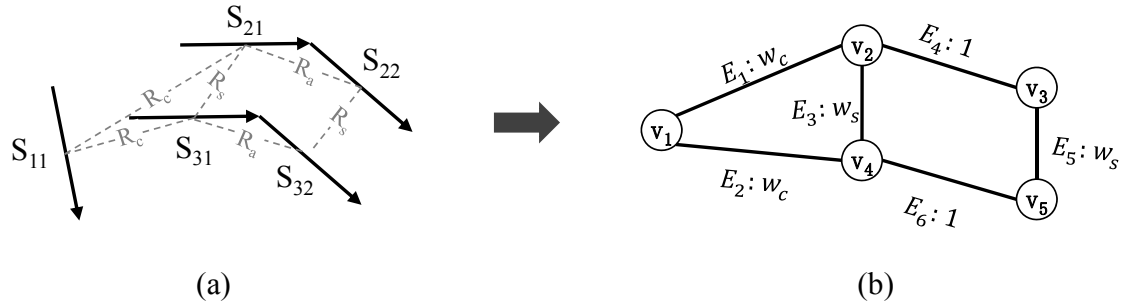
$$w_s(v_i, v_j) = t_s \cdot M_s, \text{ when } E(v_i, v_j) = R_s$$

$$w_c(v_i, v_j) = t_d/T, \text{ when } E(v_i, v_j) = R_c$$

Using the above graph definition, the gesture in 3.15(a) is represented in the form of weighted graph, which characterize both geometric and topological features as shown in Fig. 3.18(b). The comparison of two gestures is transformed into a graph matching problem.

Another example is shown in Fig. 3.19. In this example, we show that gestures which have a same appearance may generate different graphs. If the two strokes are written in a synchronous manner, the corresponding graph is shown in Fig. 3.19(b). In another case, if two strokes are written in sequence, the graph representation is shown in Fig. 3.19(c). It proves that our graph modeling can well capture the stroke temporal relations and give out different representation even if gestures have a same appearance. It is discriminative for gesture recognition.

While many other sketching matching works [LLLW15] [FFJ11] [LC02] use more comprehensive topology relation to capture the geometry or structural information between strokes such as Cross, Half-Cross, Parallelism, our definition focuses more on their temporal and motion relation which contains the main semantic intention of a multi-touch gesture. Meanwhile, a gesture which performed by multiple fingers would not have a complex structural relation between strokes. Therefore, our graph preserves the position and geometry information of each substroke on vertex and uses edge to describe their temporal relations which has never been done before. It is discriminative and informative for multi-touch gesture matching.



| Vertex | | |
|------------------------------------|--|-----------------------------|
| (v_1, \dots, v_5) | $F(v_1) = (0.25, 0.8, 0.1, 0.8)$ | Defined by Equ. 3.1 |
| | \vdots | |
| Edge | | |
| Adjacent edge (E_4, E_6) | $E_4 = w_a(v_2, v_3) = 1;$ $E_6 = w_a(v_4, v_5) = 1;$ | Defined by Equ. 3.2 |
| Synchronous edge (E_3, E_5) | $E_3 = w_s(v_2, v_4) = (0.25, 0, 0);$ $E_5 = w_s(v_3, v_5) = (0.25, 0, 0);$ | Defined by Equ. 3.4 and 3.5 |
| Consecutive edge (E_1, E_2) | $E_1 = w_c(v_1, v_2) = 0.25;$ $E_2 = w_c(v_1, v_4) = 0.25;$ | Defined by Equ. 3.7 |

(c)

Figure 3.18: (a) Substrokes and their relations as depicted in Fig. 3.17(a). (b) The graph representation of the gesture. (c) The attributes associated to the vertices and edges.

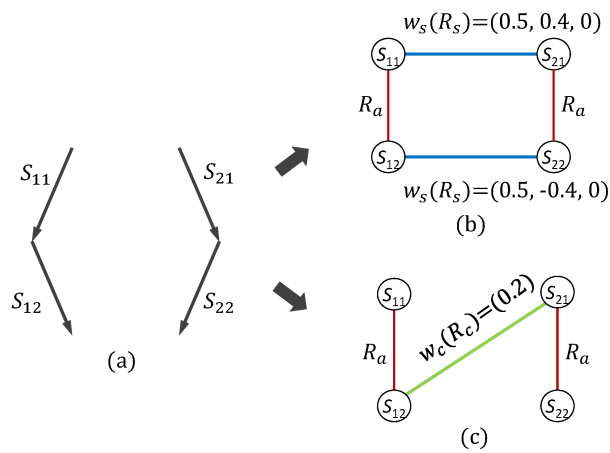


Figure 3.19: Two graph representations of a gesture. (a) The original gesture which is also shown in Fig. 3.14. (b) The graph representation if two strokes are written in a synchronous manner, i.e. (s_{11} synchronizes to s_{21} , s_{12} synchronizes to s_{22}). (c) The graph representation if the stroke s_{21}, s_{22} is written after the stroke s_{11}, s_{12} .

3.3.3 Graph matching and classification

The work presented gives a full graph representation for multi-touch gesture. Due to the fact that gesture may contain different number of strokes, it is difficult to extract a fixed length global feature set for classification. Therefore we propose to measure the similarity between two graphs to achieve the graph classification. A traditional way is to calculate the graph edit distance for the difference between two graph. Generally, the problem of searching the optimal graph matching and the corresponding edit operations is known to be NP-hard. But since we preserve the vertices adjacent relations, the vertices which connected with adjacent edge can be treated as a subgraph. Therefore, instead of performing a global matching algorithm between the two entire graphs such as in [RB09] [RNB07], we firstly find the optimal subgraph matching between two gesture graphs and then extend the optimization at the global level. Note that the subgraph matching is achieved only according to vertex matching, the edge matching is then implied by the alignment of the vertices.

3.3.3.1 Subgraph matching for stroke comparison

As we denoted in section 3.3.2.2, the adjacent relation/edge indicates that the two connected vertices belong to a same original stroke. These vertices can be regarded as a subgraph vertices set. Given two subgraphs $V_s = (v_1, \dots, v_n)$ and $U_s = (u_1, \dots, u_m)$. We use DTW algorithm to compute the minimal matching cost of the vertices, the minimal matching cost is defined as

$$c(V_s, U_s) = D_\Phi[d](V_s, U_s) = \frac{1}{N} \sum_{n=1}^N d(v_{\phi_v(n)}, u_{\phi_u(n)}) \quad (3.8)$$

where the DTW distance $D_\Phi[d](V_s, U_s)$ is the alignment distance according to the Viterbi path defined as

$$\Phi = (\phi(1), \dots, \phi(N)), \quad (3.9)$$

with the alignment pair

$$\phi = (\phi_v, \phi_u) : 1, \dots, N \rightarrow \{v_1, \dots, v_n\} \times \{u_1, \dots, u_m\}, \quad (3.10)$$

which denotes an alignment of vertex pair in corresponding regions in V_s and U_s . The DTW finds the optimal alignment path in the sense that it minimizes the distance between two sequences. The local distance function d of two vertices is the Euclidean distance of their feature vectors F as described in section 3.3.2.1.

We may easily deduce the vertex matching from the Viterbi path Φ . For two subgraphs which contain different number of vertices, three standard edit operations are considered in order to compute the edit cost. We denote the substitution of two elements u and v by

$(u \rightarrow v)$, the deletion of element u by $(u \rightarrow \varepsilon)$ and the insertion of element v by $(\varepsilon \rightarrow v)$. Note that in the Viterbi path Φ a vertex can be aligned to multiple vertices under the DTW procedure. In this case, we keep the closest two vertices as matching pair and leave the remaining of multiple matched vertices as delete/insert vertices. Thus, we denote the vertex matching of two subgraphs as Φ^* with

$$\phi^* = \begin{cases} (\phi_v^* \rightarrow \phi_u^*) \text{ for } V_c^* \times U_c^* \\ (\phi_v^* \rightarrow \varepsilon) \text{ for } V_c \setminus V_c^* \\ (\varepsilon \rightarrow \phi_u^*) \text{ for } U_c \setminus U_c^* \end{cases} \quad (3.11)$$

where $(\phi_v^* \rightarrow \phi_u^*)$ is the set of matching pair deduced from Φ . In this set, each vertex in V_c^* is uniquely matched to a vertex in U_c^* , vice versa. The remained non-matched vertices in V_c and U_c will be regarded as deletion $(\phi_v^* \rightarrow \varepsilon)$ and insertion $(\varepsilon \rightarrow \phi_u^*)$ vertices, respectively.

We then deduce the optimal subgraph assignment by computing a matrix containing all pairwise subgraph matching. Given two graphs $V = \{V_s^1, \dots, V_s^n\}$ and $U = \{U_s^1, \dots, U_s^m\}$. A cost matrix C is defined as

$$C = \begin{bmatrix} c_{1,1} & \cdots & c_{1,m} \\ \vdots & \ddots & \vdots \\ c_{n,1} & \cdots & c_{n,m} \end{bmatrix} \quad (3.12)$$

where $c_{i,j}$ denotes the assignment cost of a subgraph-to-subgraph matching $c(V_s^i, U_s^j)$. The subgraph assignment problem can be reformulated as finding a permutation $p = p_1, \dots, p_n$ that minimizes the cost $\sum_{i=1}^n c_{i,p_i}$. This is a Linear Sum Assignment Problem that can be efficiently solved by Munkres' algorithm [Mun57] in polynomial time. We refer the readers to [RNB07] [RB09] for more details about this algorithm. In this approach, it first finds the subgraph matching permutation that minimize the total cost. Then the remained non-matched subgraphs will be labeled as deletion $(\phi_v^* \rightarrow \varepsilon)$ or insertion $(\varepsilon \rightarrow \phi_u^*)$. An example of the full vertices matching process is shown in Fig. 3.20

Consequently, the global subgraph-to-subgraph matching is obtained by solving the cost matrix C . We can easily deduce the global vertex matching, denoted as Φ_{all}^* , from the optimal subgraph-to-subgraph matching. Note that all vertices in a deletion or insertion subgraph are labeled as deletion or insertion vertices, respectively. In another words, the Φ_{all}^* indicates the edit operation of all the vertices from graphs G_1 to G_2 . The cost of vertex edit operation is composed of the cost of substitution, deletion and insertion denoted as

$$C_v(G_1, G_2) = \sum_{n=1}^{N_1} d(v_{\phi_v^*(n)}, u_{\phi_u^*(n)}) + \sum_{n=1}^{N_2} c(v_{\phi_v^*(n)}, \varepsilon) + \sum_{n=1}^{N_3} c(\varepsilon, u_{\phi_u^*(n)}), \quad \phi^* \in \Phi_{all}^* \quad (3.13)$$

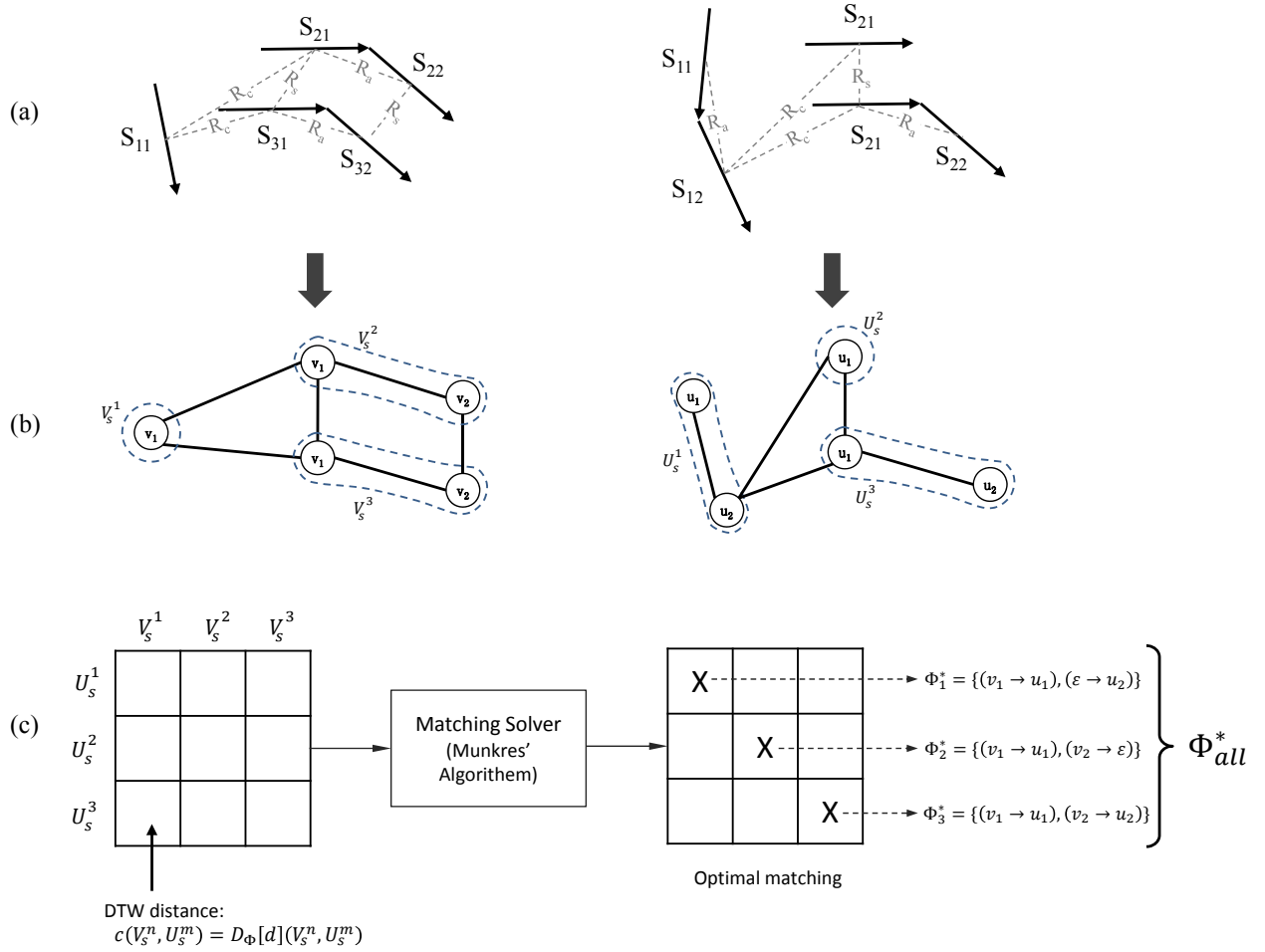
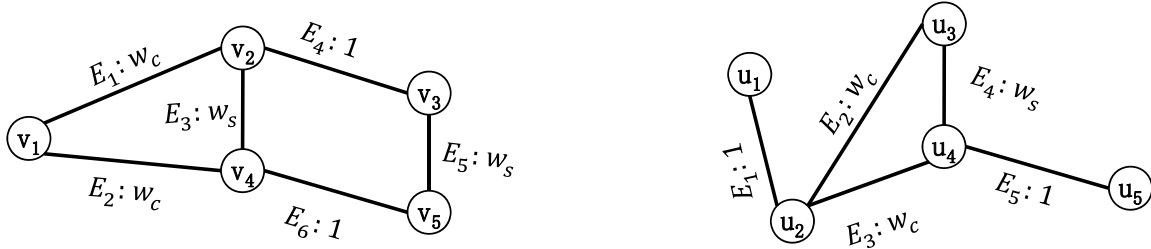


Figure 3.20: (a) Two gestures represented by sub-strokes and sub-stroke relations. (b) The graph representation of two gestures. The vertices which belong to a same original stroke are grouped into a subgraph V_s^n, U_s^m . (c) The DTW distance matrix is solved by Munkres' Algorithm to find the optimal vertices matching between two graphs. Consequently, the vertices edit operation set Φ_{all}^* can be deduced from the DTW alignment.

where d is the Euclidean distance of the vertices' feature vectors, c is the cost of insertion and deletion operation. Note that the deletion and insertion operations indicate a missing information in either graph. Therefore, we define the cost of deletion/insertion operation of each vertex as the value of its length feature.



| Edge edit operation | | |
|----------------------------|--|---|
| Substitution | $G_1(E_1) \rightarrow G_2(E_2)$ $G_1(E_2) \rightarrow G_2(E_3)$ $G_1(E_3) \rightarrow G_2(E_4)$ $G_1(E_6) \rightarrow G_2(E_5)$ | Cost is computed by Euclidean distance |
| Deletion | $G_1(E_4) \rightarrow \varepsilon$ $G_1(E_5) \rightarrow \varepsilon$ | Cost is a constant value |
| Insertion | N/A | Cost is a constant value |

Figure 3.21: The edge matching and edit operation of the two gestures in Fig. 3.20. The edge matching is implied by the vertices edit operation set Φ_{all}^* .

3.3.3.2 Edge matching

Once we obtained the vertex edit operation, the edit operations on edges are implied by edit operation on their adjacent vertices. Let $v_1, v_2 \in G_1$ and $u_1, u_2 \in G_2$, there are edges $e_1 = (v_1, v_2) \in G_1$ and $e_2 = (u_1, u_2) \in G_2$, the edge matching/substitution ($e_1 \rightarrow e_2$) is implied if two vertex matching ($v_1 \rightarrow u_1$) and ($v_2 \rightarrow u_2$) exist. Note that our graph contains three types of edge $\omega : E \rightarrow \Sigma_R : R_a, R_s, R_c$. Since each type has its unique attribute function, edge matching ($e_1 \rightarrow e_2$) could be assigned if both edges are same type. Otherwise they will be labeled as edge deletion ($e_1 \rightarrow \varepsilon$) or edge insertion ($\varepsilon \rightarrow e_2$). Note that the R_a edge only indicate the adjacent relation of the vertices without attributes and

has been processed in the subgraph matching step. Therefore, in this step the edge edit operation cost is calculated from the attribute function $w_s(R_s)$ and $w_c(R_c)$ of R_s and R_c , respectively. The edit operation cost for both R_s and R_c are defined as same as equation 3.13, where d is replaced by the Euclidean distance of $w_s(R_s)$ and $w_c(R_c)$. The cost of deletion/insertion operation of edge is set to a constant value. Consequently, we obtain two costs of edit operation denoted as C_s, C_c .

3.3.3.3 Graph classification

Based on the graph edit distance, our work explore three strategies for graph classification.

Strategy 1: So far, we have obtained three terms of edit operation cost, one vertex operation cost C_v and two edge operations C_s, C_c . Traditionally, the graph edit distance is defined as the summation of all the costs of edit operations [RB09] as

$$d(G_1, G_2) = (C_v + C_s + C_c). \quad (3.14)$$

In such a case, the simplest way for clustering the graph is using a distance based classifier such as K-nearest-neighbor.

Strategy 2: A better option regarding graph classification is to use the graph embedding algorithm. We refer to the graph embedding method presented by Kapsar and Horst [RB10]. The key idea of this approach is to use the distances of an input graph to a number of prototype graphs as a vectorial description of the graph. The definition is detailed as follow:

Definition 2. Graph Embedding. Let us assume G is an input graph, $P = p_1, \dots, p_n$ is a prototype set with n graphs, the mapping $\varphi_n^P : G \rightarrow \mathbb{R}^n$ is defined as the function

$$\varphi_n^P(G) = (d(G, p_1), \dots, d(G, p_n)), \quad (3.15)$$

where $d(G, p_i)$ is any graph dissimilarity measure between graph G and i -th prototype graph.

It means that each axis of the vector space $\varphi_n^P(G)$ is associated with a prototype graph p_i and the coordinate values of an embedded graph G are the distances of G to the prototypes. In this way we can represent any graph as a vector of real numbers with the help of prototype graphs. Hence, the vector $\varphi_n^P(G)$ can be regarded as the feature vector of graph G . Note that prototypes are normally selected from the training graph set under a certain criteria. The selection of the n prototypes is also a critical issue since not only the prototypes but also their number n affect the graph mapping φ_n^P . The prototype selection method is k-means clustering for each class. We refer the readers to [RB10] for more details about the graph embedding and prototype selection.

The problem now is reformulated as finding a pattern recognition way using the vectorial description of distances. We use SVM as the classifier in the experiment part.

Strategy 3: As *Strategy 2* introduces a way to extract a feature vector for an input graph, a more precise feature set, obtained by assumption that the edit operation costs are independent (in contrast to the summation for the graph edit distance), is investigated. Similar to *Strategy 2*, the feature set is computed by graph embedding. However, the distance between two graphs is defined as a three dimensions feature vector:

$$d(G_1, G_2) = (C_v, C_s, C_c). \quad (3.16)$$

By this way the local shape and structural/relation information is handled separately. But the new feature vector after the graph embedding will be in a length as three times of the number of the prototypes, which would significantly increase the classification complexity. We will compare these three strategies in the experiment part.

3.3.4 Experiments

We conducted experimental evaluation to verify the capability of our graph method. We firstly introduce our new multi-touch gesture dataset and three common used datasets. The results are then presented and compared with other result reported in the literature.

3.3.4.1 Dataset

As defined in [SW13], most common used datasets contain mono-stroke or multi-stroke symbols. To the best of our knowledge, there are few benchmark datasets for multi-touch and sequential multi-touch gesture recognition. To reflect the real challenges and complexity in multi-touch gesture recognition, we design more comprehensive multi-touch gesture and update our previous dataset. The new dataset is named as **MTGSetB** which contains more multi-stroke, multi-touch and sequential multi-touch classes. This section will give a first result baseline to address this new challenge.

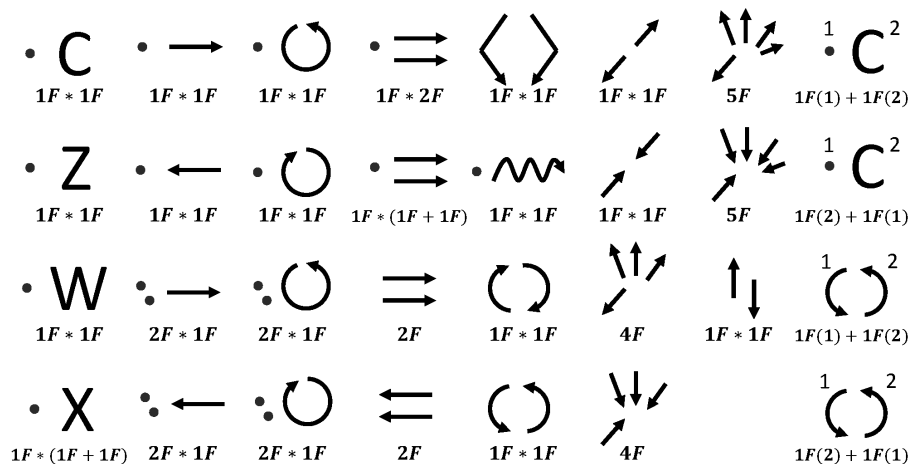


Figure 3.22: The multi-touch gesture templates in MTGSet.

The dataset consists of 31 classes, with 27 multi-touch gestures and 4 multi-stroke mono-touch gesture. Fig. 3.22 shows the pictograms we provide to the participants. A stroke indicates a finger movement and a dot means a finger holding on the screen. The textural description under each gesture illustrates how a gesture is performed. For instance, $2F*1F$ denotes a gesture is performed by two fingers in one hand and one finger in other hand. $nF(n > 1)$ indicates the number of finger involved while the asterisk (*) specifies the synchronous movement of the two hands. Meanwhile, if a gesture contains sequential stroke, a plus (+) is used to describe the order of the strokes. Note that the last column in Fig. 3.22 consists of 4 sequential mono-touch gestures. Each two of them have the same shape but with different written order (either from left to right or right to left). These gestures are also performed in a synchronous manner as multi-touch gesture in our dataset. We introduce these gestures to verify that our graph modeling and matching method are able to distinguish the gestures which have the same shape but different writing sequence. We invited 33 participants using a 27 inch touch screen for the data acquisition. In addition of these pictograms, participants are also provided with a short animation of each gesture for understanding the written order and direction (We use the same tool in Fig. 3.9). Each participant is asked to perform gestures from the first one to the last one and repeat for 6 times instead of repeating a same gesture 6 times. 6138 gestures are collected after discarding some mistaken gestures. The dataset is partitioned that 2790 gestures from 15 users are used for training, the remaining 3348 gestures from 18 users for testing. Fig. 3.23 presents a part of examples and their variations.

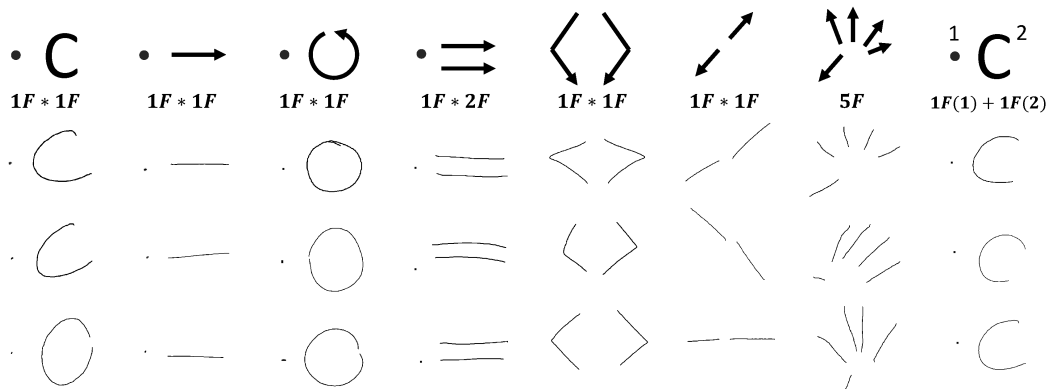


Figure 3.23: Samples and their variations in MTGSet.

Meanwhile, we also test our method on three common used online symbol datasets, **LaViola**, **ILG** and **NicIcon**. Note that these three datasets contain only mono-touch gestures which are not the cases we focus on. We make use of these datasets for the reason of evaluating the capability of our graph method on mono-touch symbols and comparing to other benchmark gesture classification methods.

The **LaViola** dataset [JZ07b] contains 11,160 digits, characters and symbols acquired from 11 users. This dataset has a high number of classes (48). Mono-stroke symbols

constitute the majority of samples while only a few classes contain two or three strokes samples. The data is clean and style-consistent. Results on this dataset have been reported in [DA13] [SKC08]

The **ILG** dataset [RFLDA12] is a collection of mono-stroke pen-based gesture. 38 users were asked to perform gestures for triggering 21 different commands in a simulated image edition software. The dataset is partitioned to 3 groups. Note that the first two groups contain user-defined gestures (user is allowed to design own gesture to trigger command, see the details in [RFLDA12]). Obviously, this two groups can only be used for writer-dependent experiments since it only offers few training samples for each class. To obtain a more general result, we select the third group (1926 samples) which has more classical properties: users all performed the same 21 gestures.

The last **NicIcon** dataset [NWV08] contains 26,163 pen-based symbols from 34 writers. Users were asked to draw the sketch of 14 special icons that are important in the domain of crisis management and incident response system. This dataset is more challenging because the samples are quite noisy and varying number and order of strokes.

3.3.4.2 Comparative results

The first experiment aims to evaluate the meta parameter and the three types of classification for our approach. Since our approach mainly targets on the multi-touch gesture problem, this experiment is only conducted on the MTGSet. For *Strategy 1*, we choose the 1-nearest-neighbor algorithm for classification. Obviously, it is costly to search the entire training set to find the nearest neighbor. We implement the same prototype selection algorithm in *Strategy 2* and *Strategy 3*, and compare their accuracy rate under different number of prototypes. Table 3.1 shows the accuracy rate according to different number of prototypes.

Table 3.1: Recognition accuracy as the function of prototype number per class.

| | Prototype number per class | | | | | | |
|-------------------|----------------------------|--------|--------|--------|---------------|--------|--------|
| | 1 | 3 | 5 | 10 | 20 | 30 | 50 |
| <i>Strategy 1</i> | 74.97% | 81.33% | 81.48% | 85.48% | 87.14% | 86.43% | 85.21% |
| <i>Strategy 2</i> | 96.35% | 97.34% | 97.51% | 98.22% | 98.39% | 98.29% | 98.22% |
| <i>Strategy 3</i> | 97.64% | 97.91% | 98.02% | 98.59% | 98.97% | 98.29% | 98.22% |

Not surprisingly, the accuracy rate increases accordingly with the increasing of the prototype number per class for all three classifiers. They reach at the peak at 20 prototypes per class and slightly decrease for more prototypes. It is remarkable that the *Strategy 2*

and *Strategy 3* show more stable results comparing to *Strategy 1*. They achieve a good performance even with a small prototype set by the help of graph embedding and SVM classifier. Generally, the *Strategy 2* and *Strategy 3* outperform the *Strategy 1*, but are very close between each other.

We note that there are few recognition systems which globally analyze and recognize the multi-touch gestures as symbolic gestures. We can hardly compare our method to other’s work. Therefore, we make use of our previous work HBF49 [DA13] as the benchmark result for the symbol recognition task. We implement both method on all four datasets to compare the accuracy rate. In this experiment we choose the *Strategy 3* with 20 prototypes per class since it achieves the best results in the previous experiment. Table 3.2 shows the results.

Table 3.2: Recognition accuracy of different methods.

| | Mono-touch gestures | | | Multi-touch gestures |
|---------------------------|---------------------|--------|---------|----------------------|
| | La viola | ILG | NicIcon | MTGSet |
| HBF49+SVM | 93.64% | 93.54% | 97.44% | 91.36% |
| Strategy 3(20 prototypes) | 93.18% | 91.30% | 93.17% | 98.97% |

By comparing these datasets, it can be seen that the HBF49 feature set performs better on the three mono-touch gesture sets. The reasons are that these three datasets do not need the synchronization relation between strokes and the features for a single stroke in our approach are far simpler than in the HBF49. Therefore, our approach is less powerful on the shape analysis aspect comparing to the HBF49. But the results of our approach are still competitive. On **la viola** and **ILG** sets, it shows close results compared with the HBF49. On the other hand, on MTGSet our graph modeling and matching method (98.97%) significantly outperforms the HBF49 (91.36%). This is mainly because the MTGSet contains gestures which are similar in shape but have different inner strokes temporal relations.

The comparison between HBF49 features and our graph modeling proves that our method has the ability to capture the synchronous and asynchronous writing relations between strokes. It is therefore dedicated to the multi-touch gesture recognition problem. That is why we offer a freely available multi-touch gesture database as a first baseline to multi-touch gesture recognition problems.

3.4 Conclusion

In this chapter, we report two graph models with two recognition systems for multi-touch gesture recognition problem. Comparing to the existent symbol recognition method, our graph models focus more on the feature extraction for inner stroke relations including spatial and temporal relations. In other words, our methods analyze the multi-touch gestures not only from their shape but also from their writing order and synchronism. As the first trial, the first system helped us to verify the effectiveness of graph modeling and graph embedding for feature extraction and gesture recognition. To better characterize the motion relations for multi-touch gestures, we then propose the second system based on quantified motion features. With the graph embedding and SVM classification methods, our system is proved to be able to distinguish the multi-touch gesture from global interpretation. Meanwhile, we make effort to create a challenge and comprehensive multi-touch gesture dataset (MTGSet). We have made it available online and hope it could be used as a benchmark dataset in community for further multi-touch gesture studies.

Chapter 4

Reject Option Based Early Recognition Algorithm

4.1 Introduction

In the previous section, we propose to use multi-touch gestures as indirect commands and introduce a classification method to recognize the gestures after their completion. However, in a common sense multi-touch gestures have widely been used for direct manipulation. One problem is that whether both of the two interactions can coexist in a same application, i.e. users can use multi-touch gestures for both interface/virtual element manipulation and executing commands. An example can be found in [PM13], authors use mono-touch gesture to achieve scrolling, rotation as manipulation and text input as command. A conflict is that these two interactions offer completely different feedback to the users. The direct manipulation need to give an instant feedback along with the finger's trajectory, whereas the indirect command need to wait until the end of a gesture. Obviously it is not possible to distinguish the user's intention at the very beginning of a gesture. In Fig. 4.1 we illustrate this conflict with an example in [PM13].

Therefore, the problem becomes to whether we can correctly recognize a gesture by a short beginning part so that we can quickly find the user's intention and choose a correct feedback. How much information we need to recognize a gesture from a gesture set and how fast we can achieve. With these questions, we explore the **Early Recognition (ER)** strategy for multi-touch gestures.

Fig. 4.2 illustrates the general difficulties of ER. Since these three gestures have a common beginning part, there is no evidence to distinguish the gestures before λ_1 . The ER is actually a tradeoff between 'earliness' and 'accuracy', the more time/trajectory it waits for, the more distinctive information it would have for decision. An ideal ER system should be able to learn the common part and avoid making a decision before λ_1 . Meanwhile, it should also be able to distinguish the gesture A from B and C soon after λ_1 . Moreover, gestures have different common parts between each other, e.g. gesture B

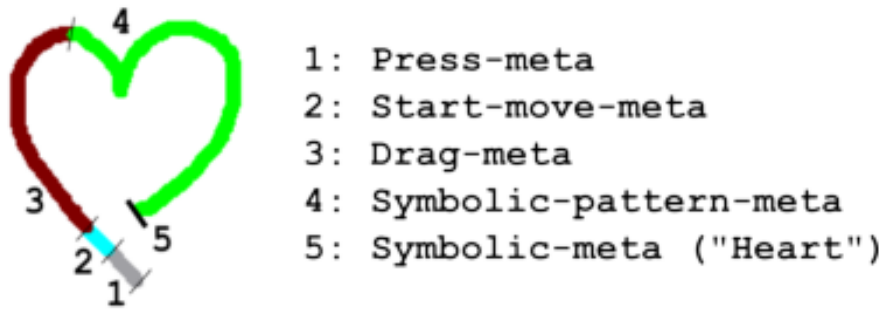


Figure 4.1: Early recognition for a “Heart” gesture. In stage 3, the trajectory is recognized as “Drag” manipulation which gives an instant feedback along with the trajectory. In stage 4, the trajectory is recognized as a “Symbolic-pattern”, i.e. an indirect command. System need to wait until the end of the trajectory to interpret it as a “Heart” symbol. [PM13].

and C have a longer common part than gesture A and B. Each gesture need to have an independent ER template against other gestures.

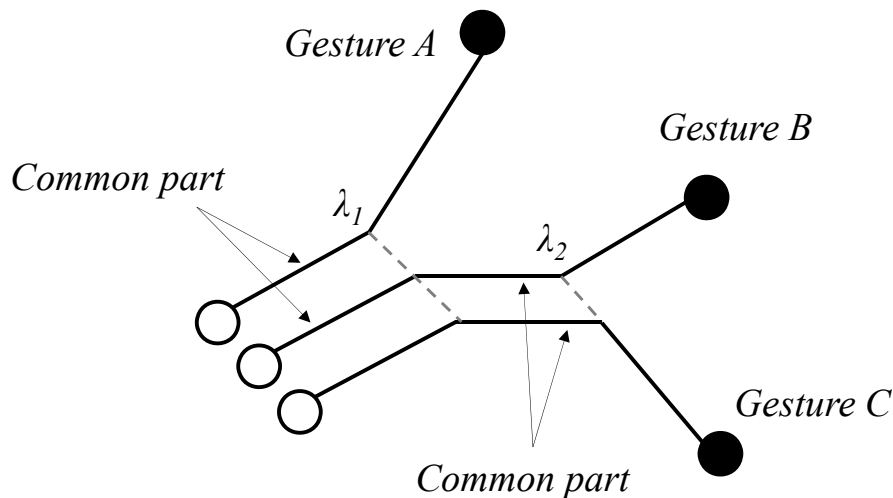


Figure 4.2: The common part ambiguity for early recognition. Three gestures have different common parts between each other.

From the state-of-the-art, the ER works have been mostly developed for motion prediction problems [MUK⁺06] [KSNT11]. A basic idea is to employ a partial matching method, where the recognition result of an input pattern is determined by the matching distance of its beginning part from reference patterns. The Dynamic Time Warping (DTW) algorithm is a widely used method to search for an optimal partial alignment. Another approach is combination of classifiers $\{h_1, \dots, h_t, \dots, h_T\}$ [ISS10][UA08], where h_t is a weak frame classifier at t^{th} frame (i.e., time t). The recognition result at the t^{th} frame will be determined by combining t recognition results provided by $\{h_1, \dots, h_t\}$. This method assumes the input pattern having the same performing speed as the reference pattern.

The time frame based ER works have a pre-defined context that every gesture is written in a same velocity. In our opinion, the gesture trajectory on each time frame is just an external representation, the substantive information is the motion's variation, i.e. the length of the trajectory. For instance, a gesture which moves very slowly contains less information frame by frame. Therefore, We propose that the early recognition should be investigated based on length of the motion's trajectory rather than the time frame. Note that this strategy excludes the factor of writing velocity variation, but meanwhile it loses the capacity to distinguish two gestures which have same shape but only differ in writing velocity.

We study in this work early recognition for handwritten touch gestures. Unlike the previous works that investigate the early recognition based on time frame, we believe that the time frame does not represent the motion information, e.g. a gesture which moves very slowly contains less information frame by frame. We propose that the early recognition should be investigated based on difference of the motion rather than difference of the time.

For the practice of ER system in a real application, another difficulty is the gesture size normalization. In training process, usually a classification system need to normalize each gesture to a fixed size bounding box before feature extraction so that the value of features can be unified to a fixed scale. However, in a gesture size free context, it is difficult to normalize the early part of a gesture without knowing the size of the full gesture. Fig. 4.3 shows an example of this problem. Fig. 4.3(a) is a normalized gesture which is assumed to be a template. Fig. 4.3(b) (c) are two unknown gestures with different size. Assume that we need to achieve the early recognition by partial matching with the partial trajectories in a length of l . Without normalization, a trajectory with a length l could be a small part of a large gesture or a large part of small gesture. Apparently, it is difficult to achieve the matching with their original size since they have different appearance. However, it is also difficult to normalize the partial trajectories in (b) and (c) into a same scale as in (a) because the potential size of the full gestures is unknown at this early stage. Therefore, the problem can be depicted as how to achieve the partial matching in a size free context.

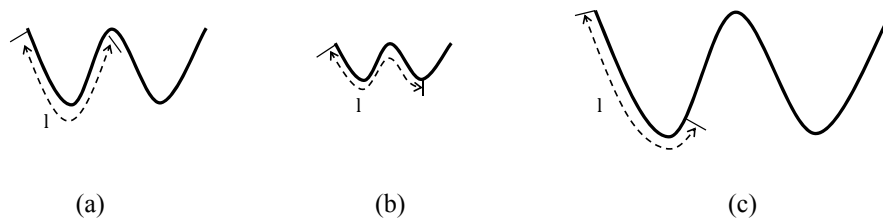


Figure 4.3: (a) A normalized gesture as a template. (b) (c) In a size free context, due to the input gestures having a variety of the size, a trajectory with a length of l may cover different parts of a same type gesture.

In this work, we propose to build an early recognition system being able to deal with

ambiguous common parts under free drawing context for handwriting gesture recognition. We control the progress of the gesture using its length instead of time duration. A reject option is proposed to postpone the decision until enough confidence is achieved. The rest of the paper is organized as follows. Section 4.2 presents the structure of the multi-classifier method and the reject option strategy. Next in section 4.3, we report the experimental result to show the earliness and accuracy of the system. Experiments are conducted on two freely available dataset **ILG** [RFLDA12] and **MGSet** [CAMVG15]¹. The ILG dataset contains common mono-touch gestures which are assumed for abstract command while the MGSet contains special multi-touch gestures which can be both used for abstract command and direct manipulation. Finally, we conclude this work and discuss the perspectives in section 4.4.

4.2 Multi-classifier Early Recognition

To deal with the size normalization problem for ER in a size free context, we propose a multi-classifier recognition system as shown in Fig. 4.4. Each classifier is trained by a fixed length of partial gesture so that different classifier is responsible to recognize different length of coming incomplete gesture. However, as we explain in the previous section, a trajectory with a length l could be a small part of a large gesture or a large part of small gesture. In other words, a coming incomplete gesture (i.e. an incomplete trajectory) can not be explicitly sent to a certain classifier according to its length because we can not estimate the size of its potential full gesture. Therefore, the incomplete trajectory will be recognized by all classifiers. The recognition result is determined by a fusion of the results from all classifiers. Details will be described in section 4.2.1. To deal with the ambiguous beginning parts shown in Fig. 4.2, the result is filtered by a reject option, i.e. a result with low confidence value will be rejected so that system will wait for enough information to make a decision. The rejection algorithm and fusion of classifiers will be proposed in section 4.2.2.

4.2.1 Segment Classifier

Consider a set of N training gestures $x_i | i = 1, \dots, N$, each gesture x_i is a sequence of points $x = p_0, \dots, p_e$ normalized and centered in the unit square bounding box. As we discussed in section 4.1, users may perform a same gesture at different velocity. In other words, during a fixed time interval Δt , the length of the gestures performed by different users may be different. Therefore, our early recognition is based on the curvilinear distance segmentation rather than a time segmentation, i.e. each classifier is trained by fixed length partial trajectories. Fig. 4.5(a) illustrates the segmentation of a gesture based on

¹<https://www-intuidoc.irisa.fr/en/category/bases-de-donnees/>

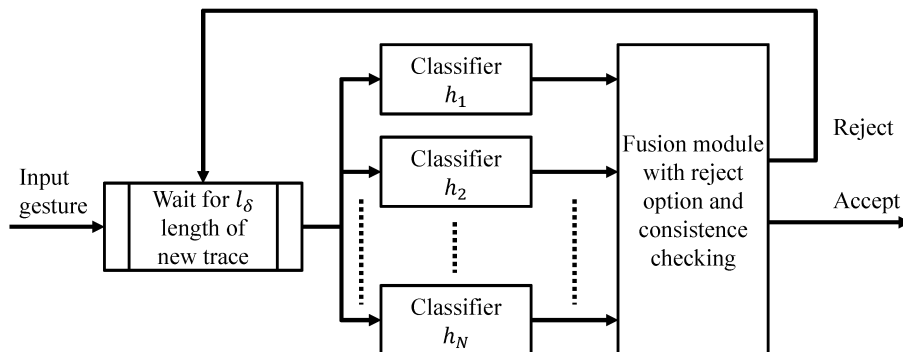


Figure 4.4: The structure of multi-classifier early recognition system.

a interval l_Δ . The interval l_Δ is an empirical length depending on the number of classifiers and the size of bounding box.

From all the training gestures, we build N segment sets S_i , with $i = 1, \dots, N$, where each S_i set represents all the segments of Seg_i whose length is equal or less than il_Δ . Fig. 4.5(b) shows the training procedure of each classifier and its corresponding segment set. Note that although the normalization step normalize the size of each gesture into a fixed bounding box, the trajectory length of different gestures are still different. Consequently, a long trajectory can generate more segments to feed the segment sets while a short one may generate only one segment. Therefore, the number of training segments in different set S_i may be different. S_1 will always cover the beginning part of all the training gestures, while S_N only contains the gestures which are longer than $(N - 1)l_\Delta$. We denote the segment classifiers as $\{h_1, \dots, h_i, \dots, h_N\}$, each classifier h_i is trained by the feature vectors of the i th segment set S_i .

In the recognition step, as we explained in Fig. 4.3 an arbitrary input gesture x at length l ($(i - 1)l_I < l < il_I$) can not be specifically recognized by classifier h_i because of the size free context. Therefore, an arbitrary gesture x should be processed by all the classifiers and determined by the one giving the highest probability value. Let $h_j(x, c_i)$ be the probability of the best class c_i obtained by the classifier h_j , the result of multi-classifier is

$$H(x, c_i) = \max_{j=1, \dots, N} h_j(x, c_i) \quad (4.1)$$

4.2.2 Rejection Algorithm

Referring to the work in [ZSHN10] [MA06], our reject option is designed from two aspects: ambiguity and outlier. Fig.4.6 illustrates reject option boundary based on a classification space. The ambiguity refers to the patterns which are near the pair-wise classification hyperplane. These patterns reflects the common part ambiguity as depicted in Fig. 4.2. The outlier refers to the patterns which are far away from the class center. Because of the size free context that an input pattern can not be specified to a certain classifier, the

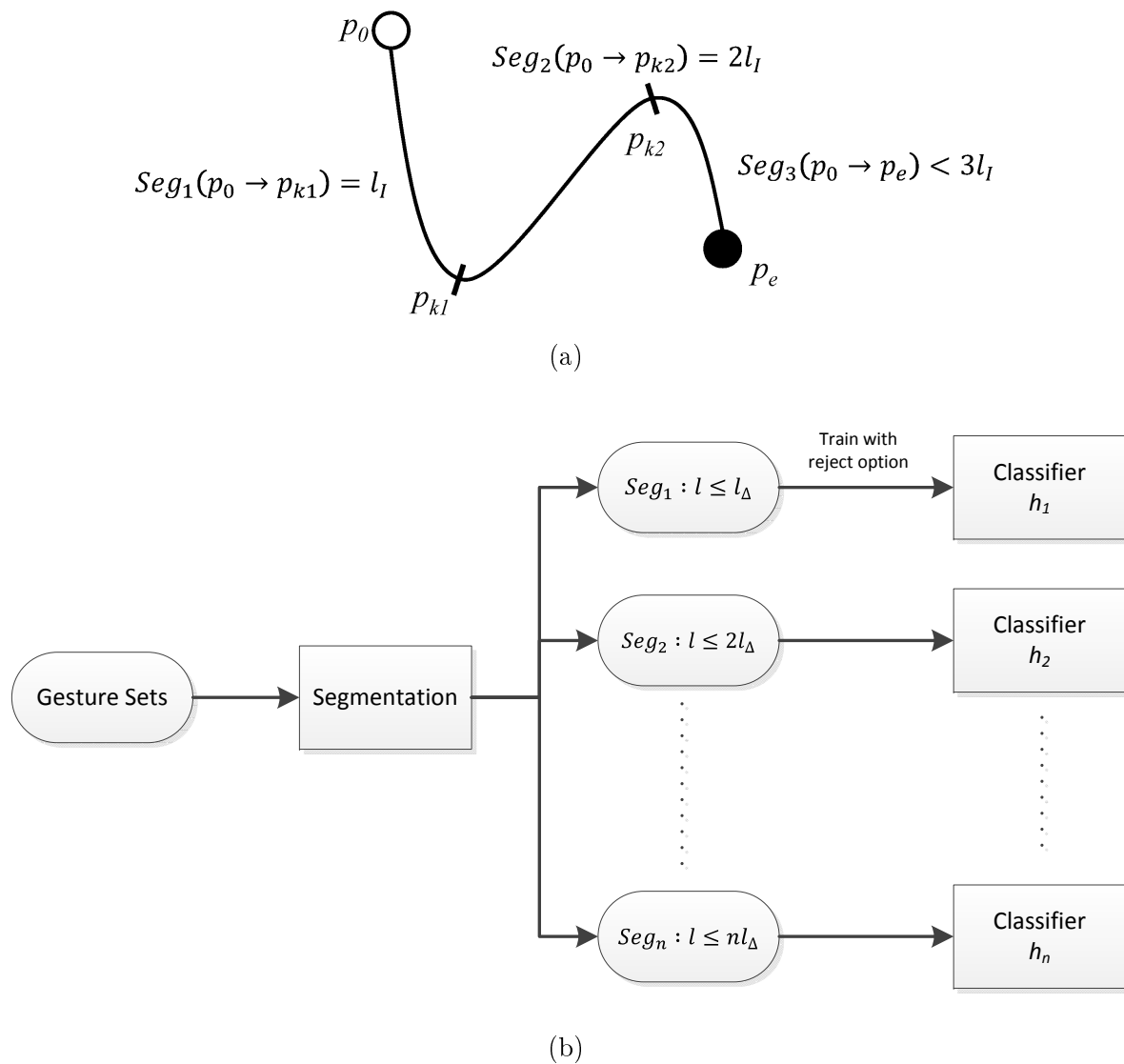


Figure 4.5: (a) Trajectory of an example gesture. p_0 and p_e are the starting and ending point, respectively. p_{k1} is the keypoint where the length of seg_1 (from p_0 to p_{k1}) is l_Δ . p_{k2} represents the point at $2l_\Delta$. Since the total length is less than $3l_\Delta$, this trajectory will offer three segments for training. (b) Classifiers are trained with different segments.

outlier rejection is used by each classifier to explicitly reject the pattern which does not belong to the scope this classifier.

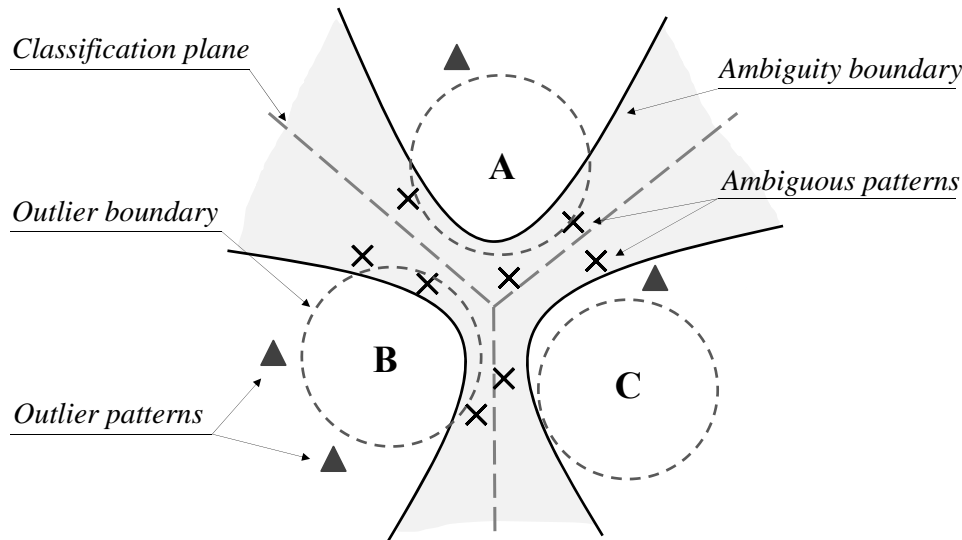


Figure 4.6: Ambiguous patterns and outlier patterns in multi-class recognition rejection problem. The dotted straight lines represent the pair-wise hyperplanes to separate two classes. The curves are ambiguity rejection boundaries for each class.

4.2.2.1 Ambiguity rejection

We deduce from Fig.4.6 that a good ambiguity rejection solution is to define a class-pair dependent threshold which rejects the ambiguous pattern close to the pair-wise hyperplane. However, it is inefficient to maintain the entire class pair space when the class number is large. A trade-off is to use class dependent threshold that defines an ambiguity boundary against all the other classes. We define as in [MA06] the reliability function ψ_i^{Amb} to well interpret the ambiguity condition. The ambiguity determines if a shape is near the decision boundaries. So let $S = (s_1, s_2, \dots, s_k)$ be the confidence or distance scores for each class output by the classifier. We can use the difference between the best class C_1 and the second one C_2 to form the reliability function as:

$$\psi_i^{Amb} = \frac{s_i - s_j}{s_i}, \quad (4.2)$$

where $i = C_1, j = C_2$. And we have the rejection decision as:

$$r^{Amb} = \psi_i^{Amb} < T_i^{Amb}. \quad (4.3)$$

T_i^{Amb} is the class dependent threshold that the result will be rejected if $\psi_i^{Amb} < T_i^{Amb}$.

4.2.2.2 Outlier rejection

Since each classifier is trained by a set of segments of a certain length, a classifier should be able to reject a gesture which is not similar to any of the training data. It ensures that in the multi-classifier structure, only the relative classifier will give response to an input gesture while others would make rejection.

Shown in Fig.4.6, the outlier samples locate far from the center of each class. Therefore, the most important information for this rejection option is the intrinsic description of the learned data. Depending of the used classifier, this information is not always directly available [MA06]. If the classifier outputs approximate the probability density of the learned data as in classifiers like RBFNN, Mixture of Gaussian, then the reliability function can be defined as equation 4.4.

$$\psi_i^{Out} = s_i \quad (4.4)$$

In our case, we use SVM classifier based on the distance to graph prototypes representing the different classes (see graph embedding in section 3.3). So by construction, the distance to these prototypes is a good feature to extract the intrinsic description of the training data. Thus in the experiments we use the equation 4.5 to compute the minimum distance of sample x to the set of prototype P_i of class i using the distance d as defined in equation 3.14.

$$d_i(x) = \min_{p_j \in P_i} d(x, p_j) \quad (4.5)$$

Then the reliability function ψ_i^{Out} can be defined with equation 4.6 and used in the rejection decision as equation 4.7.

$$\psi_i^{Out} = e^{-\frac{(d_i - \mu_i)^2}{2\sigma_i^2}} \quad (4.6)$$

where μ_i and σ_i are the mean distance and deviation for each class i learned from the validation set.

$$r^{Out} = \psi_i^{Out} < T_i^{Out} \quad (4.7)$$

where s_i is the output score of the best class. T_i^{Out} is the class dependent threshold that the result will be rejected if $\psi_i^{Out} < T_i^{Out}$.

4.2.2.3 Threshold optimization

We firstly define some notations to better explain the result of the reject option. Considering a set of N training samples, Table 4.1 shows the notations to represent the number of samples in different condition after recognition and reject option.

Table 4.1: Notations to represent the number of samples in different conditions. With these notations: $N = N_A + N_R = N_{cor} + N_{err} = N_A^T + N_A^F + N_R^F + N_R^T$

| Sample set (N) | Reject option | |
|------------------------------------|--------------------------|--------------------------|
| | Accept (N_A) | Reject (N_R) |
| Correctly classified (N_{cor}) | True Accept (N_A^T) | False Reject (N_R^F) |
| Mis-classified (N_{err}) | False Accept (N_A^F) | True Reject (N_R^T) |

To evaluate the threshold, we compute the False Accept Rate (FAR), and False Reject Rate (FRR) as:

$$FAR = \frac{N_A^F}{N} \quad (4.8)$$

$$FRR = \frac{N_R^F}{N} \quad (4.9)$$

For ambiguity rejection, N_R^F are the training samples which are correctly classified but wrongly rejected by reject option while the N_A^F are wrongly classified but accepted. Note that it is better to prepare a validation dataset since the high precision in training data makes N_A^F close to 0, which leads to unavailable optimization. For outlier rejection, since each classifier is trained with a set of segments in certain length, the positive samples are the classifier's training set while the negative samples are the training sets for other classifiers. The acceptance of negative samples will be count for N_A^F and the rejection of positive samples will be N_R^F .

In rejection, the aim is to obtain the lowest error rate while rejecting least correct results. Intuitively, the optimization of the threshold is to find a trade-off between the FAR and FRR . Therefore, the optimal threshold for class i is defined as:

$$T_i^{opt} = \arg \min_{T_i} \sqrt{\alpha_e FAR_i^2(T_i) + \alpha_r FRR_i^2(T_i)} \quad (4.10)$$

where the weights α_e and α_r are used to balance the impact of each rate. In general case, these parameters are set to 1. Since we use the class-wise threshold, we measure the FAR_i and FRR_i based on each class i to learn the threshold. The two thresholds are learned independently.

Finally, the rejection of an input gesture is made if it is rejected by either reject option.

$$r = \max(r^{Amb}, r^{Out}) \quad (4.11)$$

If the input gesture is accepted, the probability $h_j(x, c_i)$ of the class i , as shown in (4.1), is the conjunction of both reliability function:

$$h_j(x, c_i) = \psi_i^{Amb} * \psi_i^{Out} \quad (4.12)$$

With the reject option, the equation (4.1) will be changed to

$$H(x, c_i) = \begin{cases} \text{Reject,} & \text{if } \prod_j^j r_{j,i}(x) = 1 \\ \max_j (\overline{r_{j,i}}(x) * h_j(x, c_i)), & \text{otherwise} \end{cases} \quad (4.13)$$

4.3 Experimental Result

The evaluation experiment has been conducted on the **MTGSetB** and **ILG** datasets as referred in section 3.3.4.1. Note that for both datasets, we partition 20% of the data from the training set as a validation set to learn the ambiguity threshold.

The number of classifiers in our experiment is set to 3. Before training, each gesture is normalized into a 500x500 pixels bounding box and segmented to 3 partial gestures with length 250, 500, 750 pixels to feed for the 3 classifiers, respectively. The classifier we used for each h_i is Graph + LibSVM as described in 3.3. The confidence scores for ambiguity threshold learning are the probabilities from LibSVM. For outlier threshold, we use clustering algorithm to find three centers for each class, and compute the distances of a input gesture to the centers. The minimum distance is used as s_i in (4.4) to learn the outlier threshold. These can be replaced by any classifiers which give output confidence score for each class.

For the early recognition, each input gesture is recognized on every 50 pixels of its incremental length. We firstly evaluate the early recognition with regarding to the different length of input gestures. Referring to the notations in table 4.1, we measure the False Accept Rate ($FAR = N_A^F/N$) and Reject Rate ($RR = N_R/N$) when using the reject option and compare them with the traditional Error Rate ($ER = N_{err}/N$) without reject strategy. The results are shown in Fig. 4.7.

Both results show that without the rejection algorithm, the ER is very high at the beginning since gestures are still ambiguous to take a distinction. Accordingly, the rejection algorithm is effective to reject most of the gestures at beginning. The RR decreases along with the decreasing of ER (without reject) which means that it well rejects the ambiguous gestures but accept the gesture as soon as it has enough distinctive information. This strategy leads to a good performance of FAR which is very low at the beginning and always lower than ER at any input length. Meanwhile, the RR is always higher than ER in the ending part, which means that some correctly classified gestures are rejected. This is the negative effect of the reject option; a low error rate is obtained at the cost of a high reject rate.

In operational use case, a reasonable strategy to prevent noisy decisions consists in filtering the decision by considering several consecutive outputs of the classifier. Consequently, a decision is finally accepted when the classifier gives n consecutive times the

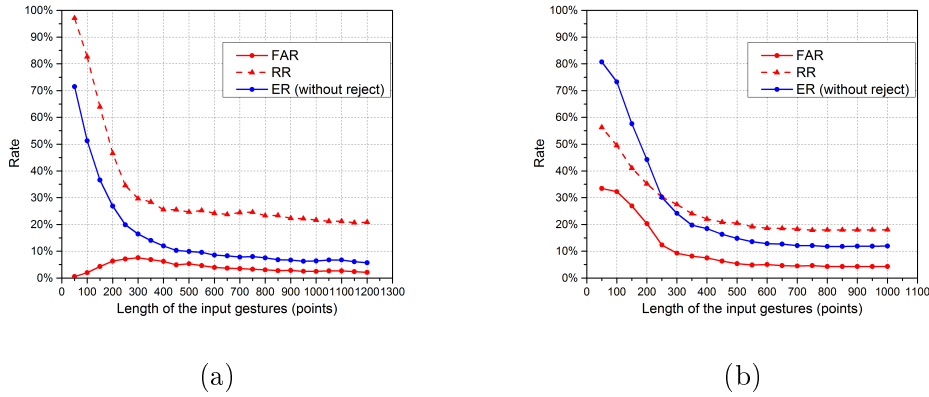


Figure 4.7: Recognition results with respect to the length of input gesture on two datasets. FAR and RR are obtained using the reject option while ER is the traditional mis-classified rate.

same output. Results on the two datasets are shown in Table 4.2 from $n = 1$ to 6. Referring to the notation of Table 4.1, TAR is True Accept Rate ($TAR = N_A^T/N$), FAR is False Accept Rate, RR is Reject Rate which represent the percentage of gestures which are rejected at every length until their completion, CR is correct rate ($CR = N_{cor}/N$). The earliness means the average percentage of the a gesture being written at the time it is recognized. Since the ILG data does not contain the time label, the average decision time (Avg.T) is only measured on MTGSet.

Table 4.2: Recognition rate with consistence checking.

| Dataset | n | Reject Option | | | | | No Reject Option | | | |
|---------|---|---------------|--------|--------|-----------|------------|------------------|--------|-----------|------------|
| | | TAR | FAR | RR | Earliness | Avg. T(ms) | CR | ER | Earliness | Avg. T(ms) |
| MGSet | 1 | 81.89% | 14.56% | 3.54% | 37.04% | 456.21 | 24.88% | 75.12% | 8.13% | 297.23 |
| | 2 | 83.44% | 10.85% | 5.71% | 46.82% | 523.34 | 48.78% | 51.22% | 21.32% | 368.07 |
| | 3 | 82.38% | 8.85% | 8.77% | 55.89% | 591.33 | 67.60% | 32.40% | 33.98% | 437.85 |
| | 4 | 82.20% | 6.06% | 11.73% | 66.16% | 669.86 | 79.59% | 20.41% | 45.44% | 518.21 |
| | 5 | 80.35% | 4.60% | 15.05% | 71.03% | 738.17 | 85.83% | 14.17% | 54.93% | 598.04 |
| | 6 | 77.42% | 3.41% | 19.17% | 77.54% | 811.38 | 88.62% | 11.38% | 62.34% | 660.90 |
| ILG | 1 | 30.65% | 67.15% | 2.20% | 34.81% | N/A | 21.22% | 78.78% | 18.03% | N/A |
| | 2 | 64.15% | 26.42% | 9.43% | 75.53% | N/A | 42.85% | 57.15% | 56.17% | N/A |
| | 3 | 73.98% | 11.22% | 14.80% | 92.24% | N/A | 68.29% | 31.71% | 82.16% | N/A |
| | 4 | 77.72% | 6.26% | 16.02% | 97.62% | N/A | 79.51% | 20.49% | 92.67% | N/A |
| | 5 | 77.80% | 4.88% | 17.32% | 99.19% | N/A | 85.45% | 14.55% | 97.27% | N/A |
| | 6 | 77.72% | 4.55% | 17.72% | 99.68% | N/A | 87.56% | 12.44% | 99.08% | N/A |

It shows an acceptable result on MGSet dataset that the accuracy rate of first time decision is 81.89% which is obtained with an average of 37.04% length of gestures. Com-

paring to the third result with no reject option, where the decision is also achieved around 33.98% length by 3 consecutive same results, the FAR is less than half of the ER. With the increasing of the time for consistence checking, the decision is postponed to obtain less errors. The FAR decreases from 14.56% to 3.41% while the RR increases from 3.54% to 19.17%. It indicates that we have to find a trade-off between the error rate and the reject rate. The result by $n = 2$ may be considered as an acceptable one where the FAR is 10.85% and RR is 5.71%. The comparable result from no reject option is shown at $n = 4$ where the CR is 79.59% (3.85% lower than TAR:83.44%) and ER is 20.41% (9.56% higher than FAR:10.85%). In other words, the reject options minimize the error rate by offering reject. Although there are 5.71% samples are rejected during the recognition, we believe that in a real practice it is better to reject an input and provide some ambiguous options to select than giving a wrong result. By this way, user only need to make a selection instead of removing the wrong input and re-draw it again.

However, the result on ILG shows not as good as MGSet. The accuracy rate is only 30.65% for the first decision. From the Fig. 4.7(b), the FAR is around 20% to 30% from 50 to 200 points. Decisions made on this stage cause much more errors than MGSet. Therefore, the first time decision may not be acceptable in this situation. The accuracy rate on ILG dataset shows a great improvement using consistence checking. With $n = 2$, the TAR is 33.5% higher than $n = 1$ while the FAR decreases 52.73% comparing to $n = 1$. A higher time of consistence checking seems not useful since the corresponding Avg. length is over 90%.

4.4 Conclusion

In this chapter, we focus on the difficulties of involving both direct manipulation and indirect command gesture in a same context. To resolve the conflict of these two interactions, we propose a reject option based multi-classifier system for handwritten gesture early recognition. The principle is to recognize gestures as soon as possible from their beginning part but also avoid the recognition if gestures contain similar beginning parts. We propose a multi-classifier structure that different classifiers are responsible to recognize different part of partial gestures. Since our system is considered to be used in a large screen and gesture size free context, taking into account the gesture size normalization inconsistency between training and testing, the early recognition result is determined by the fusion of all classifiers. The reject option for each classifier is designed to deal with the ambiguous early parts between gestures. The experiment gives a promising result on MTGSet. The system achieve 83.44% accuracy rate with 46.82% earliness of input gestures. Comparing to the no reject option system, the error rate is very low at the beginning part which proves that our reject algorithm works well to reject the ambiguous gestures.

Our future work will focus on investigating the automatic selection of the optimal number of classifiers and segment length for training instead of using empirical selection as in our experiment. Meanwhile, we will develop a application to mix the usage of direct manipulation and command shortcut for multi-touch gestures and better analyze the usability of early recognition for handwritten gestures. In the next section we develop a structure document composition context as a first prototype which involves multi-touch gestures for complex commands.

Chapter 5

Structured Document Composition in Multi-user Context

5.1 Introduction

A large multi-touch display allows multiple users to simultaneously interact in the same context and work together. Indeed, many researches and commercial products propose tangible interfaces which support simultaneous participation of multiple users. However, most of these interfaces only allow users to interact with virtual elements which need only simple direct manipulation. To the best of our knowledge, there are no research focusing on the freely-drawn sketch or indirect commands for multiple users.

In this chapter we study the difficulties of multi-user freely-drawn sketch context. This work is supported by **Excense** company who dedicates to design a handwriting diagram context that make people propose and exchange their idea between each other. We choose the use case of sketch drawing to illustrate the complexity of multiple users composing a structured document. Indeed sketches as mind map or flowchart need lot of different gestures to draw a various type of symbols as nodes and arrows. Fig. 5.1 shows a prototype of the multi-user interface, it allows two users to simultaneously input symbolic elements in a mind map context. Generally, an ideal mind map diagram composition interface should consists roughly of the following features [Blo96]:

- 1) Stroke segmentation/grouping, to isolated symbols.
- 2) Symbol recognition.
- 3) Identification of spatial and logical relations among symbols.
- 4) Direct manipulation or indirect command gesture to interact with the existed elements.
- 5) Text separation.
- 6) Text recognition.

Our current study focuses on the first two features. The rests stand as future perspective. In a multi-user context, it becomes more difficult to make a clear stroke segmen-

tation/grouping since there are no clear spatial or temporal boundaries between isolated symbols from the stroke stream. Meanwhile, to provide a freely-drawn context, users are allowed to input a symbol by multi-touch gestures. Therefore, the challenges of our study can be described as: how to recognize the multi-stroke and multi-touch gestures in a multi-user composition context.

In section 5.2, we introduce the dataset we acquired to support our study. We will describe the data acquisition procedure and discuss the multi-user features compared to a traditional diagram dataset. In section 5.3, we present our first strategy for stroke grouping and gesture recognition method based on eager interpretation. The experimental results are given in section 5.4. Finally the last section concludes with the perspectives to improve the results and go further in the process.

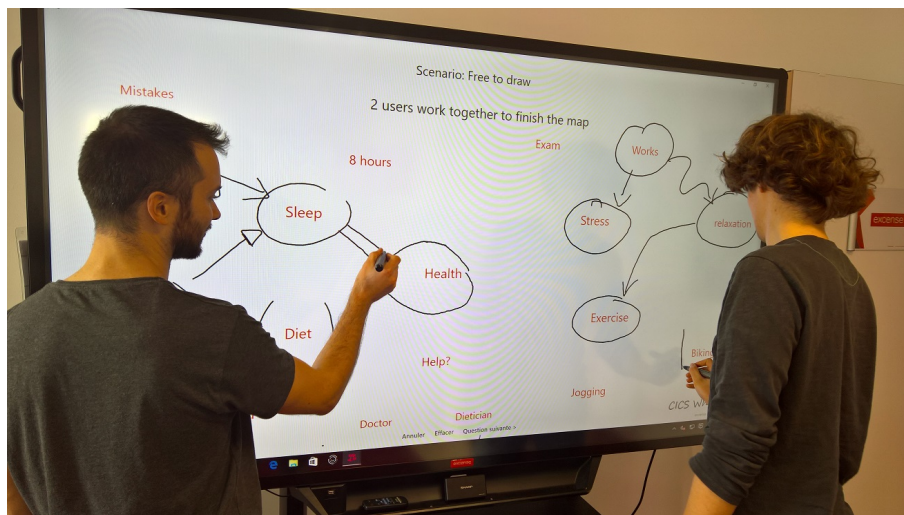


Figure 5.1: The diagram data acquisition procedure on a 80" touch screen. Two users are drawing the diagram together using stylus.

5.2 Multi-user diagram database

Since there are few works on multi-user handwritten analyzing, it is essential to begin with a representative multi-user handwritten document dataset. In order to be closely engaged with a practical application, we propose to use mind map diagram as the multi-user scenario. In this section we present the data acquisition procedure and the multi-user features of the dataset. This dataset is achieved with the help of **Excense** company.

5.2.1 Diagram acquisition

A mind map is a diagram used to visually organize information. It is often created around a single concept to which associated representation of ideas such as words, images, symbols are added. The ideas are usually connected by lines, arrows or grouped by shapes,

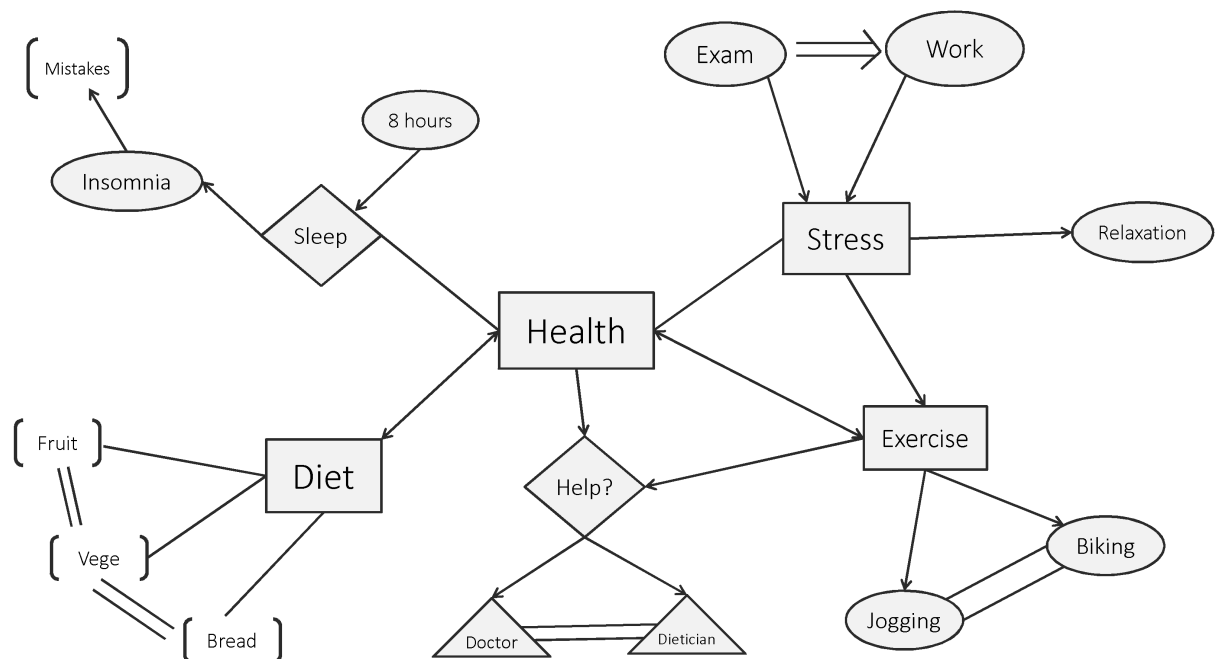
boundaries. An example of a mind map diagram is shown in Fig. 5.2 (a). As a first step towards a more complex dataset, we focus more on the graphical symbols and their links rather than the text. We offer the users a white board scenario with all descriptive texts on it. Each time we ask two users (standing side by side) to draw the correct graphical symbols around the texts and connect them using lines, double-lines or arrows using either fingers or styluses. An example of the collected handwritten diagram is shown in Fig.5.2 (b). We designed two layouts of this mind map for more diversity in the data acquisition.

A total of 21 people partitioned half to half into two groups participated. Each pair of participants was asked to switch their position after completing a map and drew again. As each couple draw also the two layouts, it means that each couple of users generate 4 diagram samples. Consequently, 42 handwritten diagrams have been collected. 10 participants of group 1 were asked to draw the symbols on a 27" touch screen by fingers. To record the way people naturally draw, participants were given all the freedom to draw the symbols in any way or order they prefer. Since users in this group draw symbols by fingers, some participants tend to use multi-touch manner to draw the multi-stroke symbols such as 'bracket' and 'double line'. Users from group 2 made the acquisition on a 80" touch screen by stylus. In this case they have more free space to draw the symbols and may simultaneously draw different symbols very close to each other. Since all the symbols are drawn by stylus, the multi-touch case does not exist in this group. Fig. 5.1 shows the data acquisition scenario from group 2. The stroke grouping and groundtruth labeling tasks were achieved manually after data acquisition with a dedicated application.

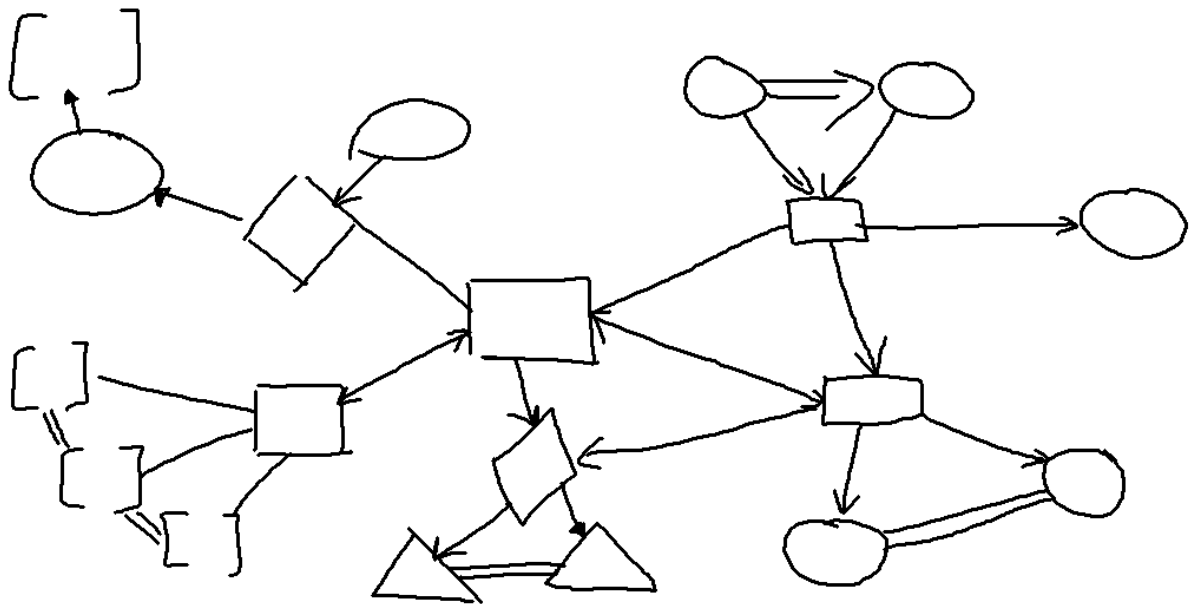
5.2.2 Diversity of the content

The predefined symbols can be classified into 9 categories shown in Fig. 5.3. In observing the collected data, users may draw a same symbol with different number of strokes. Examples of rectangle symbols are displayed in Fig. 5.4. Note that one of the examples shows a very special broken straight stroke case. The User intends to draw a straight stroke, but his finger accidentally lifted up because of the unstable friction on the screen. This situation causes some symbols containing an unusual large number of strokes.

As we introduced in the previous section, some 'bracket' and 'double line' symbols may be drawn by multi-touch manner, which means the two strokes of the symbol are drawn simultaneously. Some 'arrow' symbols are written by one stroke, while some others are firstly written as a 'line' and waited after a long time delay the head added. If the head of an 'arrow' is drawn within the same stroke or immediately added just after the straight line stroke, we group the strokes and label them as 'arrow'. If the head is added after a significant long time delay, during which the body of arrow is supposed to be processed as the 'line', we would make an independent 'direction' class for the head. The heads of double ways arrows are also labeled as 'direction'. Table 5.1 shows the distribution of the symbols in the dataset with also the average and max number of strokes used to draw



(a)



(b)

Figure 5.2: Example of a mind map diagram and the corresponding handwritten diagram without text.

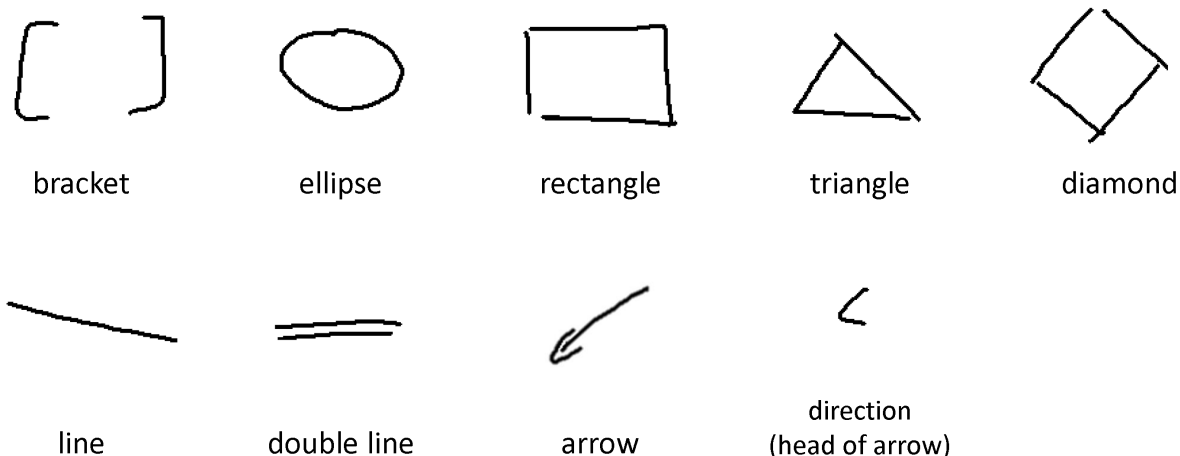


Figure 5.3: Samples of isolated symbols in diagram.

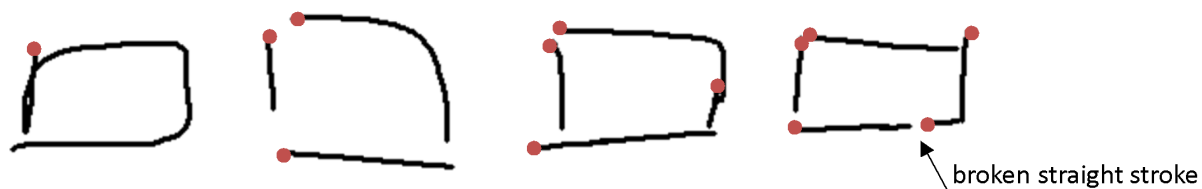


Figure 5.4: Variability of the rectangle symbol.

them.

The most important feature of this dataset is the mixture of multi-touch and multi-stroke symbols by multiple users in the same time. A key problem is how to correctly group the strokes under this complicated context. The Fig. 5.5 (a) shows an example where a 'rectangle' and an 'arrow' symbol are drawn simultaneously. The diagram in the right describes the temporal activity of their strokes. Obviously, the alternately emerging of the strokes from two users makes it more difficult to group the strokes. Meanwhile, since the symbol is allowed to be written in multi-touch manner, the synchronized strokes (as shown in Fig. 5.5 (b)) can hardly be determined whether they belong to one user or two. Approximately 65% strokes are simultaneously written in this dataset.

5.3 Eager interpretation based recognition system

From the practice point of view, a multi-user document composition system aims at providing a real-time feedback context so that users can easily exchange their ideas. We exploit eager interpretation, which updates the analyzed document after each input stroke and providing an instant corresponding feedback as illustrated in section 2.4. Ideally, a

Table 5.1: Symbols' distribution and their average and maximum stroke number in the complete dataset composed of 20 diagrams drawn by fingers and 22 drawn with stylus.

| | Symbols | Av. #strokes | Max. #strokes |
|-------------|---------|--------------|---------------|
| Bracket | 140 | 2.05 | 3 |
| Ellipse | 278 | 1.02 | 2 |
| Rectangle | 158 | 1.89 | 7 |
| Triangle | 88 | 1.39 | 5 |
| Diamond | 92 | 1.48 | 4 |
| Line | 446 | 1.00 | 2 |
| Double line | 140 | 2.02 | 3 |
| Arrow | 424 | 1.86 | 4 |
| Direction | 242 | 1.19 | 2 |

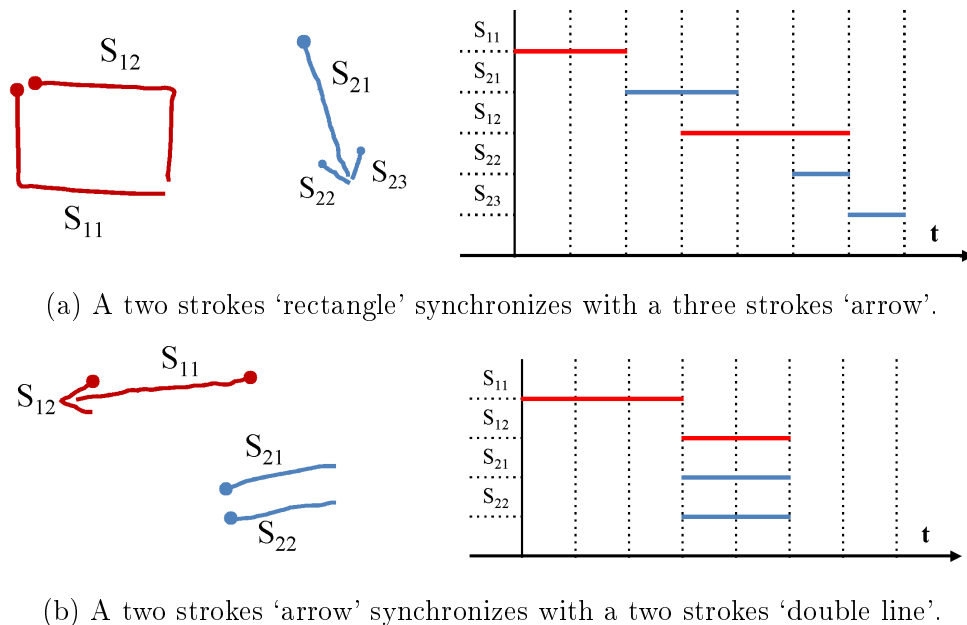


Figure 5.5: Example of temporal activity of strokes under 2 users condition. S_{ij} indicates the j th stroke from user i .

recognition result should be given on the screen in a short delay after the completion of each symbol. However, since the dataset contains multi-stroke gestures, the ending of a stroke is not exact the ending of a gesture. Therefore, the system need to adopt early recognition strategy to detect if input strokes can form a meaningful shape or need to wait for more strokes. On the other hand, the most important feature of this dataset is the mixture of multi-touch and multi-stroke symbols by multiple users in the same time. The alternately emerging of the strokes in the stroke stream from two users is so complex that two strokes concatenated in time domain may not even belong to a same user. Moreover, the existence of multi-touch gesture makes it more difficult that two synchronized strokes can belong to either a multi-touch gesture or two gestures from two users. Therefore,

instead of designing an explicit stroke grouping method, we use brute force grouping technique as a first attempt to recognize all combinations of strokes and let the classifier take the decision of selection the correct gesture thanks to a dedicated training. Although the computation cost would become very high when the number of candidate strokes is large, due to the fact that the eager interpretation system gives feedback in a short delay, the number of strokes in a short time window will not be so large for brute force computation.

Fig. 5.6 shows the global framework of our proposed eager interpretation system. After each end of a stroke, this stroke will be firstly stored in a stroke list. Then system will generate all combination of strokes if there exists multiple strokes in the list. A classifier is then used for each stroke combination to detect if any of them can form a concrete symbol. If not, strokes will keep in the stroke list and wait for more strokes from users.

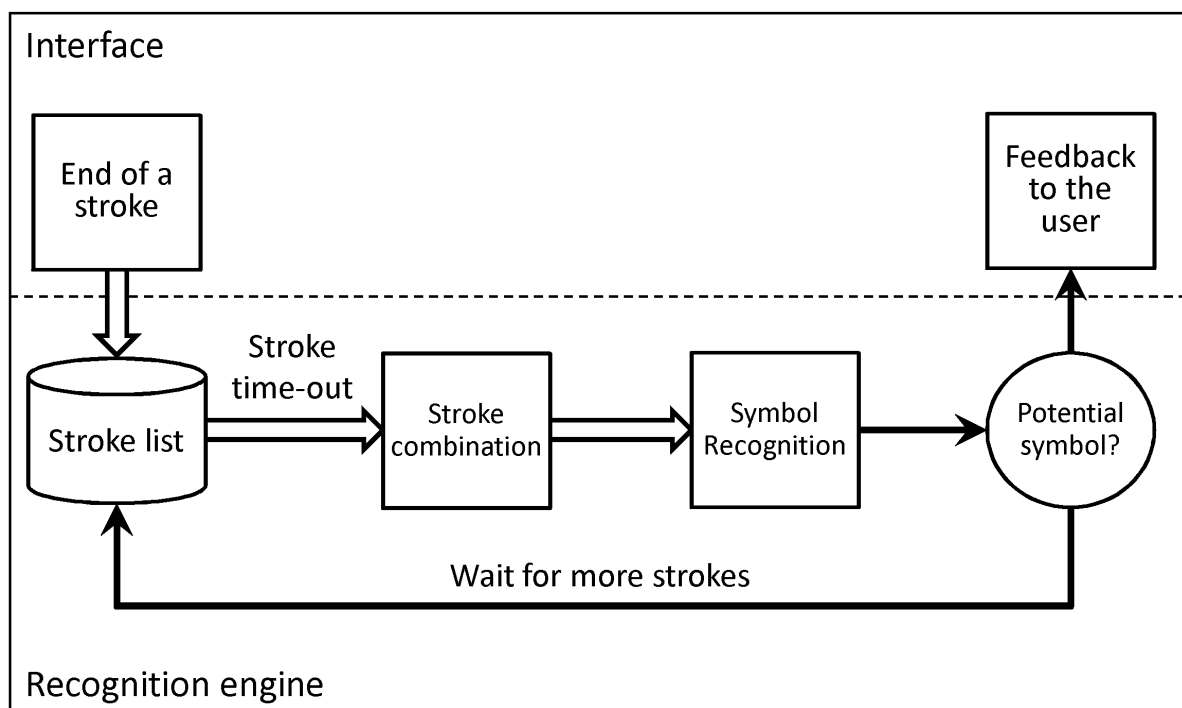


Figure 5.6: The framework of eager interpretation system.

Note that the classifier is responsible to not only recognize the correct symbol, but also need to reject the wrong combination and unfinished partial gesture. We therefore train the classifier with wrong combination sample and unfinished sample as negative symbol in addition to the original isolated symbol set. These two negative sample sets are generated as following:

Wrong Combination: a wrong combination is a stroke set whose strokes come from different gestures. Given a training gesture set $G = \{g_1, g_2, \dots, g_n\}$ where the subscript index follows the input stream of the gestures, each one composed of 1 or more strokes.

To generate this negative sample set, we extract random number of strokes among the strokes belonging to two consecutive gestures g_i and g_{i+1} . As a negative set, obviously it can not cover all the negative possibility. However, different size of this negative set can be generated to balance the ratio of positive (correct gestures) and negative (wrong segmentation) samples.

Unfinished Gesture: an unfinished gesture is a sub-stroke set of a completed gesture. Given a n stroke gesture $g = \{s_1, s_2, \dots, s_n\}$, we extract all its sub-stroke set following the input sequence as $\{s_1\}, \{s_1, s_2\}, \dots, \{s_1, s_2, \dots, s_{n-1}\}$. As the size of this set is comparable to the number of positive gestures (depending of the average number of strokes per gesture), we use them all. However, subsampling is possible to better balance the training set.

As shown in Fig.5.6, strokes which are recognized as wrong combination or unfinished gesture will be restored into the stroke list and wait for new strokes. We differ them as two independent negative classes because of a special situation: an unfinished gesture can be comparable to an isolated full symbol (shape and synchronization). Take a straight line for example, without context information, it can be recognized as either an isolated *line* symbol or the beginning part of a potential *rectangle* as unfinished gesture (Shown in Fig. 5.7).

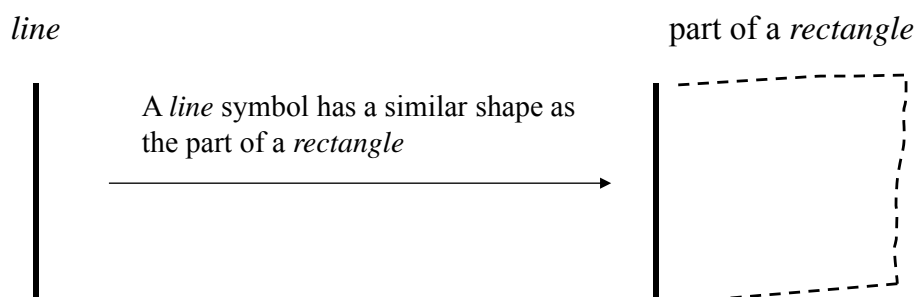


Figure 5.7: The decision for some shape should be postponed in case of forming another potential gesture.

From the classifier point of view, it is difficult to distinguish these two cases since they have the same appearance. From the practice point of view, it is reasonable to postpone the decision after the end of a stroke in case of forming another potential gesture with following strokes. To take into account this principle in the decision process, we can not use only the classifier outputs (as done in the previous chapter with reject option). Indeed, the stroke itself will not change in time, so if a potential conflict is detected thanks to the class scores (e.g. 'line' score and 'unfinished' score are close), this conflict will remain the same in future. If a symbol is completed by new strokes, then the new symbol can have a strong score and its strokes will be removed from the stroke list. If the partial symbol is never completed, it means that it is a full symbol and after a certain period the next decision should be used instead of 'unfinished'. That is why we propose to use the "age" of the symbol's strokes to weight the "unfinished" class score. By this way, young

strokes are more likely to be unfinished symbols and then the possibility for a symbol to stay unfinished decrease in time. The possibility then decreases with time elapsing. If the stroke can not be combined with others after a long time delay, the stroke itself is an isolated symbol that a decision has to be made for it. Therefore, we add a weight to the output of unfinished gesture class as:

$$p'_{un}(t_d) = \lambda(1 - \frac{t_d}{t_f}) * p_{un} \quad (5.1)$$

where p_{un} is the confidence value of unfinished gesture class output by classifier. λ is a constant value determining the increased amount of possibility at beginning. We empirically set $\lambda = 1.5$. t_d is the time delay between the stroke ending time and the recognition time. t_f is the final decision time at when the possibility will be decreased to 0 so that any other result of isolated symbol will be larger than the unfinished gesture class. An example is shown in Fig. 5.8.

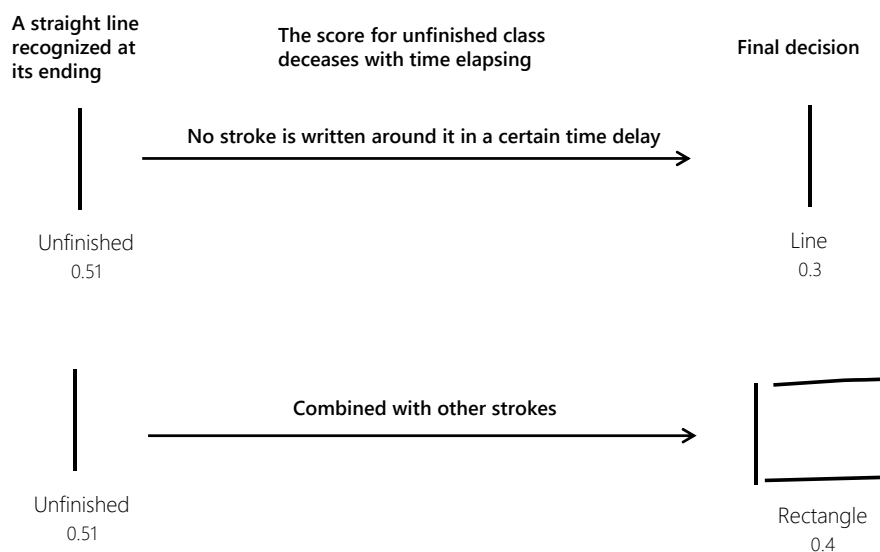


Figure 5.8: The score of unfinished class is manually increased at the ending time of a stroke and decreased with time elapsing.

A problem is how to select an appropriate t_f for the final decision time. A small t_f may force the system to make an early decision while the rest part of the symbol is still in writing. A larger one will postpone the decision and cause the stroke list to store too many strokes which makes more expensive to compute all combinations and increase the risk of generating false positive gestures. In the following experiment section we evaluate the recognition results with different values of t_f .

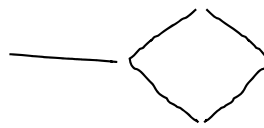
Fig. 5.9 gives a full view of stroke combination and recognition process. Taking 3 strokes for example, the figure shows all the 7 possible stroke combinations (candidates) and their corresponding recognition scores. Among all the 7 candidates, 5 possible groups

(shown in round squares) can be deduced. The score of the group is the average of all its candidates.

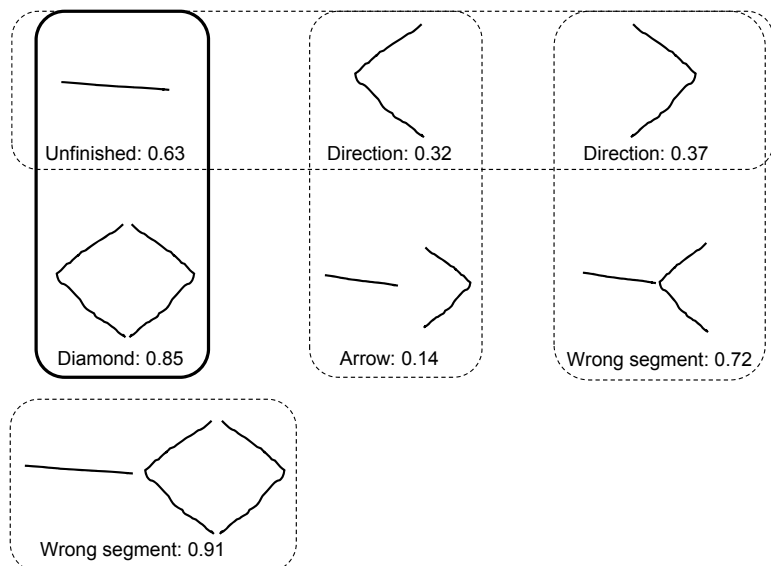
$$Score[group] = \frac{1}{N} \sum_i^N Score[Candidate_i] \quad (5.2)$$

The final decision is the group which has a largest average score of all its candidates. Note that the group whose all candidates are “Wrong Segment” will be discarded if there exists any other solid decision. Consequently, the Diamond and Unfinished gesture are chosen as the most confident results.

3 strokes in stroke list:



Independent 3 strokes:



2 strokes combinations:

3 strokes combinations:

Figure 5.9: An example of 3 strokes in the stroke list. All the 7 possible stroke’s combinations are shown in the figure. The round square shows the 5 possible decisions made for the 3 strokes. According to their score, an *unfinished* gesture and a *Diamond* gesture are chosen as the final decision. This unfinished stroke will be restored in the stroke list again.

5.4 Experiments

The multi-user diagram database is partitioned into two parts. 1346 gestures from 30 diagrams are used for training. 665 gestures from 12 diagrams are used for testing. Note that in a real testing environment for eager interpretation, users may be influenced by

the feedback such as mis-recognition or long recognition delay. Since the dataset are collected without giving any recognition feedback to the users, there is no noise in the stroke stream. We evaluate the performance of system in terms of correct stroke grouping rate and recall rate based on different values of parameter t_f . The underlying classifier is the Graph Modeling with Motion Based Features classifier illustrated in section 3.3 in order to well recognize the multi-touch gestures. The results are presented in Table 5.2.

Table 5.2: Correct stroke grouping and recognition rate

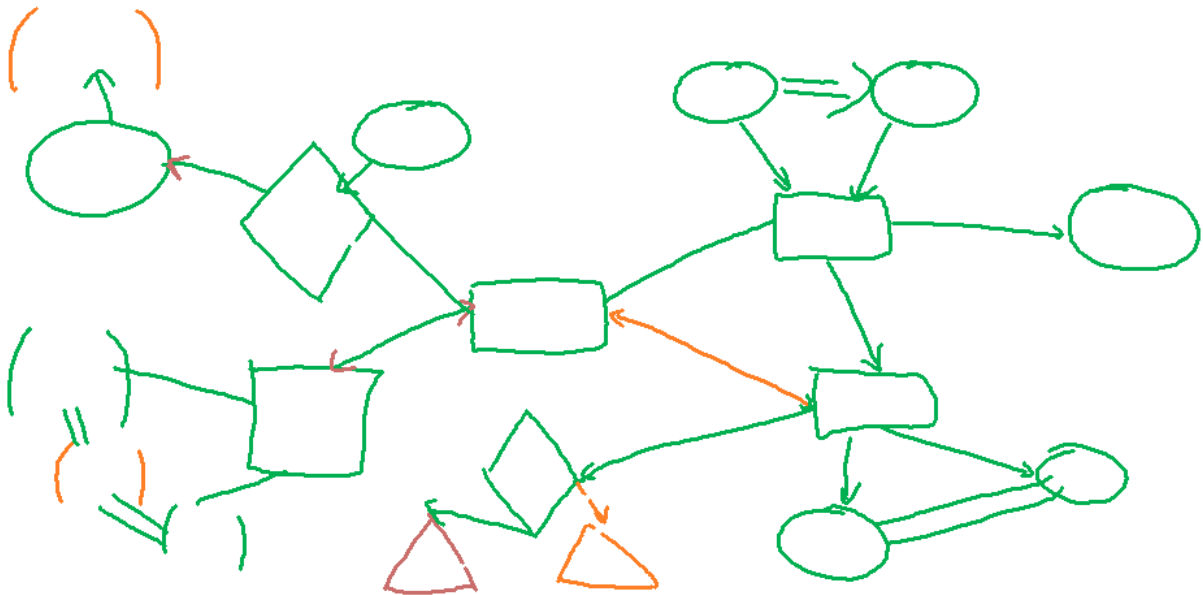
| $t_f(\text{ms})$ | 300 | 500 | 1000 | 2000 |
|---------------------|--------|---------------|--------|--------|
| Segment Number | 691 | 687 | 675 | 649 |
| Correct Segments | 592 | 596 | 593 | 578 |
| Segment Recall Rate | 89.43% | 90.03% | 89.58% | 87.31% |
| Symbol Recall Rate | 75.68% | 76.28% | 76.13% | 74.77% |

The performances in terms of stroke grouping rate are around 90%. It means that system makes 1 mis-grouping every 10 multi-stroke gestures and leads to a definite mis-recognition. The global recall rates are around 75% which is significantly lower the grouping rate. It means that with the correct stroke grouping, system results around 15% mis-recognition rate. As we illustrated in section 3.3.4.2, our Graph Modeling with Motion Based Features classifier is specially designed for multi-touch gestures. With the additional complex negative sample sets, it gives out an even low result.

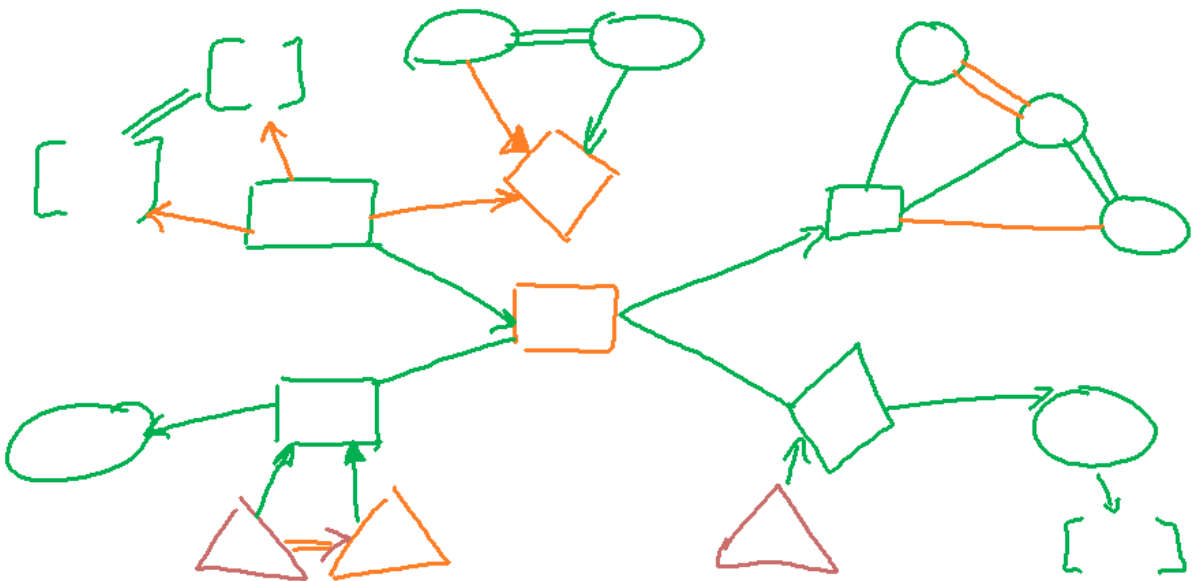
As far as final decision time t_f is concerned, the result at 500ms gives the best performance. It proves that neither too fast nor too slow could achieve a better result. An early decision may cause a partial gesture being recognized as a isolated symbol, while a later decision may cause a large stroke waiting list that leads to more mis-grouping.

Fig. 5.10 shows a screen shot of two full example diagrams. The green gestures are correctly grouped and recognized. Oranges are mis-grouped gestures. Most of them are multi-stroke gestures that wrongly being segmented into several parts. The reds ones are mis-recognized by classifier.

Fig. 5.11 shows the detail of the drawing of Fig. 5.10 (b) after 50 strokes drawn. Fig. 5.11 (b) shows the current stroke list and their decisions at each step. The group of strokes recognized as *unfinished* stay in the stroke list for next step. The group recognized as concrete symbol are removed from current list. We can notice that the stroke drawn at step 51 is recognized as *unfinished* for two steps until step 53 where it recognized as *line*. This illustrates the impact of “age” of the stroke on the decision (see equation 5.1). At step 55 the triangle is mis-segmented into two parts as a *line* and later (step 57) a *direction*.

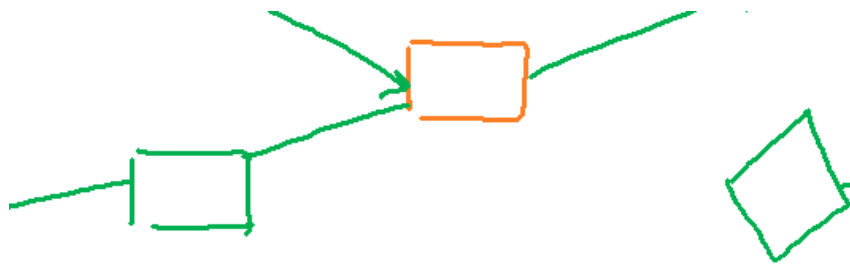


(a) Example of recognition results of layout 1.



(b) Example of recognition results of layout 2.

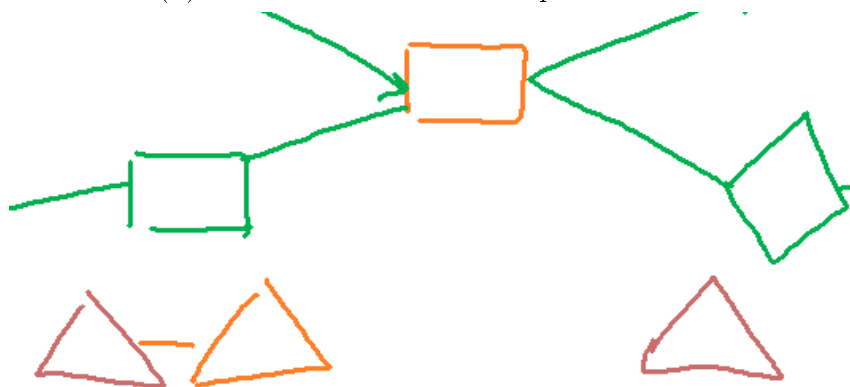
Figure 5.10: Recognition result for two example diagrams. Gestures in green are correctly recognized. Oranges are mis-grouped gestures. Reds are correctly grouped but mis-recognized gestures.



(a) Start context at step 50.

| | |
|---|---|
| Step 51 <i>unfinished</i> | Step 55 <i>unfinished</i> |
| Step 52 <i>unfinished</i> <i>unfinished</i> | Step 56 <i>unfinished</i> <i>diamond</i> |
| Step 53 <i>unfinished</i> <i>line</i> | Step 57 <i>unfinished</i> <i>direction</i> |
| Step 54 <i>diamond</i> <i>unfinished</i> | |

(b) Stroke stream between steps 51 and 57.



(c) Final context at step 57.

Figure 5.11: Stroke stream and their recognition results between steps 50 and 57 from Fig. 5.10 (b). The recognition decisions of each step are bounded in dashed rectangles.

5.5 Conclusion

We have presented a new dataset of multi-user handwritten diagram. The dataset contains a large quantity of graphic symbols drawn by 21 users. The most important feature is that each diagram is drawn by the collaboration of two users. The multi-user freely-drawn handwritten recognition is a challenging problem that few recognition systems attempt it. Our dataset opens a new frontier for the diagram recognition research. We present the difficulties to achieve a real-time stroke grouping and symbol recognition in multi-user context. Therefore, new questions are opened: How we can effectively group the strokes into distinct symbols? How we can give a real-time recognition feedback to the user?

To solve these problems, our first attempt is developing an eager interpretation system which launch the recognition at each end of a stroke. Instead of designing a explicit stroke grouping method to cope with the complex stroke relation for multi-stroke and multi-touch gestures, we employ the brute force stroke grouping method to traverse all stroke combinations in a short time window. The underlying classifier is trained with negative samples so that it can reject the wrong segmentation and unfinished gestures. A postponed parameter is employed for unfinished gesture class so that a partial stroke can wait for enough time before recognition in case of forming another potential gesture with following strokes. First experimental results shows that system can well recognize majority of gestures from the stroke stream.

There are several perspectives about this work. The first one is to exploit the complex stroke relations so that an explicit stroke grouping module can be achieved before gesture recognition. It would therefore simplify the classifier with less negative sample to detect. The second perspective is to involve different gestures command and manipulation such as *Erase*, *Translation* or even *zooming*, etc. i.e. to mix direct and indirect command in the gesture stream. To deal with direct command, the eager gesture recognition strategy should be merged in our global strategy, for example by not waiting the end of a stroke to try to recognize the gestures. The goal is to design an application that all commands and manipulations are achieved by gestures. As illustrated in section 5.1, more future work should be done following the features 3-6 involving context grammar and text recognition to achieve a complete application.

Chapter 6

Conclusion & Perspectives

6.1 Conclusion

The development of touch gesture operation facilities human computer interaction. Currently, multi-touch gestures have been mainly used and studied for direction manipulation such zooming, rotating, etc. There is few research studying to use multi-touch gesture to execute indirect commands. The work of this thesis aims at expanding the usage of multi-touch gestures and focuses more on the techniques of multi-touch gesture modeling and recognition.

To globally analyze and characterize a multi-touch gesture, the core problem is the synchronization relations, in both temporal and spatial domains, between touch trajectories. The main contribution of our work is the proposal of different graph modeling to integrate the shape information with temporal and motion relations between trajectories. The first graph-based approach extracts static shape feature on each trajectory and uses Allen’s relations as temporal description between the trajectories. We demonstrated that this modeling is simple and efficient to characterize and recognize multi-touch gestures. However it is not precise enough to capture the dynamic motion relations between trajectories.

We then progress to the second graph modeling which segments trajectories into small pieces and extracts synchronization features and dynamic motion features between each piece pair. To achieve the graph recognition, based on this gesture graph we studied the graph matching algorithm which finds the optimal vertex to vertex and edge to edge matching so that the cost of edit distance between two graphs can be calculated. Based on this edit distance, a graph embedding algorithm is then employed to transform the graph into a vectorial description to allow the use of classical pattern recognition classifiers.

To evaluate this graph modeling and recognition system for multi-touch gestures, we found that there is few available multi-touch gesture dataset for a benchmark test in the community. We therefore designed and collected the MTGSet which is a multi-touch gesture dataset containing different type of gestures in terms of different stroke number

and complex inner-stroke relations. This dataset is fully annotated and free available on the website. We demonstrate that our graph modeling recognition method can better recognize these multi-touch gestures comparing to other reference systems based on static features.

The second phase of our work concerns the development of using multi-touch gestures for commands. Due to the fact that usually multi-touch gestures are regularly used for direct manipulation, we extend the study of the mixed usage of multi-touch gestures, i.e. using multi-touch gestures for both direct manipulation and indirect command in a same context. We propose an early recognition system, which is based on a multi-classifier, able to recognize a gesture as soon as possible, so that the user can have a suitable feedback. Considering the ambiguous common part between gestures, classifiers are designed with a reject option so that it can detect these ambiguous beginning parts and avoid making an ambiguous decision without enough information. The system achieves 83.44% accuracy rate with 46.82% earliness of input gestures on MTGSet.

The final phase of our work is to design a structured document composition context. This work is supposed to develop a real application so that all the touch gesture interaction techniques can be verified in this context. In an ideal context, it allows multiple users to compose structured diagrams using multi-stroke gestures and manipulate the diagram using multi-touch gestures. At the beginning step of this research, we firstly collected a multi-user diagram composition dataset. We designed an eager recognition system aiming at recognizing gestures at every ending of a trajectory and providing a real-time feedback to the user. We achieved a promising result but still have a long way to go to use it for practice.

6.2 Perspectives

There is a significant tendency that large touch screens or even touch screen walls become more prevalent in our daily life. The goal of our research is to provide more freedom to the user to control the interaction system with different type of gestures. From the current study, we can suggest some perspectives to be achieved in a future work.

- The graph modeling need to involve more shape feature for a gesture. Our graph modeling is originally designed for multi-touch gestures which contain many dynamic features between strokes. According to the results on mono-touch gesture set, the involved shape features are too weak to recognize mono-touch gesture comparing to others. From the current graph modeling, one solution is to generate more substrokes for each stroke so that the shape will be represented more precisely. However, it will also significantly increase the scale of a graph that results in a more expensive graph matching.

- In early recognition, we empirically select the number of classifiers and each corresponding training segment length. In an ideal case, an early recognition system need to

automatically learn these parameters from training dataset so that it can find the optimal early decision time for each gesture. Our future work will focus on investigating an adaptive system to learn these information.

- Last but not least, the eager interpretation system for multi-user structured document composition need more improvements. On the one hand, a better stroke grouping strategy need to be explored. Instead of using brute force stroke combination, a spatial feature based preprocess is required to prune the number of stroke combination. On the other hand, as discussed above a more powerful isolated symbol classifier is essential to the current system. Moreover, with a better early recognition system, we will try to engage both direct manipulations and indirect commands into this application.

Résumé en Français

1. Introduction

Les interactions gestuelles tactiles font parties des interactions personne-machine les plus naturelles. Elles permettent une expérience utilisateur plus intuitive et plus appropriées que l'usage du clavier et de la souris. Avec le développement des écrans tactiles, un nombre croissant d'utilisateurs se familiarise avec ce type d'interactions gestuelles.

D'un point de vue interaction, les gestes tactiles sont utilisables selon deux modalités : la manipulation directe de certains éléments de l'interface et de façon complémentaire le déclenchement a posteriori d'une commande résultant de l'interprétation d'un geste symbolique que l'on qualifiera de commande indirecte. Un usage explicite d'une commande directe peut être donné en considérant le cas de la manipulation d'un plan cartographique où l'utilisateur pourra déplacer la carte par des glissements du doigt ou encore changer le zoom en écartant ou en resserrant deux doigts, faire pivoter la carte en appliquant une rotation d'un doigt autour d'un autre. Dans un cas général, une telle commande directe produit un retour continu au fur et à mesure de la production du geste initiateur. Ainsi, avec l'exemple du glissement de doigt (scroll) sur la carte, le déplacement de celle-ci se produit concomitamment avec celui-ci.

A l'inverse, les gestes de commandes indirectes produisent un effet unique, comme c'est le cas par exemple, avec la reconnaissance de l'écriture manuscrite. Des exemples de telles interactions sont proposés dans [AZ09] pour permettre de dessiner des icônes à partir de raccourcis gestuels. Un autre exemple classique est celui de la saisie manuscrite de caractères, dans ce cas l'utilisateur dessine la forme du caractère en place de le taper sur le clavier. Une fois le tracé effectué, le système de traitement l'interprète et produit un résultat associé, tel que l'affichage d'une lettre.

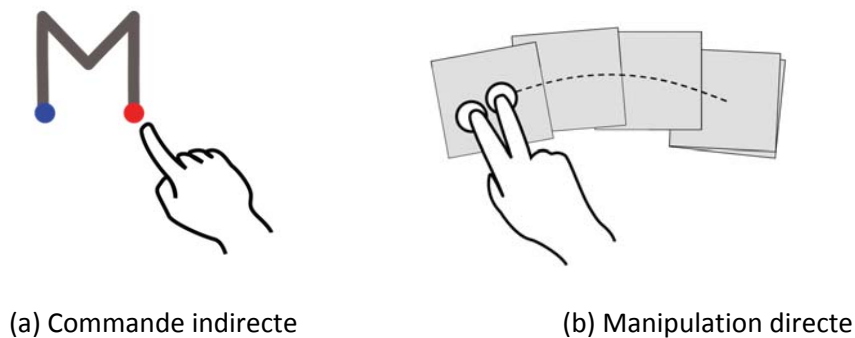


Figure 1. Deux types d'interaction tactile

De nos jours, les gestes tactiles mono-points (un seul point de contact à chaque instant) sont largement utilisés et étudiés à la fois pour de la manipulation directe et pour des commandes indirectes. Les utilisateurs ont même la possibilité de spécifier leurs propres gestes et de les associer aux commandes de leurs choix. Par contre, concernant les interactions multipoints, il existe de nombreuses limitations qui les contraignent à de la manipulation directe, telle que zoomer avec deux doigts. Le travail proposé dans cette thèse étend l'usage des gestes multipoints pour les rendre utilisable dans un cadre de commande indirecte (tel que copier, coller, etc.) et pour élargir les manipulations directes envisageables. Un des enjeux de notre problématique est de proposer des

modèles d'analyse et de reconnaissance aptes à traiter les gestes multipoints, comme il en existe aujourd'hui pour les gestes mono-points.

Pour modéliser les gestes multipoints, nous avons privilégié une approche basée graphe (section 2). Les caractéristiques du geste multipoint telles que sa forme, les relations spatiales et temporelles entre les trajectoires des doigts vont être prises en compte pour construire le graphe de représentation. Une description vectorielle du graphe (graph embedding) couplée à un classifieur SVM permettra d'assurer la reconnaissance du geste. Afin de mettre au point, d'entraîner et de tester nos différents systèmes, nous avons conçu, collecté et annoté une base d'exemples, MTGSet, qui réunit 7 938 gestes de 41 classes différentes. Pour traiter les problèmes d'ambiguïtés entre manipulation directe et commande indirecte sans connaissance du contexte, nous proposons une stratégie de reconnaissance précoce pour des gestes multipoints (section 3). L'objectif est de reconnaître le geste en considérant le début du tracé pour permettre un retour utilisateur avant la fin du tracé. Nous proposons également des stratégies d'implémentation pour traiter une application effective. Finalement, nous abordons les difficultés spécifiques de la composition multi-utilisateurs de documents dans un contexte multipoints (section 4). Dans ce cas, il s'agit pour reconnaître les gestes de chaque utilisateur de pouvoir distinguer les traces respectives produites par chacun.

2. Reconnaissance de gestes multipoints

A la différence des gestes mono-points où les tracés sont produits en séquence, dans le cas des gestes multipoints, les différents tracés ont des relations temporelles beaucoup plus complexes. Deux gestes ayant le même rendu visuel final peuvent avoir été produits de façon très différente. Nous chercherons à modéliser les relations inter-tracés telles qu'elles ont été produites afin de pouvoir distinguer ces différents types de geste. Compte-tenu du nombre variable de points de contacts, il est difficile d'extraire un nombre fixe de caractéristiques pour décrire les relations spatiales et temporelles entre les tracés. Pour cela, nous proposons d'utiliser un modèle à base de graphes pour représenter ces relations spatiales et temporelles, ainsi que l'aspect de chaque élément du tracé. Dans cette section, nous proposons deux types de graphes avec la méthode de classification associée pour permettre la reconnaissance du geste correspondant.

Modèle de graphe utilisant les relations d'Allen

La figure 2 présente le synoptique général de l'approche proposée.

Construction du graphe

La figure 3 montre le graphe pour un geste composé de deux traits. Chaque trait est représenté par trois nœuds, un nœud associé à chaque extrémité, V_b (begin) et V_e (end), et un nœud central V_s . Ces trois nœuds serviront pour introduire les relations entre les traits grâce aux arêtes qui en seront issues.

Ensuite, l'information de forme du trait est encodée dans le graphe. Nous utilisons pour cela un dictionnaire de formes élémentaires (ligne, arc, ellipse, ...) pour décrire de façon discrète chaque trait. À la différence d'autres travaux [LLLW15] [AMG07] où ces formes sont prédéfinies de façon empirique, nous avons construit le dictionnaire par un algorithme de clustering non supervisé à partir d'une base d'apprentissage. Pour cela, l'algorithme des K-means avec la distance Euclidienne dans

l'espace des caractéristiques HBF49 [DA13] est utilisé. Chaque trait est alors associé au représentant du cluster auquel il appartient. Dans la mesure, où seules des informations de formes locales à chaque trait sont utilisées, cette stratégie est appelée Local Shape Representation (**LSR**).

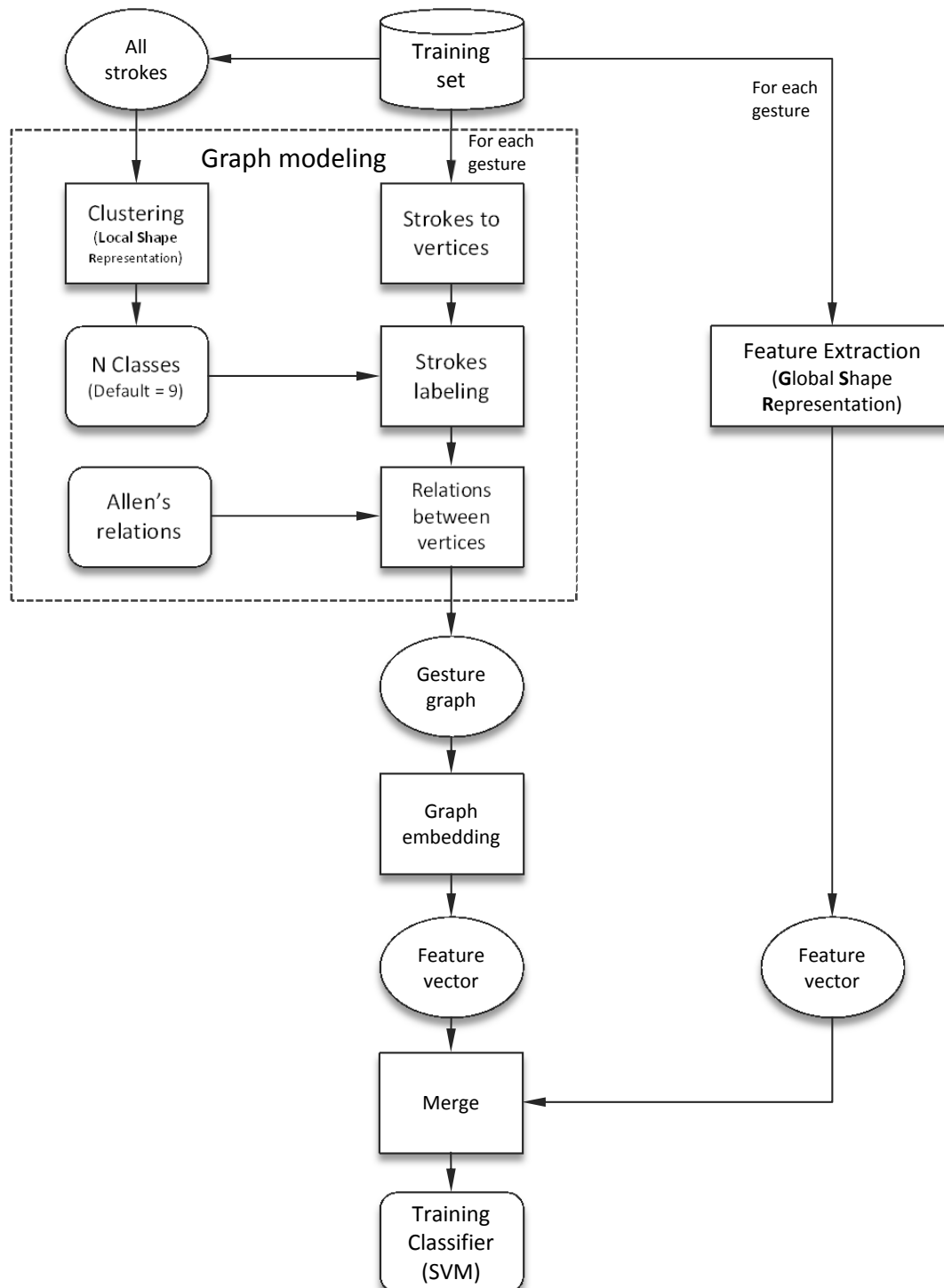


Figure 2. Structure générale de la construction du graphe et de sa reconnaissance.

Nous mesurons ensuite les relations spatiales et temporelles entre les différents traits. À titre d'exemple, considérons un geste avec deux traits dont le graphe général est donné figure 3. Sur ce

graphe, toutes les relations potentielles figurent et devront être évaluées. Ces relations sont représentées par les arêtes E_s , E_{st} and A_{st} entre deux nœuds du graphe. Nous avons choisi les relations d'Allen [All83] initialement utilisées pour quantifier 7 relations temporelles entre deux évènements. Sur ce principe, nous étendons les relations d'Allen à la fois sur l'axe du temps et sur les axes x et y pour évaluer des relations spatiales entre deux traits.

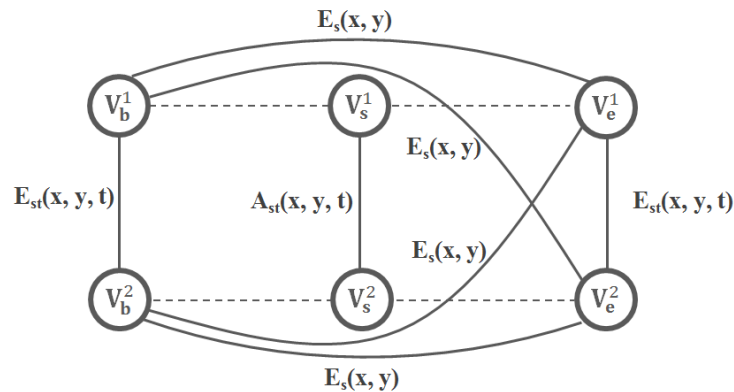


Figure 3. Structure générale d'un graphe d'un geste avec deux traits.

1) **Arête entre les nœuds centraux (A_{st}):** Le jeu complet des relations d'Allen est utilisé pour établir les relations vis-à-vis du temps, de la position en x et de la position en y entre deux traits V_s . L'indice st signifie que les relations spatiales et temporelles sont considérées. Il en résulte que l'arc A_{st} contient un triplet de relations comme attribut.

2) **Arêtes entre les nœuds extrémités (E_s and E_{st}):** Ces arêtes modélisent les relations existantes entre les extrémités des traits. A la différence des arêtes entre les nœuds centraux qui avaient un nombre fixe de trois relations, ici un nombre variable de 1 à 3 relations peuvent exister. Enfin, quand l'on considère une paire de nœuds extrémités opposés, seules les relations spatiales sont étudiées et l'arc se réduit à E_s . Cet arc porte alors les relations spatiales en x et en y entre un nœud début et un nœud fin de trait.

Il est à noter que les relations portées par les arêtes entre les nœuds extrémités renforcent les mesures établies par les relations d'Allen. Bien que certaines de ces relations soient déjà présentes implicitement dans les arêtes des nœuds centraux associés aux traits, nous choisissons de conserver cette redondance. Ainsi, des gestes avec le même nombre de traits pourront avoir des structures de graphe différentes.

Plongement de graphe

La méthode de « graph embedding » cherche à ramener dans un espace vectoriel de dimension fixe un graphe de topologie variable afin de pouvoir utiliser des méthodes classiques de classification statistique. Dans ce travail, nous avons adopté l'approche proposée par Sidere et al. [SHR09]. L'étape initiale consiste à construire une matrice dont chaque ligne représente une étiquette possible d'un sommet ou d'une arête du graphe tandis que chaque colonne correspond à une sous-structure choisie du graphe. Cette matrice sert d'accumulateur pour dénombrer les occurrences des labels dans les sous-graphes iso-morphiques aux sous-structures. Ensuite, la concaténation des lignes de la matrice permet d'obtenir la représentation vectorielle.

Pour les colonnes de la matrice, nous avons choisi de considérer trois sous-structures : un seul sommet, deux sommets reliés par une arête et trois sommets reliés par deux arêtes. Nous aurons un total de 32 lignes, représentant les 9 (par défaut) symboles de forme issus du clustering attribuables au sommet V_s , le label début et fin et pour les sommets extrémités V_b et V_e et les 7 relations d'Allen pour les trois relations E_{st} (temps, x et y). Conséquemment, un vecteur de taille $3 \times 32 = 96$ sera disponible à la suite de cette opération.

Représentation Globale de la Forme (Global Shape Representation, GSR)

Nous avons ci-dessus intégré la description locale (LSR) des traits à la représentation globale définie par le graphe. Une alternative à l'utilisation du graphe serait d'extraire directement du geste complet un vecteur de représentation global (GSR). À cet effet, nous extrayons globalement les caractéristiques HBF49 du geste complet, ces dernières pourront être considérées soit de façon isolée, soit en étant adjointes à celles provenant de la modélisation par graphe.

Expériences

Ces expériences mettent en œuvre la base MTGSetA, qui contient 1 800 gestes multipoints, représentant 18 classes et provenant de 10 personnes différentes. La figure 4 illustre les 18 classes de gestes.

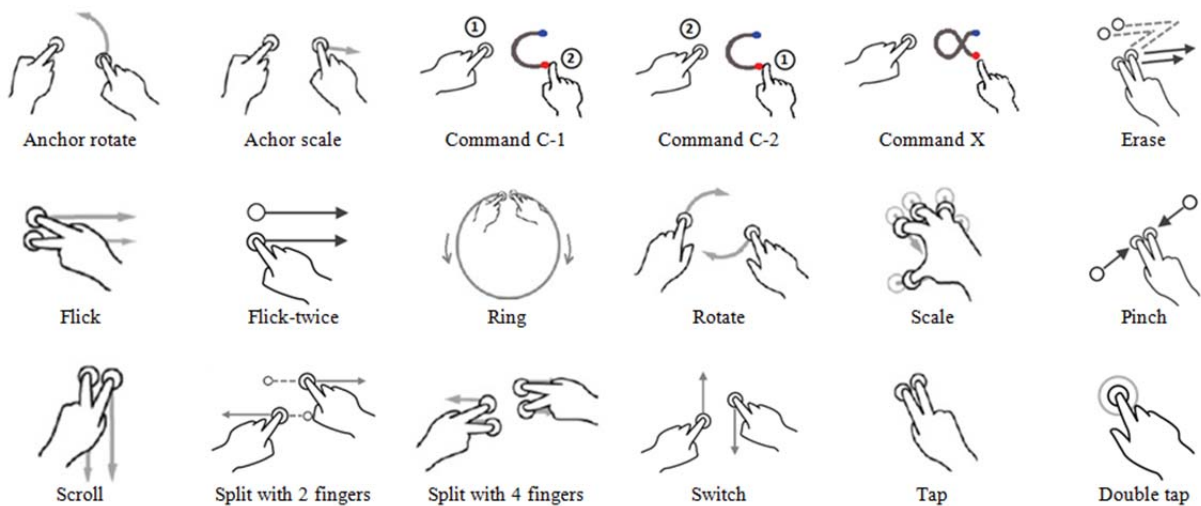


Figure 4. Pictogrammes des gestes de la base

Dans la mesure où les gestes tactiles ont été traduits par des vecteurs de caractéristiques, tous les classifieurs statistiques sont envisageables. Nous avons retenu pour cette expérience un classifieur SVM issu de la librairie LIBSVM en retenant un noyau gaussien et les méta-paramètres par défaut. Trois séries d'expériences sont proposées pour évaluer ce premier système.

Expérience 1 (GSR): seules les caractéristiques HBF49 en mode global sont utilisées. Celles-ci ont prouvé être très performantes dans des contextes mono-points. Nous les utilisons directement sur les tracés multipoints sans prendre en compte le modèle de graphe. En d'autres termes, dans cette expérience les conditions de synchronisation des tracés sont perdues.

Expérience 2 (Graph(LSR)): ici, nous utilisons le graphe pour modéliser le tracé puis sa description par un vecteur de taille fixe. Comme nous l'avons expliqué dans la section 3.2.1, l'approche LSR est

basée sur un clustering non supervisé pour quantifier la forme des traits, le nombre optimal de cluster a été fixé empiriquement à 9.

Expérience 3 (Graph(LSR)+GSR): Finalement, nous concaténons les deux représentations précédentes pour intégrer à la fois les descripteurs de formes locaux et globaux.

Toutes ces expériences sont faites avec un protocole de validation croisée (5-cross-validation) pour permettre d’être indépendant des utilisateurs.

Tableau 1. Résultats de reconnaissance sur la base MTGSetA

| Méthode | Taille du vecteur | Taux de reconnaissance (%) | Ecart type (%) |
|----------------|-------------------|----------------------------|----------------|
| GSR | 49 | 90.44 | 0.034 |
| Graph(LSR) | 96 | 92.56 | 0.013 |
| Graph(LSR)+GSR | 145 | 94.50 | 0.020 |

Le tableau 1 résume les taux de reconnaissance obtenus avec les deux approches et leur combinaison. Les résultats montrent que l’approche structurale par modèle de graphe, contenant des informations spatiales, temporelles et de formes locales, obtient un taux de reconnaissance de 92,56% ce qui est meilleur que celui obtenu avec les caractéristiques globales HBF49 (90,44 %). Cela montre que la modélisation proposée sous forme de graphe est bien apte à capter des informations importantes pour ces gestes multipoints. De façon complémentaire, la troisième expérience qui intègre les deux jeux de caractéristiques précédents obtient les meilleurs résultats avec un taux de 94,50 %, ce qui est significativement supérieur.

Modélisation par graphe avec des caractéristiques de déplacement

Le graphe précédent comporte certaines limitations. Les dépendances spatiales et temporelles ne sont prises en compte qu’aux extrémités des traits. De plus, le modèle ne retient que des descriptions symboliques (les relations d’Allen et les labels des traits) en tant qu’attribut du graphe. Ces descriptions discrètes ne sont sans doute pas suffisamment précises pour décrire les formes et les relations entre ces formes. Pour circonvier à ces limitations, nous proposons dans cette section une nouvelle approche qui intègre des attributs scalaires permettant une représentation plus continue de l’espace des caractéristiques. La figure 5 présente le schéma général de l’approche.

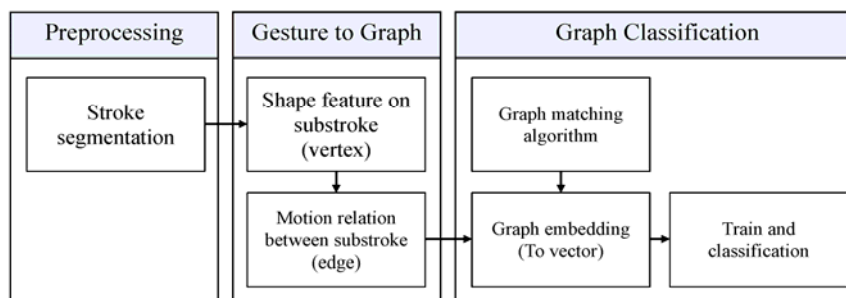


Figure 5. Approche de reconnaissance basée graphe en trois étapes

Prétraitement et découpage des traits

Il faut trouver un bon compromis entre extraire un mouvement global entre deux traits, ce qui est trop réducteur de la diversité des situations rencontrées, et extraire un mouvement local entre chaque point du tracé, ce qui serait très lourd et inutilement bruité. La solution proposée consiste à ré-échantillonner les traits avec un pas spatial prédéfini, et travailler avec les sous-traits ainsi définis entre chaque paire de points.

Du geste au graphe

A ce stade, de façon élémentaire, un geste est disponible sous la forme d'un ensemble de sous-traits. Dans cette section, nous verrons comment extraire deux informations importantes : la forme de chaque sous-trait et les relations topologiques entre paire de sous-traits. Ces informations sont portées par un graphe.

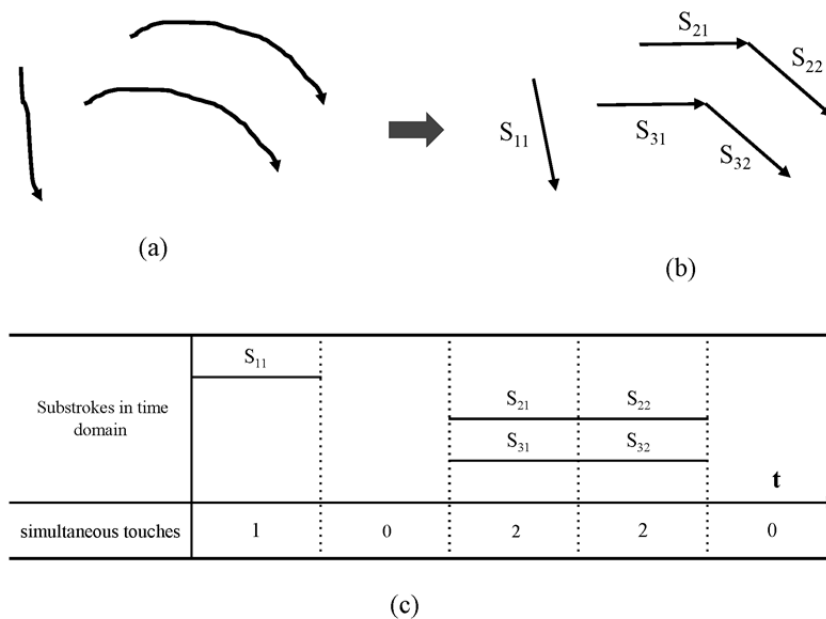


Figure 6. Segmentation des traits et diagramme d'activité des sous-traits.

Chaque sous-trait est défini par quatre paramètres géométriques, sa longueur (l), son inclinaison, l'angle (θ) et son centre de gravité (c_x, c_y)

Pour représenter l'évolution temporelle des sous-traits et leur mouvement relatif, nous introduisons trois types de relations entre deux sous-traits : les relations d'adjacence (a), de synchronicité (s) et de consécuité (c).

Relation 1 (Adjacence, R_a). Les sous-traits S_{ij} et S_{kl} sont adjacents s'ils appartiennent au même trait et sont temporellement consécutifs.

Avec l'exemple de la figure 6(c), nous avons : $R_a(S_{21}, S_{22}) = 1$; $R_a(S_{31}, S_{32}) = 1$. Cette relation permet d'identifier les sous-traits consécutifs d'un même trait.

Relation 2 (Synchronicité, R_s). Deux sous-traits S_{ij} et S_{kl} sont synchrones s'ils appartiennent à des traits différents et ont été tracés en même temps.

Cette relation indique que deux sous-trait ont été tracés simultanément par deux doigts, cela correspond à un cas typique pour une interaction multipoints. Un vecteur de caractéristiques va mesurer le mouvement relatif entre ces deux sous-trait. Une méthode classique [OIL11] pour caractériser le mouvement de deux doigts s'appuie sur les points de départ et d'arrivée de chaque sous-trait. On calculera les mouvements de translation, rotation et homothétie pour passer des points de départ aux points d'arrivée. La figure 7 illustre la définition de ces trois mouvements.

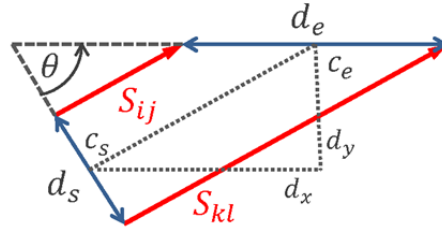


Figure 7. Caractéristiques de mouvement entre deux sous-trait synchrones S_{ij} , S_{kl} . Le vecteur de translation (d_x, d_y) est basé sur les centres de gravité c_s et c_e , des paires des points de départ et d'arrivée respectivement. Le facteur d'échelle (d_s, d_e) mesure les distances entre les points de départ et d'arrivée. L'angle θ définit la rotation permettant de passer de la direction du couple de points de départ au couple de points d'arrivée.

A partir de ces grandeurs, telles que définies par la figure 7, on construit le vecteur de mouvement suivant:

$$M(S_{ij}, S_{kl}) = (\sqrt{d_x^2 + d_y^2}, d_e - d_s, \theta/2\pi)$$

Relation 3 (Consécutivité, R_c). Deux traits S_i et S_k sont dits consécutifs, soit tracés en séquence quand ils ne partagent pas de sous-trait synchrones.

Dans la mesure où le modèle est construit au niveau des sous-trait, cette relation va être établie entre le dernier sous-trait d'un trait et le premier sous-trait d'un trait consécutif. Un attribut mesurant le retard temporel t_d entre les deux sous-trait est calculé

$$w_c(R_c) = w_c(S_{ij}, S_{kl}) = t_d/T$$

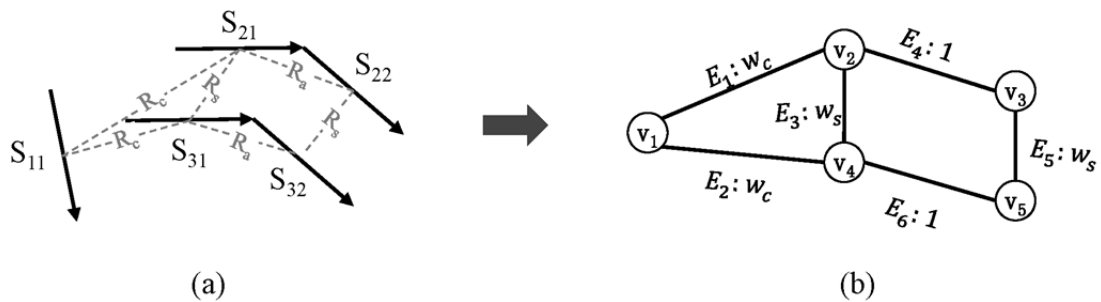
Cette grandeur est normalisée par la durée totale du geste pour la ramener dans l'intervalle $[0,1]$.

Globalement, l'ensemble des relations de l'exemple de la figure 6 est présenté sur la figure 8 (a). En utilisant les caractéristiques géométriques et les relations définies ci-dessus, nous pouvons construire un graphe, appelé Multi-Touch-Stroke Graph (MTSG), tel que présenté à la figure 8(b).

Mise en correspondance de graphes

Nous proposons de calculer une distance d'édition pour obtenir une mesure de (di-)similarité entre deux graphes représentatifs de deux gestes. Au lieu de réaliser une mise en correspondance globale sur les deux graphes entiers comme dans [RB09][RB10], nous cherchons d'abord une mise en correspondance optimale de sous-graphes en ne considérant que les sommets, les arêtes seront ensuite automatiquement ajoutées.

Nous commençons par grouper les sommets qui appartiennent au même trait pour former des ensembles de sous-graphes qui sont en fait dans ce cas des chaînes. L'algorithme DTW est utilisé pour calculer le coût d'association de deux de ces sous-graphes. Une matrice de coût C récapitule l'ensemble des coûts d'association des différentes paires possibles de sous-graphes. Puis l'algorithme de Munkres [Mun57] permet de trouver en temps polynomial l'ensemble optimal des assignations de sous-graphes. Il est possible alors de déduire la fonction de mise en correspondance entre chaque sommet des deux graphes. La distance d'édition se calcule alors simplement comme la distance euclidienne entre les vecteurs des sommets associés. La figure 9 présente la mise en correspondance des sous-graphes et la matrice de coûts C .



| Vertex | | |
|------------------------------------|--|-----------------------------|
| (v_1, \dots, v_5) | $F(v_1) = (0.25, 0.8, 0.1, 0.8)$ | Defined by Equ. 3.1 |
| | ⋮ | |
| Edge | | |
| Adjacent edge (E_4, E_6) | $E_4 = w_a(v_2, v_3) = 1;$ $E_6 = w_a(v_4, v_5) = 1;$ | Defined by Equ. 3.2 |
| Synchronous edge (E_3, E_5) | $E_3 = w_s(v_2, v_4) = (0.25, 0, 0);$ $E_5 = w_s(v_3, v_5) = (0.25, 0, 0);$ | Defined by Equ. 3.4 and 3.5 |
| Consecutive edge (E_1, E_2) | $E_1 = w_c(v_1, v_2) = 0.25;$ $E_2 = w_c(v_1, v_4) = 0.25;$ | Defined by Equ. 3.7 |

(c)

Figure 8. (a) Sous-trait correspondant à la figure 6(a) et leurs relations. (b) Le graphe correspondant. (c) Les attributs associés aux sommets et aux arêtes.

Une fois que la distance entre les sommets est calculée, le calcul est étendu en prenant en compte les arêtes adjacentes. Comme il y a trois types d'arêtes avec chacune des attributs différents, les coûts d'association ne sont faits que sur les arêtes de même type.

Classification basée graphe

Nous proposons une alternative à l'utilisation directe de la distance précédente par une méthode de K-ppv en reprenant l'idée proposée par Kapser and Horst [RB10] proposant une représentation vectorielle du graphe qui utilise la distance disponible. L'idée de base est d'utiliser la distance du

graphe à plusieurs formes prototypes pour construire un descripteur sous forme de vecteur. Un classifieur de type SVM est ensuite utilisée pour l'étape de classification du graphe.

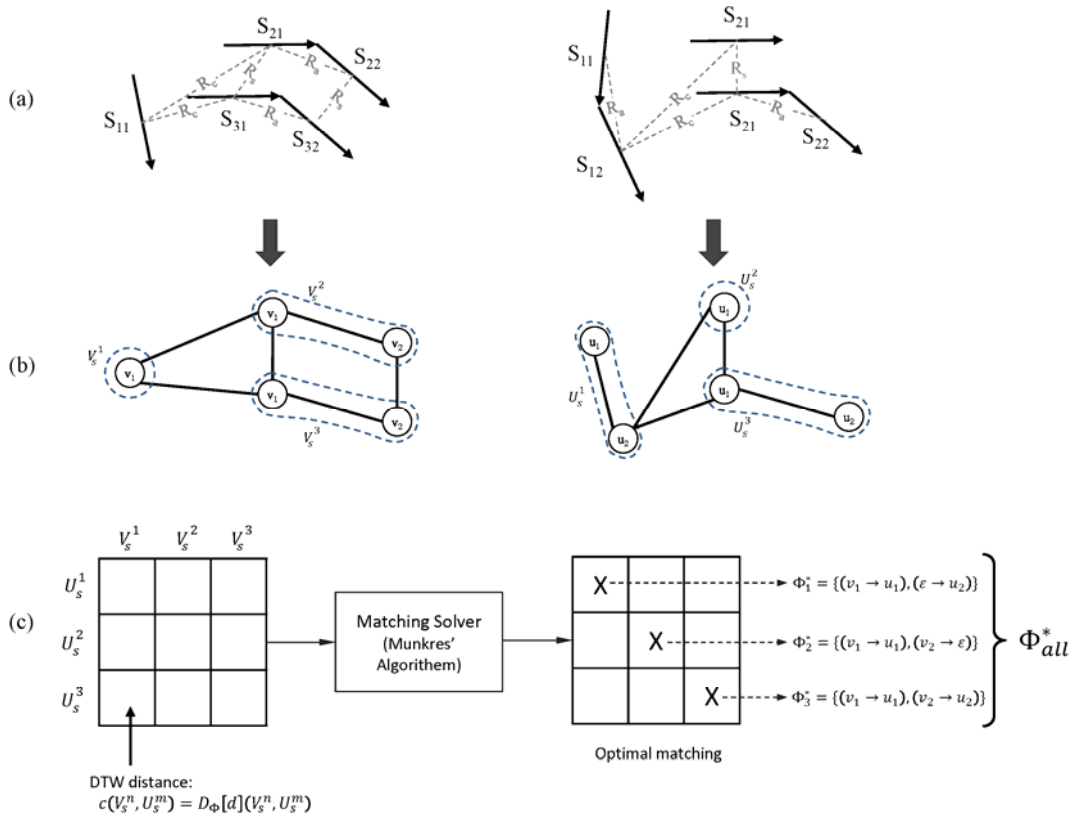


Figure 9. (a) Deux gestes avec les sous-traites et leurs relations. (b) Le graphe représentatif de chaque geste. Les sommets qui appartiennent au même trait sont regroupés dans les sous-graphes V_s^n, U_s^m . (c) L'assignation optimale de la matrice de coûts DTW solutionnée par l'algorithme de Munkres, et la mise en correspondance des sommets Φ_{all}^* .

Expérimentations

Nous effectuons les expérimentations sur la base MTGSetB, qui est une extension de MTGSetA. Nous ajoutons des classes plus complexes et étendons donc l'ensemble à 6138 échantillons et 31 classes. Dans le même temps, nous utilisons trois jeux de symboles mono-touche standard : LaViola, ILG et Niclcon, pour évaluer la performance de notre méthode sur les symboles mono-point. Nous comparons dans le Tableau 2 notre modélisation de graphe + plongement de graphe avec l'utilisation du jeu de caractéristiques HBF49 [4], qui est une approche statique conçue à l'origine pour le geste mono-point.

Tableau 2. Taux de reconnaissance pour les 4 bases de test

| | LaViola | ILG | Niclcon | MTGSetB |
|----------------|---------|--------|---------|---------|
| HBF49 | 93.64% | 93.54% | 97.44% | 91.36% |
| Graph modeling | 93.18% | 91.30% | 93.17% | 98.97% |

Nous constatons que l'ensemble de caractéristiques HBF49 fonctionne mieux sur les trois jeux de gestes mono-point. Les raisons sont que ces trois ensembles de données n'ont pas de relation de synchronisation complexe entre les traits et que les caractéristiques extraites pour chaque trait dans notre approche sont beaucoup plus simples que celles de HBF49. Par conséquent, notre approche est moins puissante sur l'aspect analyse de la forme par rapport à HBF49. Cependant, notre méthode de modélisation et d'appariement de graphes surpasse de manière significative le HBF49 sur MTGSetB. Principalement parce que le MTGSetB contient des gestes qui sont de forme similaire mais qui ont des relations temporelles internes différentes. Notre méthode est donc dédiée au problème de reconnaissance gestuelle multi-point.

3. Reconnaissance précoce basée sur l'utilisation du rejet

Dans la section précédente, nous avons proposé une méthode de classification pour reconnaître après leur achèvement les gestes multi-point de commandes indirectes. Si dans une même application, les utilisateurs doivent pouvoir utiliser des gestes multi-point en commande directe et aussi indirecte, un conflit entre ces deux interactions apparaît. En effet elles offrent une rétroaction complètement différente pour les utilisateurs. La manipulation directe doit donner une rétroaction instantanée pendant la trajectoire du doigt, tandis que la commande indirecte peut attendre la fin d'un geste, alors que le début des deux gestes peut être très similaire. Dans cette section, nous explorons une stratégie de reconnaissance précoce visant à reconnaître un geste dès le début afin de prendre une décision dès que possible.

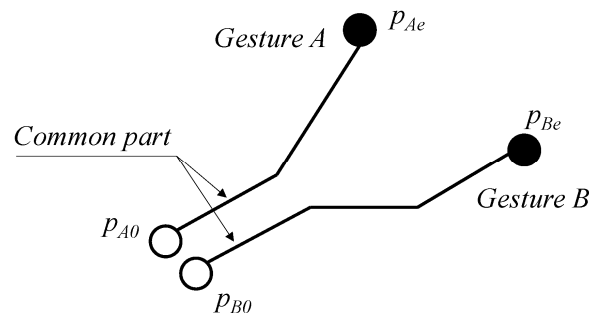


Figure 10. Deux gestes avec une partie commune ambiguë pour la reconnaissance précoce.

Reconnaissance précoce par combinaison de classifieurs

Nous proposons une structure multi-classifieur pour la reconnaissance précoce des gestes, comme le montre la figure 11. Chaque classifieur est entraîné sur une sous-partie du geste de sorte que les différents classifieurs sont responsables de différentes phases du geste. Dans un cas idéal, un geste d'entrée incomplet à venir (d'une longueur de l) est d'abord affecté au classificateur h_1 . S'il ne peut pas être distingué avec un autre geste en raison de leur début commun, on attendra une autre longueur de trajectoire et l'assignera au classificateur h_2 . Le même processus est utilisé pour les classifieurs suivants jusqu'à ce qu'un classificateur trouve suffisamment de différences pour faire une reconnaissance. Cependant, nous n'imposons pas de taille minimale ou maximale à un geste, il est impossible à partir d'un geste en cours de formation de savoir quelle proportion du geste est réalisée. Il peut donc y avoir conflit entre un début de geste et un autre geste complet. Par conséquent, dans notre système multi-classifieur, nous assignons chaque geste incomplet à tous les classifieurs et laissons un module de fusion pour décider si le geste peut être accepté ou non.

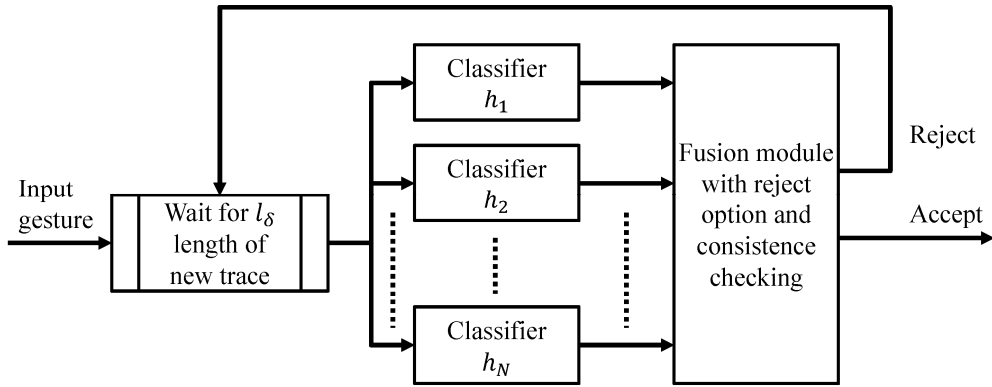


Figure 11. Architecture du système multi-classifieur pour la reconnaissance précoce.

Option de rejet

Sur la base de la structure multi-classifieur, nous mettons en œuvre deux options de rejet pour chaque classificateur: le rejet d'ambiguïté et le rejet de distance. Le premier est utilisé pour détecter la partie commune entre les gestes en conflits et permettre au système d'attendre plus de trajectoire. Le second est utilisé pour rejeter un geste partiel qui n'est pas compatible avec un des classificateurs et laisser les autres prendre une décision. La figure. 12 illustre la différence entre ces deux options de rejet.

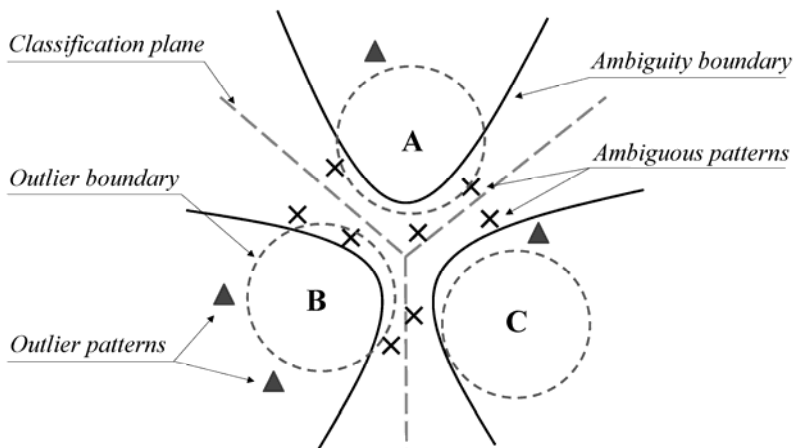


Figure 12. Différence entre le rejet d'ambiguïté et le rejet de distance.

Nos options de rejet sont basées sur les scores triés par ordre décroissant de chaque classe donnés par chaque classifieur. Les fonctions de confiance pour le rejet d'ambiguïté et le rejet de distance sont données par :

$$\psi_i^{Amb} = \frac{s_i - s_j}{s_j}$$

$$\psi_i^{Out} = s_i$$

Ces scores de confiance sont alors comparés à des seuils pour décider de l'acceptation ou non de la forme. Ces seuils sont appris sur une base de validation en minimisant le taux de fausse acceptation (FAR) et le taux de faux de rejet (FRR).

Expérimentations

Nous évaluons notre système sur la base MTGSetB qui contient 3589 gestes d'entraînement et 2549 gestes pour l'ensemble de test. Nous utilisons 20% des échantillons de l'ensemble d'entraînement comme un ensemble de validation pour l'apprentissage des seuils. Le nombre de classifieurs est fixé à 3. Avant l'apprentissage, chaque geste est normalisé dans un cadre de 500x500 pixels et segmenté en 3 gestes partiels de longueur 250, 500, 750 pixels pour alimenter respectivement les 3 classifieurs. Notez que chaque geste n'a pas plus de 750 pixels. Si un geste est inférieur à 500 pixels, il ne sera segmenté que sur deux gestes partiels et alimentera les deux premiers classifieurs. Lors du test, un geste est tenté d'être reconnu tous les 50 pixels de sa trajectoire. Dans le cas d'utilisation réelle, une stratégie raisonnable consiste à filtrer les décisions en considérant plusieurs sorties consécutives du classificateur (i.e. à différentes longueurs). Par conséquent, une décision est finalement acceptée lorsque le classificateur donne n fois consécutives la même sortie. Dans le tableau 3 nous comparons la précision de reconnaissance avec une stratégie sans rejet pour $n = 1$ à 6.

Tableau 3. Résultat pour la reconnaissance précoce de gestes avec ou sans utilisation du rejet pour n décisions identiques consécutives.

| n | Avec Rejet | | | | Sans Rejet | | |
|----------|-----------------------|-------------------------|---------------|---------------|------------------------|---------------|---------------|
| | Taux vrai acceptation | Taux fausse acceptation | Taux de Rejet | Précocité | Taux de Reconnaissance | Taux d'Erreur | Précocité |
| 1 | 81.89% | 14.56% | 3.54% | 37.04% | 24.88% | 75.15% | 8.13% |
| 2 | 83.44% | 10.85% | 5.71% | 46.82% | 48.78% | 51.22% | 21.32% |
| 3 | 82.38% | 8.85% | 8.77% | 55.89% | 67.60% | 32.40% | 33.98% |
| 4 | 82.20% | 6.06% | 11.73% | 66.16% | 79.59% | 20.36% | 45.44% |
| 5 | 80.35% | 4.60% | 15.05% | 71.03% | 85.83% | 13.72% | 54.93% |
| 6 | 77.42% | 3.41% | 19.17% | 77.54% | 88.62% | 11.39% | 62.34% |

Ici, le taux de rejet signifie que le geste partiel n'a pas n fois consécutives le même résultat ou a été rejeté par les trois classifieurs jusqu'à la fin. La précocité signifie le pourcentage moyen d'un geste qui est écrit au moment où il est reconnu. Le résultat pour $n = 3$ peut être considéré comme optimal lorsque les taux de fausse acceptation et de rejet sont équilibrés autour de 8,8% et la décision est prise en moyenne à 55,89% du tracé. Sans option de rejet, il faut monter à $n = 5$ pour un résultat comparable en terme de précocité (à 54,93%). Par contre, le taux d'erreur (13,72%) est supérieur de 55% au taux de fausse acceptation (8,85%) obtenu avec notre stratégie d'option de rejet.

4. Composition de document structurés dans le cadre multi-utilisateur

Dans cette dernière partie nous nous sommes intéressés à la problématique de la composition de documents lorsque plusieurs utilisateurs participent à cette tâche. Le point fondamentale qui change par rapport aux contextes des chapitres précédents est que plusieurs gestes peuvent être composés simultanément. Lorsqu'il y a plusieurs contacts en même temps, il n'est pas possible a priori de décider si ces traits composent un seul geste ou sont dessinés par deux utilisateurs. En plus de la tâche de reconnaissance il y a donc une tâche de segmentation de la séquence en gestes. De plus, pour garder une fluidité d'utilisation, il faut reconnaître un geste dès qu'il est terminé.

Base de diagrammes multi-utilisateurs

Pour réaliser cette étude, nous avons commencé par collecter une nouvelle base de gestes réalisés en condition multi-utilisateurs. Des diagrammes de type "cartes mentales" ont été recopiés (sans texte) par deux utilisateurs simultanément. 21 utilisateurs ont participé à collecter 42 diagrammes composés de 2011 gestes de 9 classes différentes. Certains symboles sont multi-points, d'autres multi-traités ou encore mono-traités. La Figure 13 donne un exemple de diagramme saisi.

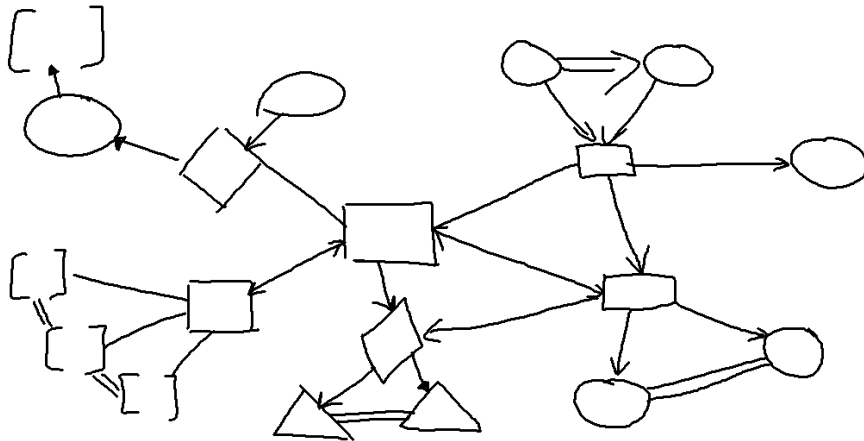


Figure 13. Exemple de diagramme multi-utilisateur composé de 9 gestes différents : crochets, ellipse, rectangle, triangle, losange, ligne simple, double ligne, flèche et pointe de flèche.

Système de reconnaissance

La figure 14 illustre le fonctionnement global du système. À chaque fin de trait saisi par un des utilisateurs, ce dernier est ajouté à une liste de trait. À partir de cette liste, nous allons générer tous les gestes possibles. Chaque hypothèse de geste est évaluée par un classifieur. Il y a trois types d'hypothèses qui peuvent être rencontrés : les gestes complets qui doivent être renvoyés à l'utilisateur ; les gestes incomplets qui doivent attendre d'être complétés ; et les mauvaises compositions de traits qui doivent être rejetées.

Les gestes détectés par le système sont renvoyés à l'utilisateur et leur traits sont retirés de la liste de traits en attente. Les traits restants seront ré-utilisés lorsqu'un nouveau trait sera saisi.

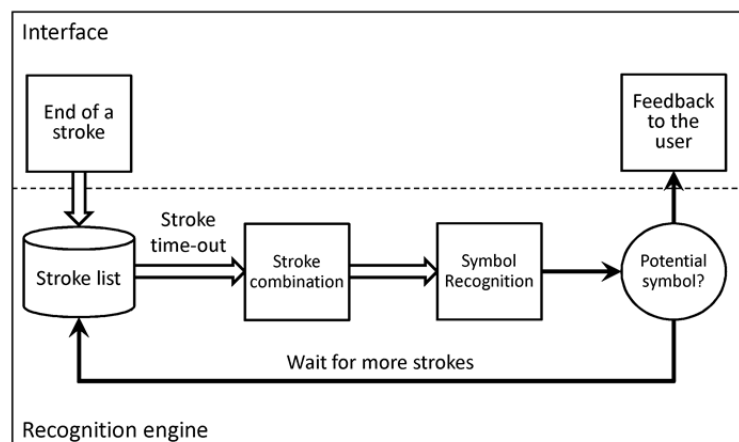


Figure 14. Architecture du système de reconnaissance multi-utilisateur.

Pour prendre ces trois types de décision, le classifieur est entraîné en utilisant une base de symboles valides, une base de symboles incomplets et une base de mauvaises combinaisons de traits. Comme évoqué dans la section précédente, certains gestes complets peuvent correspondre au début d'un geste plus long. Dans ce cas un conflit entre la classe du geste court et la classe "geste incomplet" bloque la reconnaissance du geste court. Pour éviter cette situation, nous proposons une pondération de la classe "geste incomplet" grâce à l'âge du trait concerné. En effet les traits composant un symbole complexe sont généralement dessinés dans un laps de temps assez court, donc un trait qui vient juste d'être saisi sera sûrement complété pour composer un symbole plus complexe, alors qu'un trait qui reste en attente depuis trop longtemps doit être utilisé.

Résultats

La tableau 4 montre les résultats obtenus en fonction de t_s l'âge maximum d'un trait non utilisé. Plus t_s est long, plus les gestes simples en conflit restent en attente longtemps. Si les traits sont utilisés trop tôt, certaines compositions sont manquées, si les gestes sont conservés trop longtemps, certains mauvais regroupements sont pris pour ces gestes. Le meilleur compromis est obtenu pour 500ms avec un taux de reconnaissance de 90%.

Tableau 4. Taux de bonne segmentation et taux de reconnaissance en fonction de t_s .

| t_s (ms) | 300 | 500 | 1000 | 2000 |
|---------------------|--------|--------|--------|--------|
| Segment Number | 691 | 687 | 675 | 649 |
| Correct Segment | 592 | 596 | 593 | 578 |
| Segment Recall Rate | 89.43% | 90.03% | 89.58% | 87.31% |
| Symbol Recall Rate | 75.68% | 76.28% | 76.13% | 74.77% |

5. Conclusion

Notre travail explore la possibilité d'utiliser le geste multi-point pour la manipulation directe et la commande indirecte. Nous avons développé un système de reconnaissance utilisant une modélisation par graphes pour reconnaître une commande multi-point comme un symbole isolé. Pour évaluer la performance de notre système, nous avons construit un jeu de données MTGSet, qui contient 6138 gestes multi-point avec des relations spatiales et temporelles internes complexes. Nous avons atteint 98,97% de taux de reconnaissance sur ce jeu de données et montré que notre système de reconnaissance peut bien capturer les propriétés des gestes multi-point en le comparant à d'autres approches classiques. Pour explorer la possibilité d'utiliser un geste multi-point pour la manipulation directe et la commande indirecte dans un même contexte, nous avons proposé une stratégie de reconnaissance précoce visant à reconnaître un geste à partir de sa partie initiale. Nous avons proposé une structure multi-classifieur avec option de rejet. Le résultat expérimental montre que nous avons atteint un taux de reconnaissance de 82,38% avec une précocité moyenne de 55,89% en conservant un taux d'erreur est inférieur au système sans option de rejet.

Enfin nous avons proposé une architecture de reconnaissance adaptée à la saisie de gestes tactiles multi-points par plusieurs utilisateurs simultanément. La solution proposée est entièrement entraînable à partir d'une base de gestes saisis en conditions réelles (une base dédiée a été collectée). Les résultats avec 9 classes de symboles pour la saisie de cartes mentales sont très stables, autour de 90% avec tous les gestes reconnus au plus tard après 500ms.

Publications of the author

Zhaoxin Chen, Eric Anquetil, Christian Viard-Gaudin, Harold Mouchère. Analyzing and recognizing multi-touch gestures using graph models. *Pattern Recognition*, submitted in November 2016.

Zhaoxin Chen, Eric Anquetil, Harold Mouchère, Christian Viard-Gaudin. A graph modeling strategy for multi-touch gesture recognition. *14th International Conference on Frontiers in Handwriting Recognition (ICFHR-2014)*, pp. 259-264, Sep 2014, Crete island, Greece.

Zhaoxin Chen, Eric Anquetil, Harold Mouchère, Christian Viard-Gaudin. Recognize multi-touch gestures by graph modeling and matching. *17th Biennial Conference of the International Graphonomics Society*, pp. 51-54 Jun 2015, Pointe-à-Pitre, France.

Zhaoxin Chen, Eric Anquetil, Harold Mouchère, Christian Viard-Gaudin. The MUMTDB dataset for evaluating simultaneous composition of structured documents in a multi-user and multi-touch environment. *15th International Conference on Frontiers in Handwriting Recognition (ICFHR 2016)*, pp. 379-383, Oct 2016, Shenzhen, China.

Zhaoxin Chen, Eric Anquetil, Harold Mouchère, Christian Viard-Gaudin. Early Recognition of Handwritten Gesture based on Multi-classifier Reject Option. *14th International Conference on Document Analysis and Recognition (ICDAR 2017)*, submitted in March 2017.

Bibliography

- [AFMVG11] Ahmad-Montaser Awal, Guihuan Feng, Harold Mouchère, and Christian Viard-Gaudin. First experiments on a new online handwritten flowchart database. In *Document Recognition and Retrieval XVIII*, pages 7874 – 78740A, San Fransisco, United States, Jan 2011.
- [All83] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.
- [AMG07] Thierry Artières, Sanparith Marukatat, and Patrick Gallinari. Online handwritten shape recognition using segmental hidden markov models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):205–217, 2007.
- [AW10] Lisa Anthony and Jacob O. Wobbrock. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, pages 245–252, 2010.
- [AZ09] Caroline Appert and Shumin Zhai. Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09*, pages 2289–2298, New York, NY, USA, 2009. ACM.
- [Blo96] Dorothea Blostein. *General diagram-recognition methodologies*, pages 106–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.
- [BPH13] Martin Bresler, Daniel Prusa, and Václav Hlavác. Modeling flowchart structure recognition as a max-sum problem. In *International Conference on Document Analysis and Recognition*, pages 1215–1219. IEEE Computer Society, 2013.
- [BPP⁺14] Martin Bresler, Truyen Van Phan, Daniel Prusa, Masaki Nakagawa, and Václav Hlavác. Recognition system for on-line sketched diagrams. In *14th International Conference on Frontiers in Handwriting Recognition, , Crete, Greece, September 1-4*, pages 563–568, 2014.

- [CAMVG15] Zhaoxin Chen, Eric Anquetil, Harold Mouchère, and Christian Viard-Gaudin. Recognize multi-touch gestures by graph modeling and matching. In *17th Biennial Conference of the International Graphonomics Society*, Pointe-à-Pitre, France, June 2015.
- [CLC13] Cérés Carton, Aurélie Lemaitre, and Bertrand Coüasnon. Fusion of statistical and structural information for flowchart recognition. In *12th International Conference on Document Analysis and Recognition, Washington, DC, USA, August 25-28*, pages 1210–1214, 2013.
- [DA13] Adrien Delaye and Éric Anquetil. Hbf49 feature set: A first unified baseline for online symbol recognition. *Pattern Recognition*, 46(1):117–130, 2013.
- [DLJZ07] Kai Ding, Zhibin Liu, Lianwen Jin, and Xinghua Zhu. A comparative study of gabor feature and gradient feature for handwritten chinese character recognition. In *2007 International Conference on Wavelet Analysis and Pattern Recognition*, volume 3, pages 1182–1186, Nov 2007.
- [FFJ11] Manuel J. Fonseca, Alfredo Ferreira, and Joaquim A. Jorge. *Sketch-based Interfaces and Modeling*, chapter Sketch-based Retrieval of Vector Drawings, pages 181–201. Springer London, London, 2011.
- [HR07] Suyu Hou and Karthik Ramani. Calligraphic interfaces: Classifier combination for sketch-based 3d part retrieval. *Comput. Graph.*, 31(4):598–609, August 2007.
- [ISS10] Katsuhiko Ishiguro, Hiroshi Sawada, and Hitoshi Sakano. Multi-class boosting for early classification of sequences. In *Proceedings of the British Machine Vision Conference*, pages 24.1–24.10. BMVA Press, 2010. doi:10.5244/C.24.24.
- [JZ07a] J. J. LaViola Jr. and R. C. Zeleznik. A practical approach for writer-dependent symbol recognition using a writer-independent symbol recognizer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1917–1926, Nov 2007.
- [JZ07b] Joseph J. LaViola Jr. and Robert C. Zeleznik. A practical approach for writer-dependent symbol recognition using a writer-independent symbol recognizer. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1917–1926, 2007.
- [KHDA12] Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. Proton++: a customizable declarative multitouch framework. In *The 25th Annual ACM Symposium on User Interface Software and Technology, UIST '12, Cambridge, MA, USA, October 7-10, 2012*, pages 477–486, 2012.

- [KSNT11] M. Kawashima, A. Shimada, H. Nagahara, and R. I. Taniguchi. Adaptive template method for early recognition of gestures. In *Frontiers of Computer Vision (FCV), 2011 17th Korea-Japan Joint Workshop on*, pages 1–6, Feb 2011.
- [KWK⁺10] Dietrich Kammer, Jan Wojdziak, Mandy Keck, Rainer Groh, and Severin Taranko. Towards a formalization of multi-touch gestures. In *ACM International Conference on Interactive Tabletops and Surfaces, ITS 2010, Saarbrücken, Germany, November 7-10, 2010*, pages 49–58, 2010.
- [LC02] Wing Ho Leung and Tsuhan Chen. Retrieval of sketches based on spatial relation between strokes. In *ICIP (1)*, pages 908–911, 2002.
- [LLLW15] Shuang Liang, Jun Luo, Wenyin Liu, and Yichen Wei. Sketch matching on topology product graph. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1723–1729, 2015.
- [LM01] James A. Landay and Brad A. Myers. Sketching interfaces: Toward more human interface design. *IEEE Computer*, 34(3):56–64, 2001.
- [MA06] H. Mouchere and E. Anquetil. A unified strategy to deal with different natures of reject. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 792–795, 2006.
- [MA09] Sébastien Macé and Eric Anquetil. Eager interpretation of on-line hand-drawn structured documents: The dali methodology. *Pattern Recogn.*, 42(12):3202–3214, December 2009.
- [MMM⁺12] Sriganesh Madhvanath, Dinesh Mandalapu, Tarun Madan, Naznin Rao, and Ramesh Kozhissery. Gecco: Finger gesture-based command and control for touch interfaces. In *4th International Conference on Intelligent Human Computer Interaction, IHCI 2012, Kharagpur, India, December 27-29, 2012*, pages 1–6, 2012.
- [MRLSL06] Joan Mas Romeu, Bart Lamiroy, Gemma Sánchez, and Josep Lladós. Automatic Adjacency Grammar Generation from User Drawn Sketches. In *18th International Conference on Pattern Recognition - ICPR 2006*, volume 2, pages 1026 – 1029. IAPR, August 2006.
- [MUK⁺06] A. Mori, S. Uchida, R. Kurazume, R. Taniguchi, T. Hasegawa, and H. Sakoe. Early recognition and prediction of gestures. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 560–563, 2006.

- [Mun57] James R. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, March 1957.
- [NV05] Ralph Niels and Louis Vuurpijl. Dynamic time warping applied to tamil character recognition. 2005.
- [NWW08] R. Niels, D.J.M. Willems, and L. Vuurpijl. The nicicon database of hand-written icons. In *Proceedings of the 1st International Conference on Frontiers in Handwriting Recognition*, pages 296–301, Montreal, Canada, 2008.
- [OIL11] Chi-Min Oh, Md Zahidul Islam, and Chil-Woo Lee. Mrf-based particle filters for multi-touch tracking and gesture likelihoods. In *11th IEEE International Conference on Computer and Information Technology, CIT 2011, Pafos, Cyprus, 31 August-2 September 2011*, pages 144–149, 2011.
- [ORB⁺15] Francisco R. Ortega, Naphtali Rishé, Armando Barreto, Fatemeh Abyarjoo, and Malek Adjouadi. *Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering*, chapter Multi-Touch Gesture Recognition Using Feature Extraction, pages 291–296. Springer International Publishing, Cham, 2015.
- [PM13] Eric Petit and Christophe Maldivi. Unifying gestures and direct manipulation in touchscreen interfaces. December 2013.
- [PSDA10] Eric Jeffrey Peterson, Thomas F. Stahovich, Eric Doi, and Christine Alvarado. Grouping strokes into shapes in hand-drawn diagrams. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 974–979, 2010.
- [RB09] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput.*, 27(7):950–959, 2009.
- [RB10] Kaspar Riesen and Horst Bunke. *Graph Classification and Clustering Based on Vector Space Embedding*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2010.
- [RFLDA12] Ney Renau-Ferrer, Peiyu Li, Adrien Delaye, and Eric Anquetil. The ILGDB database of realistic pen-based gestural commands. In *ICPR2012 - 21st International Conference on Pattern Recognition*, pages 3741–3744, Tsukuba, Japan, 2012.

- [RNB07] Kaspar Riesen, Michel Neuhaus, and Horst Bunke. *Graph-Based Representations in Pattern Recognition: 6th IAPR-TC-15 International Workshop, GbRPR 2007, Alicante, Spain, June 11-13, 2007. Proceedings*, chapter Bipartite Graph Matching for Computing the Edit Distance of Graphs, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [Rub91] Dean Rubine. Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '91*, pages 329–337, New York, NY, USA, 1991. ACM.
- [SBMI12] Napa Sae-Bae, Nasir Memon, and Katherine Isbister. Investigating multi-touch gestures as a novel biometric modality. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 156–161. IEEE, 2012.
- [SBMIA14] Napa Sae-Bae, Nasir D. Memon, Katherine Isbister, and Kowsar Ahmed. Multitouch gesture-based authentication. *IEEE Transactions on Information Forensics and Security*, 9(4):568–582, 2014.
- [SF10] Pedro Sousa and Manuel J. Fonseca. Sketch-based retrieval of drawings using spatial proximity. *J. Vis. Lang. Comput.*, 21(2):69–80, April 2010.
- [SHR09] Nicolas Sidere, Pierre Héroux, and Jean-Yves Ramel. Vector Representation of Graphs: Application to the Classification of Symbols and Letters. In *ICDAR*, pages 681–685. IEEE Computer Society, 2009.
- [SKC08] Fotini Simistira, Vassilis Katsouros, and George Carayannis. A Template Matching Distance for Recognition of On-Line Mathematical Symbols. In Fotini Simistira; Vassilis Katsouros; George Carayannis, editor, *11th International Conference on Frontiers in Handwriting Recognition (ICFHR 2008)*, Quebec, Canada, 2008.
- [Sup16] <https://support.apple.com/en-us/ht204895>, 2016.
- [Sut63] Ivan E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference, AFIPS '63 (Spring)*, pages 329–346, New York, NY, USA, 1963. ACM.
- [SW13] Michael Schmidt and Gerhard Weber. Template based classification of multi-touch gestures. *Pattern Recogn.*, 46(9):2487–2496, September 2013.
- [UA08] S. Uchida and K. Amamoto. Early recognition of sequential patterns by classifier combination. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, Dec 2008.

- [VLOK01] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. Experiments with adaptation strategies for a prototype-based recognition system for isolated handwritten characters. *International Journal on Document Analysis and Recognition*, 3(3):150–159, 2001.
- [WNvGV09] Don Willems, Ralph Niels, Marcel van Gerven, and Louis Vuurpijl. Iconic and multi-stroke gesture recognition. *Pattern Recognition*, 42(12):3303–3312, 2009.
- [WWL07] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: A \$1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.
- [XWJS02] Jin Xiangyu, Liu Wenyin, Sun Jianyong, and Zhengxing Sun. On-line Graphics Recognition. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, pages 256–264, 2002.
- [ZSHN10] Yuanping Zhu, Jun Sun, Yoshinobu Hotta, and Satoshi Naoi. Rejection optimization based on threshold mapping for offline handwritten chinese character recognition. In *ICFHR*, pages 72–77. IEEE Computer Society, 2010.

List of Figures

| | | |
|------|---|----|
| 1.1 | Touch gesture based manipulation interface. | 5 |
| 2.1 | Handwritten touch gesture for indirect command: (a) Users may input an icon from the menu. The menu also shows the mapping from strokes to icons. (b) Using stroke as the shortcut to input the icon instead of selecting from menu. [AZ09] | 10 |
| 2.2 | (a) A surface gesture taxonomy based on the number of strokes and touches. [SW13] (b) Examples of different type of gestures. | 11 |
| 2.3 | Indirect gesture commands supported by Mac’s trackpad. | 12 |
| 2.4 | Gestures in feature space. (a) Two gestures have no common part. (b) Middle parts are common. (c) Beginning parts are common. [MUK+06] . . | 13 |
| 2.5 | Single touch gesture set for executing commands. [WWL07] | 14 |
| 2.6 | Example of a DTW matching. [NV05] | 15 |
| 2.7 | Features used to identify strokes. [Rub91] | 16 |
| 2.8 | Example of all permutations for a two-stroke “ x ” and its unistroke representations. [AW10] | 18 |
| 2.9 | (a) Examples of architectural plans in [MRLSL06]. (b) Example of engineering drawing in [LLLW15]. | 19 |
| 2.10 | Example of stroke refinement.[XWJS02] | 20 |
| 2.11 | Examples of topology relations between primitives.[LLLW15] | 20 |
| 2.12 | A sketch and its topology graph with the relations and geometry feature on edges. R_{ad} means adjacency relation for two primitives which have a common endpoint. R_{hc} means half-cross relation that one primitive has a endpoint joining some inner point of another primitive. [LLLW15] | 21 |
| 2.13 | The displacement of three touch points from time $t - 1$ to t can be translation, scaling and rotation simultaneously. The f_1 , f_2 and f_3 are feature functions which are respectively related to translation, scaling and rotation. [OIL11] | 22 |

| | | |
|------|---|----|
| 2.14 | The motion feature functions: f_1 measures the translation vector of i^{th} point pair between time $t - 1$ and t . f_2 measures the distance $d_{i,j}$ between points in a certain time. f_3 measures the rotation angle θ_i of i^{th} node between time $t - 1$ and t . [OIL11] | 23 |
| 2.15 | Rotate gesture described by GeForMT, where $1F$ means number of finger, <i>HOLD</i> and <i>SEMICIRCLE</i> are predefine atomic gestures, o means the finger is focusing on an object. [KWK ⁺ 10] | 24 |
| 2.16 | Examples of indirect command oriented multi-touch gestures presented in [SW13]. Larger dots are depicting the start of a trajectory (from one touch), the arrows their movement and dashed smaller dots symbolize their end. Different strokes are colored differently, black elements belong to the first stroke, gray ones to the second. | 25 |
| 2.17 | The overview of a multi-touch symbolic gesture classifier's architecture presented in [SW13]. | 26 |
| 2.18 | An example of early recognition for a gesture, the recognizer give different feedback according to the progressing of trajectory. [PM13]. | 27 |
| 2.19 | (a) Conventional gesture recognition. (b) Early recognition strategy. [MUK ⁺ 06]. | 28 |
| 2.20 | Three frame classifiers combined with weight propagation. [UA08]. | 29 |
| 2.21 | (a) Examples of diagrams with structure in [BPP ⁺ 14]. (b) A digital logic sketch in [PSDA10]. | 30 |
| 2.22 | (a) Lazy interpretation of musical score. (b) Eager interpretation of musical score [MA09]. | 31 |
| 3.1 | The graph modeling and classifier training architecture of gesture recognition system. | 35 |
| 3.2 | A stroke is represented by three vertices in a graph. V_b : begin vertex; V_s : stroke vertex; V_e : end vertex. | 36 |
| 3.3 | Examples of seven Allen's relations. | 37 |
| 3.4 | An example of a general graph modeling for a two strokes gesture. | 37 |
| 3.5 | a) A flick gesture. b) Spatial relationship between strokes. c) Temporal relationship between strokes. d) Primitive set from clustering. e) Graph model with labels. | 38 |
| 3.6 | a) A anchor rotate gesture. b) Spatial relationship between strokes. c) Temporal relationship between strokes. d) Primitive set from clustering. e) Graph model with labels. | 39 |
| 3.7 | Vectorial representation of the graph in 3.5(e). | 40 |
| 3.8 | Multi-touch gestures prototypes in our experimental dataset. | 42 |
| 3.9 | (a) We record some basic information from the users. (b) Data acquisition tool. In the top right, a animation is used to show the groundtruth of a multi-touch gesture. | 42 |

| | | |
|------|--|----|
| 3.10 | Recognition rate obtained by different modules. | 44 |
| 3.11 | Confusion matrix of some typical misclassified gestures of different classification methods. The row relates to the ground truth. | 45 |
| 3.12 | Performance evaluation of different values of cluster number K used for LSR | 46 |
| 3.13 | Overview of the graph based gesture recognition in three stages. | 47 |
| 3.14 | Stroke segmentation and substroke representation. (a) a raw bracket like gesture; (b) the gesture is segmented into four substrokes and normalized inside a unit square bounding box. Each substroke has a feature vector composed of its length (l), angle (θ) and centroid (c). | 48 |
| 3.15 | Example of temporal activity of substrokes in a multi-touch interaction. (a) A three strokes gesture. (b) The substroke representation, S_{ij} indicates the j th substroke of stroke i . (c) Temporal activity of substrokes. | 49 |
| 3.16 | Motion features of two synchronous substrokes S_{ij}, S_{kl} . Translation motions (d_x, d_y) are based on c_s and c_e , the centroids of starting point pair and ending point pair, respectively. Scaling motions (d_s, d_e) are the distance of starting and ending point pairs, respectively. Rotation motion is the θ from the starting point pair to the ending point pair. | 50 |
| 3.17 | (a) Illustration of the substroke relationships from example of Fig. 3.15. (b) The corresponding adjacency matrix. | 51 |
| 3.18 | (a) Substrokes and their relations as depicted in Fig. 3.17(a). (b) The graph representation of the gesture. (c) The attributes associated to the vertices and edges. | 53 |
| 3.19 | Two graph representations of a gesture. (a) The original gesture which is also shown in Fig. 3.14. (b) The graph representation if two strokes are written in a synchronous manner, i.e. (s_{11} synchronizes to s_{21} , s_{12} synchronizes to s_{22}). (c) The graph representation if the stroke s_{21}, s_{22} is written after the stroke s_{11}, s_{12} | 53 |
| 3.20 | (a) Two gestures represented by substrokes and substroke relations. (b) The graph representation of two gestures. The vertices which belong to a same original stroke are grouped into a subgraph V_s^n, U_s^m . (c) The DTW distance matrix is solved by Munkres' Algorithm to find the optimal vertices matching between two graphs. Consequently, the vertices edit operation set Φ_{all}^* can be deduced from the DTW alignment. | 56 |
| 3.21 | The edge matching and edit operation of the two gestures in Fig. 3.20. The edge matching is implied by the vertices edit operation set Φ_{all}^* | 57 |
| 3.22 | The multi-touch gesture templates in MTGSet. | 59 |
| 3.23 | Samples and their variations in MTGSet. | 60 |

| | | |
|-----|---|----|
| 4.1 | Early recognition for a “Heart” gesture. In stage 3, the trajectory is recognized as “Drag” manipulation which gives an instant feedback along with the trajectory. In stage 4, the trajectory is recognized as a “Symbolic-pattern”, i.e. an indirect command. System need to wait until the end of the trajectory to interpret it as a “Heart” symbol. [PM13]. | 66 |
| 4.2 | The common part ambiguity for early recognition. Three gestures have different common parts between each other. | 66 |
| 4.3 | (a) A normalized gesture as a template. (b) (c) In a size free context, due to the input gestures having a variety of the size, a trajectory with a length of l may cover different parts of a same type gesture. | 67 |
| 4.4 | The structure of multi-classifier early recognition system. | 69 |
| 4.5 | (a) Trajectory of an example gesture. p_0 and p_e are the starting and ending point, respectively. p_{k1} is the keypoint where the length of seg_1 (from p_0 to p_{k1}) is l_{Δ} . p_{k2} represents the point at $2l_{\Delta}$. Since the total length is less than $3l_{\Delta}$, this trajectory will offer three segments for training. (b) Classifiers are trained with different segments. | 70 |
| 4.6 | Ambiguous patterns and outlier patterns in multi-class recognition rejection problem. The dotted straight lines represent the pair-wise hyperplanes to separate two classes. The curves are ambiguity rejection boundaries for each class. | 71 |
| 4.7 | Recognition results with respect to the length of input gesture on two datasets. FAR and RR are obtained using the reject option while ER is the traditional mis-classified rate. | 75 |
| 5.1 | The diagram data acquisition procedure on a 80" touch screen. Two users are drawing the diagram together using stylus. | 80 |
| 5.2 | Example of a mind map diagram and the corresponding handwritten diagram without text. | 82 |
| 5.3 | Samples of isolated symbols in diagram. | 83 |
| 5.4 | Variability of the rectangle symbol. | 83 |
| 5.5 | Example of temporal activity of strokes under 2 users condition. S_{ij} indicates the j th stroke from user i | 84 |
| 5.6 | The framework of eager interpretation system. | 85 |
| 5.7 | The decision for some shape should be postponed in case of forming another potential gesture. | 86 |
| 5.8 | The score of unfinished class is manually increased at the ending time of a stroke and decreased with time elapsing. | 87 |

| | | |
|------|---|----|
| 5.9 | An example of 3 strokes in the stroke list. All the 7 possible stroke's combinations are shown in the figure. The round square shows the 5 possible decisions made for the 3 strokes. According to their score, an <i>unfinished</i> gesture and a <i>Diamond</i> gesture are chosen as the final decision. This unfinished stroke will be restored in the stroke list again. | 88 |
| 5.10 | Recognition result for two example diagrams. Gestures in green are correctly recognized. Oranges are mis-grouped gestures. Reds are correctly grouped but mis-recognized gestures. | 90 |
| 5.11 | Stroke stream and their recognition results between steps 50 and 57 from Fig. 5.10 (b). The recognition decisions of each step are bounded in dashed rectangles. | 91 |

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:

Reconnaissance et interprétation des interactions tactiles multipoints

Nom Prénom de l'auteur : CHEN ZHAOXIN

Membres du jury :

- Monsieur ANQUETIL Eric
- Monsieur VIARD-GAUDIN Christian
- Madame EGLIN Véronique
- Monsieur CARDOT Hubert
- Monsieur MOUCHERE Harold
- Madame VINCENT Nicole

Président du jury : *Hubert CARDOT*

Date de la soutenance : 28 Avril 2017

Reproduction de la these soutenue

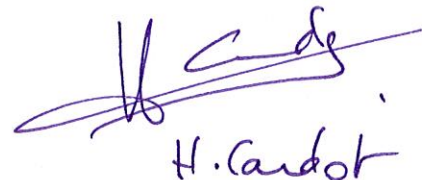
Thèse pouvant être reproduite en l'état

~~Thèse pouvant être reproduite après corrections suggérées~~

Fait à Rennes, le 28 Avril 2017

Signature du président de jury

Le Directeur,



H. Cardot



M'hamed DRISSI



Résumé

La montée en puissance des écrans tactiles offre de nouvelles possibilités d'interactions gestuelles de plus en plus riches. De nos jours, les utilisateurs se contentent souvent de gestes mono-point ou multipoints simples pour exécuter des manipulations telles que la rotation, le déplacement ou la mise à l'échelle d'objets graphiques, la plupart du temps dans un contexte mono-utilisateur. Le travail décrit ici concerne l'utilisation avancée des gestes multipoints, comportant à la fois plus de commandes de raccourci (appelées commandes indirectes) et de commandes de manipulation (appelées commandes directes) dans un contexte d'utilisateurs multiples sur le même écran. Pour cela, nous analysons la forme des trajectoires composant le geste multipoints et les relations temporelles et spatiales entre ces trajectoires afin de caractériser ce geste. Nous proposons une modélisation par graphes et développons un système complet d'analyse et de reconnaissance. Pour résoudre le conflit entre la reconnaissance des gestes de manipulation et ceux de commande (directes versus indirectes), nous proposons une stratégie de reconnaissance précoce pour les gestes multipoints basée sur une option de rejet combinant plusieurs classifieurs pour reconnaître ces gestes au plus tôt. Pour valider nos approches, nous avons construit la base MTGSet composée de 7 938 gestes isolés multipoints de 41 classes différentes et MUMTDB une base de gestes collectés dans un contexte réel d'interaction multi-utilisateurs pour l'édition de diagrammes. Les résultats expérimentaux attestent que nos approches peuvent reconnaître les gestes multipoints dans ces différentes situations

Abstract

Due to the popularization of the touch screen devices, nowadays people are used to conduct the human-computer interactions with touch gestures. However, limited by current studies, users can use only simple multi-touch gestures to execute only simple manipulations such as rotation, translation, scaling, with most of time one user even if adapted devices are now available. The work reported here concerns the expanding usage of multi-touch gestures, that make them available at the same time for more shortcut commands (called indirect commands, as copy, past, ...), more manipulation commands (called direct commands like zoom or rotation) and in the context of multiple users on the same screen. For this purpose, we analyze the shape of the gesture's motion trajectories and the temporal and spatial relations between trajectories in order to characterize a multi-touch gesture. We propose a graph modeling to characterize these motion features and develop a graph based analysis and recognition system. To resolve the conflict between interface manipulation and shortcut command inputs, we study and validate an early recognition strategy for multi-touch gesture. We built a reject option based multi-classifier early recognition system to recognize multi-touch gestures in early stage. To set-up, train and validate our systems, we built MTGSet, a multi-touch gesture dataset formed by 7938 gestures from 41 different classes collected in isolated contexts and MUMTDB a dataset of gestures collected in a real multi-user usage case of diagram drawing. The experimental results prove that our system can well recognize multi-touch gestures in these different situations.