Multi-touch Gestures Recognition Using the Strategy of Graph Embedding

Boussad Ghedamsi

▶ **To cite this version:**

Boussad Ghedamsi. Multi-touch Gestures Recognition Using the Strategy of Graph Embedding. Human-Computer Interaction [cs.HC]. 2013. dumas-00854968

HAL Id: dumas-00854968

https://dumas.ccsd.cnrs.fr/dumas-00854968v1

Submitted on 2 Sep 2013

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.

Master Thesis

Muti-touch Gestures Recognition Using the Strategy of Graph Embedding

By

Boussad Ghedamsi

Submitted to the Institute of Electrical Engineering and Computer Science (ISTIC) in fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the
UNIVERSITY OF RENNES1
[June , 2013]

Thesis supervisors: Eric Anquetil[1] and Harold Mouchère[2].

Title [1] : Professor at INSA Rennes.
Title [2] : Associate Professor at the University of Nantes.

In collaboration between:

Team [1] : INTUIDOC , IRISA.
Team [2]: IVC , IRCCyN.

Abstract

In this master thesis we propose a new system for multi-touch recognition. We consider a multi-touch gesture a gesture which can be performed on a multi-touch device like a tablet using more than one hand at the same time. A multi-touch gesture is rich in information. We can record from such gesture the position of each finger and its movement as well as the synchronization between the different fingers movements involved in same gestures.

To deal with the complexity of these kind of gestures, we propose to model them using graphs. A graph as a general data structure allows us to represent all the relations that may exist between the graph components. We propose to represent a gesture primitive by a set of labeled nodes in the graph and the relations, either temporal or spatial, that may exist between the fingers movements by labeled edges.

Graph matching is required for gestures discrimination. This task is a complex problem, to overcome this difficulty we will explore an approach which is known by the graph embedding. This approach involves the mapping of a graph into a one dimensional vector. The advantage of this strategy is the availability of statistical tools which allow us to easily compare vectors. The results obtained at the end prove that this strategy does suite the purpose of multi-touch recognition problem.

Key words: Multi-touch recognition, sensitive interfaces, graph matching, graph embedding.

Contents

Illustrations

Introduction

To date, multi-touch interfaces play an important role in every day life. They are present everywhere: incorporated in wide range of technologies like smartphones, tablets and tabletops. They can replace traditional input devices like keyboards or mousses when performing lightweight and mobile tasks like surfing the Internet, doing presentations or even recording the signatures of clients upon the delivery of their merchandise. Actually, they are commonly used in gestures recognition. Multi-touch interfaces belong to the family of sensitive interfaces. On this kind of interfaces, we are able to interact either by using stylus or through our fingers. There are inner engines that capture the movements as a signal, interpret it and execute the suitable command.

On conventional systems that are keyboarded and mouse based, the interaction is relatively simple. Because first, there is a standardized manner with which the user should go through to manipulate it and second we are accustomed to it. However, multi-touch is a new technology, and using it is not that simple. In fact, the users don't behave in the same way in order to execute a given action: users actions are synchronized differently.

The implementation of a multi-touch based system is a difficult task, we have to deal with the state transitions of many fingers and their simultaneous movement. Actually, some industrial actors incorporate basic multi-touch gestures inside their devices but we are limited in the number of gestures and the number of fingers used in. Therefore, multi-touch gesture recognition on sensitive interfaces is still a work in progress, and a lot of research work is going on.

Multi-touch gesture recognition on sensitive interfaces has been the subject for an internship last year at the I*ntuiDoc* team[1] [Rahmoun, 2012]. It was their first study of this subject, where they thought deeply the problem of gestures modeling and recognition. Finally, they concluded that it is interesting to explore an original approach which consist in dealing with multi-touch gestures by using the strategy of graph embedding. This strategy involves the mapping of a graph into one dimensional feature vector. This year, our task is to explore in depth the strategy of graph embedding and to develop a first prototype of a multi-touch recognition system. To our knowledge, our work is the first to use the graph embedding strategy for the purpose of multi-touch recognition.

The remainder of this thesis is structured as follows. In the first part, we start by introducing multi-touch interfaces and the kind of gestures that could be performed on these interfaces, after that we present the field of graphs and graph matching which are required in order to deal with multi-touch gestures. This first part is in fact an update of the bibliography report we have submitted during the first semester of this year. In the second part, we will present our new system, and finally we conclude by presenting some experiments.

1 IntuiDoc team *www.irisa.fr/**intuidoc***

1. Multi-touch interfaces

1.1 Introduction

Multi-touch interfaces are input devices that recognize two or more simultaneous touches, allowing one or more users to interact with computer applications through various gestures created by fingers on a surface [Interface, 2013] . The (*Figure [1])* shows a three fingers gesture on a multi-touch interface.

Figure 1: Multi-Touch surface

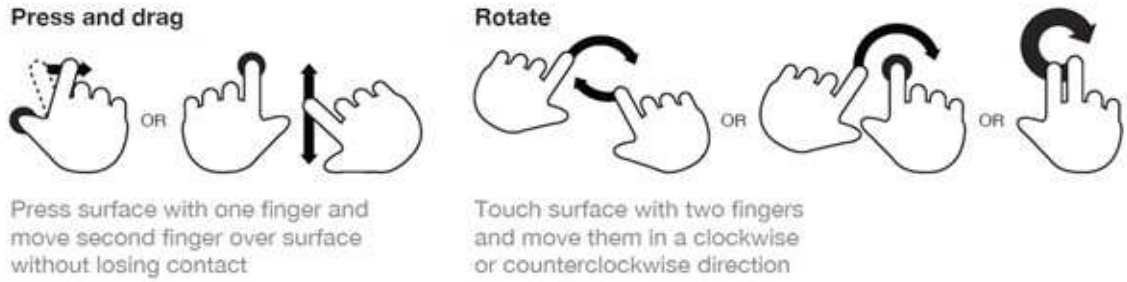


Figure 2: Multi-Touch gestures

Multi-touch gestures recognition will be the subject of our study. We will look in details the nature of those kind of gestures. A multi-touch gesture is the movement of many fingers at the same time so it is characterized by its graphical form, the relative position of each finger and the synchronization between different finger movements. (*Figure [2])* shows some variants of multi-touch gestures.

The resulting signal from the landing and movement of multiple fingers on a screen is called an on line signal. This is opposed to the so called offline signal where the data is available under picture-like format such as text files and PNG files. While the offline signal can only characterize the global form of a gesture after its accomplishment, the online signal can also characterize the dynamic of gestures. The online signal consist of many points following the flow of the fingers. Thus, the flow of each finger is represented by the landing point, the intermediate points and the end point, the different points are connected by edges (see *Figure [3]*). The movement of any finger which begins from the landing moment, finger is down, until the moment when this finger is up is called a stroke.

The on line signal is rich in information, beyond the coordinates of any finger it can also contain information about the time, pressure and the orientation. We refer to mono stroke gesture, the gesture which is formed by one drawing or one stroke, whereas a multi-stroke gesture is composed of multiple drawings, i.e many strokes. The (*Figure [3]*) shows a graphical representation of a single stroke.

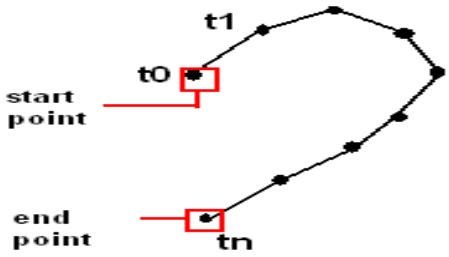


Figure 3: Stroke

1.2 Human surface computing

The (*Figure [4]*) shows different manners of interaction with surface computing. From the early five-ties until our present days each of these interaction methods has served its time. The first of these interaction methodologies is the command language interaction (CLI) where the machine can only understand commands given in some machine programming language. The graphical user interfaces (GUI) come to facilitate the usage of computers by providing graphical standards like folders, files... .But currently, natural user interfaces (NUI) are making a big break through in our lives. These interfaces are associated to Natural Interfaces for being intuitive and natural. In other words, for users these interfaces should operate just like something, or a task some one is familiar with. A multi-touch interface is an example of NUI. The organic interfaces (XUI) are a new methods of interaction with computer applications using physical shapes.

Figure 4: Surface computing

In the following sections, we will be exclusively interested by muti-touch gestures as they are the subject of our study. Exactly, we will see how to model these gestures in order to recognize them. That means, to be able to associate a given gesture to an action the system should be able to decide the same action every time the corresponding gesture is drawn or performed on a multi-

touch surface. This processes is commonly know by *recognition* is the pattern recognition community. In this area, different systems are bench marked with reference to their ability to predict the exact action that corresponds to the gesture being performed. To better recognize a gesture two important conditions should be satisfied: an accurate modeling of the gesture, which should be very discriminant to better differentiate two different gestures, and a good recognition process.

2. Pattern recognition

2.1 Introduction

Pattern recognition describes the decision to take in order to determine to which category, or a class, a given pattern belongs to. This decision is taken according to the class of a previously presented pattern. In daily basis, example of such tasks are: recognition of letters in a book, signature of persons, the face of criminals in a crowd, gestures recognition, and many more.

To achieve this task, a machine is provided with learning data coming from certain problem domain, it then tries to find significant rules, or to learn, in order to solve the given pattern recognition task. We distinguish two main methodologies of learning which are supervised learning and unsupervised learning. In the supervised leaning each training example has an associated class label, this type of learning is used for *classification* purposes. Such kind of learning can be found in biometric person identification, medical diagnosis and many other domains. In the unsupervised leaning their is no label provided for each example. More concretely, here the problem is to partition the given collection of unlabeled patterns into a K meaningful groups, refereed to as *clusters*. This kind of learning is mainly used in tasks for data analysis. In the following we will be interested by supervised learning problems, where the data is provided with class labels. Here, we can distinguish two main approaches: statistical and structural leaning methods.

2.2 Statistical and structural pattern recognition

It is very important to present the patterns to learn in such meaningful way so that the machine can better learn to discriminate between classes. A human, with his recognition capabilities that evolved with him from his childhood can easily distinguish between dogs and birds. His brain has been trained biologically to recognize birds by their wings and dogs by having four legs. But when it comes to an idiot machine this concept has no meaning. So here we feel the need to have the adequate data structures to better characterize the learned concepts. There exist two main approaches to tackle this problem which are statistical approaches and structural approaches.

2.2.1 Statistical pattern recognition approaches

In statistical pattern approaches the patterns are represented by feature vectors of n elements. So a pattern can be seen as a point in n-dimensional feature space. This approach offers numerical advantages, in particular, the mathematical wealth of the operations available in the vector space. Like the sum, the product and distance between vectors, or even using recognition models like

support vector machine (SVM), and neural networks (NN). However the drawback of this approach is in the size of the vectors characterizing the pattern which should have the same length regardless of the complexity of the form they represent and another limitation is in the vector elements which always characterize a predefined set of features.

2.2.2 Structural pattern recognition approaches

In the other hand, structural approaches try to work directly on complex structures. More concretely, representing each pattern by its basic constituent primitives. [Chan et al., 2000] proposed to represent each primitive either by line or a curve or a loop. The decomposition of each symbol into basic primitives is a difficult problem related to the identification of segmentation points. Different approaches exist in order to model these primitives, either by using grammars, sequences alignment, or data structures such as strings, trees or graphs.

- Grammar formalism

Grammar is a formalism which allows to define languages, which are formed by the combination of words to build phrases. By analogy to symbol recognition, words can be seen as the different primitives. The combination of these primitives using grammars yields a valid symbol. So grammar tells us the right rules to form valid phrases thus the right order to assemble the different primitives. The literature provides us with many grammar types, a complete state of the art can be found in [Anquetil et al. 2009].

- Sequence alignment

By considering the different primitives that constitute a pattern as sequences, we can use methods based on distance matching between two set of sequences and then between two patterns [Wu et al., 2006]. The dynamic time warping (DTW) which considers the time dimension when comparing two sequences can be used to calculate this distance as did [Bettens and Todoroff, 2009] for the recognition of dancing gestures or [Santosh, 2011] for recognizing characters. Other approaches consider learning a hidden markov model (HMM) for each class of symbol to recognize can also be used, this approach was used by [Bevilacqua et al., 2007] for building a pedagogical tool for teaching music.

- Strings, Trees and Graphs

Graphs which consist of a set of nodes and edges are the most general data structures. We focus on these structures during this thesis. Beyond describing the global properties of objects, graphs can also describe relationships between the different underlying parts of an object. These relationships can be of spatial nature or temporal nature. Actually graph are used in a wide variety of domains, like, applications in chemoinformatics [Gärtner et al., 2003], web content mining [Senellart, 2012] and graphical symbols recognition [Luqman et al., 2010]. The main drawback of graph compared to feature vectors is in the complexity of algorithms that manipulate them. Comparing vectors can require linear time whereas comparing graphs, or finding graph isomorphism is an NP-complete problem, only exponential algorithms are known today.

2.2.3 Bridging the gap

A promising approach to overcome the lack of algorithmic tools for graph comparison is graph embedding in real vector space. By doing so, a rich repository of tools for vector recognition are available. But, the problem of mapping vectors into a vector space is not a trivial task. The challenge is to define an efficient method to condense the complexity of a graph into on one-dimensional vector. Condensing means to keep only the few relevant information, but the problem is how to determine which information is interesting. The goal of this thesis is to explore the graph embedding methodologies in the context of multi-touch gestures recognition.

3. Graph matching

3.1 Introduction

In this section we are going to present tools which allow us to compare graphs, we can also talk about finding graph isomorphism or graph matching. During the first semester we have conducted a research analysis dealing with graph matching. In the present thesis we are going to present an update of the work done before by introducing new methodologies for graph matching which do better suite our needs. We identified in the present thesis new needs for our multi-touch system which consist in finding graph and sub graph isomorphism. This is why this section of graph matching will focus on the graph and sub graph isomorphism.

3.2 Graph fundamentals

A graph *G(V,E)* consists of a set of vertices *V*, and a set of edges *E*. A graph may be directed or undirected. In a directed graph, *(u,v)* means there is an edge between the two nodes u and v whose direction is from *u* to *v*. In an undirected graph, that means also that there is an edges from *v* to *u*. The (*Figure [5] and Figure[6]*) show an example of two graphs.

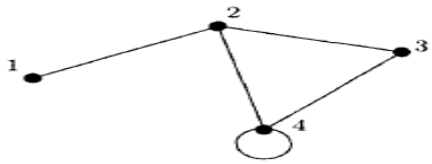


Figure 6: Undirected graph

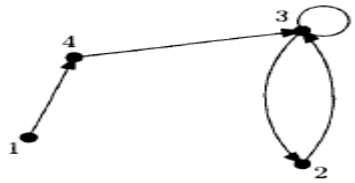


Figure 5: Directed graph

A path is a graph is a sequence of nodes (without repetition) connected with edges. A cycle in a graph is a path that ends with its starting point. The sequence [2,3,4,2] is a cycle in (*Figure [6]*). An attributed graph is a graph whose vertices and /or edges contain labels.

The Graph Isomorphism problem

Two graphs are isomorphic if they are structurally indistinguishable. If the graphs are labeled another condition that should be satisfied to guaranty the isomorphism is that labels must be preserved. More formally, two graph are isomorphic only if their images under bijection is

preserved. The (*Figure [8]* and *Figure[7]*) show two isomorphic graphs.

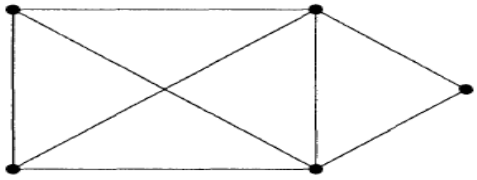


Figure 8: Isomorph graph-1

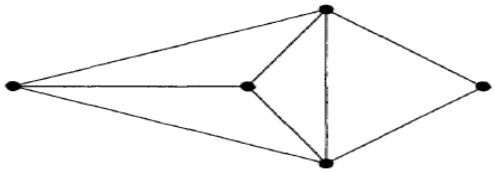


Figure 7: Isomorph graph-2

A generalization of graph isomorphism is sub graph isomorphism. G_A is sub isomorphic to G_B if there exist a sub graph G_C in G_A such as G_C and G_B are isomorphic. The complexity to determine if two graphs are sub isomorphic is an NP-complete problem. The following section is about graph matching where we will introduce methods dealing with the problem of graph comparison.

3.3 Graph matching

Graph matching methods address the problem of graph comparison. We can distinguish three different categories of graph matching methods: exact graph matching based methods, inexact graph matching based methods and finally graph embedding based methods.

3.3.1 Exact graph matching

Exact graph matching algorithms are characterized by the fact that the mapping between the nodes of two graphs must be edge-preserving. That means, if two nodes in the first graph are connected by an edge, then their mapping in the second graph must be connected by an edge as well. This condition must hold in both directions in order to guaranty an exact isomorphism between these two graphs, an additional condition is that the mapping must be bijective.

Most of the algorithms for exact graph matching are based on decision tree search with backtracking. The basic idea, is that a partial match set is iteratively expanded by adding to it a new map (a pair of nodes, one node from each graph). This new map should be valid in order to be accepted in the matching set. To be valid it should be consistent with each match present in the partial match set. To cut-off the search space some algorithms use heuristics to prune as early as possible unfruitful search space. If the algorithm reaches a state where the matching set can not be expanded further and there is still nodes not yet matched it back trace. If all the nodes are matched then this represents a successful isomorphism or sub isomorphism state. Ullman algorithm is a widely known algorithm which uses this backtracking strategy and an heuristic to cut-off the search space, also known as refinement procedure, named forward-checking [Ullman,76]. This algorithm addresses the problems of graph isomorphism and sub graph isomorphism, and despite it age, it is widely used in today's applications.

A more recent algorithm for graph isomorphism and sub graph isomorphism problem is the VF algorithm, thanks to [Cordella et al, 2000]. The name VF stands for Very Fast algorithm. The authors introduced an heuristic based on the analysis of node adjacent to nodes present in the partial map, which is very fast to compute. This leads to an outstanding algorithm for graph isomorphism over large graphs. The authors presented an improved version of this algorithm named VF2 which reduced the memory requirement from O(N2) to O(N) with respect to the number of nodes in the graphs [Conte et al, 2004].

[Messmer, 1995] proposed an interesting algorithm. This algorithm builds a decision tree from the graph database. Using this decision tree, an input graph can be matched against a whole graph database in a time with O(N2) with respect to the input graph size. This algorithm addresses both the problem of graph and sub graph isomorphism. The decision tree is build from the decomposition of the database graph set into small common parts. Their method assumes a great similarity between these graphs.

3.3.2 Inexact graph matching

Due to the noise resulting from the graph extraction in real world applications, exact graph matching are not very much used. In fact, graphs are subject to many deformations do to several causes like noise in the acquisition process. Therefor, the matching procedure should be tolerant to some deformations either in the presence or the absence of nodes, edges or labels. In the inexact matching like algorithms the edge-preservation constraint violation during the mapping between two node candidates is not forbidden. Instead, each violation constraint is assigned a cost. The global minimum matching cost represents the measure of similarity between the two candidate graphs. The more two graphs are similar the more their similarity measure tends to zero.

A tree search with backtracking strategy can also be applied as well to resolve the problem of inexact graph matching. Here, the search is directed by the cost of partial matching obtained so far, and an heuristic to cut-off the search space from estimating the matching cost of the remaining nodes. A first tree search algorithm for inexact graph matching was introduced by [Tsai et al., 79], where neither deletion nor insertion of nodes was possible. This drawback was resolved in a later work [Tsai et al., 83]. In [Wong, 90], the author presented an heuristic for speeding up the search for an optimal mapping.

Spectral methods have been used to address the problem of inexact graph matching. The idea started from the observation that the adjacency matrices of two isomorphic graphs will have the same eigenvalues and eigenvectors. Unfortunately, the converse is not true: the equality between the eigenvalues/eigenvectors of two graphs doesn't imply that the two graphs are isomorphic. An important limitation of spectral method reside in the fact that they are purely structural. In other words, they are not able to exploit the nodes and edges attributes. Example of such methods is provide in the work of [Umeyama, 98].

Kernel methods have also been used to tackle the problem of graph isomorphism. A kernel between two structures can be though as a dot product in an existing high dimensional space. This dot product is mainly used to compute metric between vectors, such metrics can be the Euclidean distance.

Several kernel based methods have been introduced. Convolution kernels infer the similarity between two structures from the similarity of their underlying constituent parts. The ANOVA kernel [Watkins, 99] and the graphlet kernel [Petri et al., 2007] are such examples of convolution kernels. Another class of graph kernel is based on the exploitation of random walks in graphs. Thus, the similarity between two graphs is determined from the similarity of their walks by considering the labels of nodes and edges [Borgwardt et al., 2005]. Other kind of kernels are based on finding identical substructures in two graphs, examples of such common structures are sub graphs, cycles, sub trees, and paths. The reference [Borgwardt, 2007] provide a better insight onto graph kernels.

3.3.3 Graph embedding

Graph embedding is an interesting alternative for resolving the problem of graph isomorphism. This new strategy combines the description power of structural methods and the wealth provide by the statistical tools for pattern recognition. Graph isomorphism is computed by mapping a high dimensional graph to a point in a suitable vector space, before using mathematical computations which are required by different statistical recognition tools. We can distinguish two different approaches that tackle the problem of graph isomorphism using the strategy of graph embedding which are: graph probing methods, and dissimilarity methods.

Graph probing based methods is based on the frequencies of appearance of a specific knowledge-dependent substructures in a graph [Luqman, 2012]. [Sidère et al., 2009] had proposed a graph embedding method which is based on the extraction of substructures of 2 nodes, 3 nodes and so on from the graph. These substructures are built from a non-isomorphic graph network. This network was built by starting from a graph of one node and at each iteration a new edge is added. In case where it was not possible to add an edge then a new node is added. The feature vector representation is then obtained by counting the frequency of each substructure in the graph. Another alternative by the same authors to compute the feature vector representation was to build a projection matrix. This matrix is characterized by its columns which contain the substructures computed before and its rows where each row contains the label of one node or edge. The projection is then to compute the frequency of appearance of a given row label in the correspond column substructure. This will be the strategy that we will adopt in our present work and an example will be given later.

In dissimilarity based methods, the embedding is obtained by computing the dissimilarity of the graph to some prototypes. To achieve this goal, these methods use the graph edit distance. Here instead of extracting a vector from the graph like the graph probing based methods do, these techniques build the vector by measuring the dissimilarity of the graph to n prototypes. As for complexity, it is very important since the graph edit distance is computationally expensive. The first difficulty of these methods is in setting the edit cost, and another problem which also holds in case of graph probing based methods is in choosing the right prototypes. An example of such graph dissimilarity embedding is presented in the work of [Pekalska and Duin, 2005].

So far we have seen the state of the art of gestures modeling, the structural approach representation using graphs seems very interesting like is was concluded by *IntuiDoc* team [Rahmoun,

2012]. We consider representing a gesture by spatial and temporal relational graph and then using the graph embedding strategies for comparing the different gestures. This strategy, as stated before, combines the description power of the structural pattern recognition methods and computational efficiency of the statistical methods. More details about this approach will be discussed in the next sections.

4. Multi-touch recognition: state of the art

The on line recognition of multi-touch gestures is a difficult problem, some work has been done and there are much more options to explore. The majority of the proposed methods use the idea of modeling an on line gesture by the number of fingers landing on the surface, the distance between fingers and the trajectory of each finger. Some works also go further by considering the velocity of fingers.

[Lu et al., 2012], proposed an automatic gesture coder, which from learning some examples given by the user generates code that recognizes multi-touch gestures. Their approach is based on two steps: firstly they learn the state machine from the event sequences given in the training data; secondly, they learn a decision tree to resolve the ambiguity that occurs in the learned states. For testing, 720 gestures were collected from 12 participants. Each time the participant is asked to draw 4 times a given gesture for each 15 target gestures. The device used during the experiment is a Motorola Xoom Multitouch Tablet (10'1" capacitive screen with 1280*800 resolution) running Android 3.0. For the evaluation 12-cross validation protocol between users was adopted. Each time the training was done on the data of one participant and testing on the data of the remaining eleven participants which is a challenging task compared to leave-one out approach. The reported performances of the system are 90% of accuracy when there was two gestures to be recognized and 80% when recognizing a 4 target gestures.

[Damaraju, 2008] proposed a recognition system based on sequence alignment using an HMM(Hidden Markov Model). First the authors developed a vocabulary of 40 different gestures to learn and recognize. For feature extractions from the movements of fingers a video processing system was used. 20 samples of each gesture were collected from each of 10 users. To train an HMM, 20 random training sample for each gesture from the entire set of 10 users are used and the remaining samples were used for testing. The results reported by the authors are: 93.8 %, 91.4 %, 92.1 % and 88.8 % for receptively recognizing gestures involving two, three, four and five fingers.

In [Kristensson, 2012], the authors presented a bi manual gesture interface for a 3D full-body tracking sensors. Their method uses a probabilistic algorithm to predict the users' intended one and two handed gesture before its accomplishment. For tracking purposes, the authors used a Microsoft Xbox 260 Kinect sensor which was connected to windows 7 laptop. To continuously tracking the movement of hands and to determine the start and the ending of a gestures a zoning technique was used. This technique involves the definition of an input zone where the user is considered acting based on the distance of the users' hands from the kinect. To evaluate the system, three data sets were used: the Graffiti data set: 27 one-handed gestures, the $1 data set [Wobbrock, 2007]: 16 gestures one-handed gestures and a set of 16 bimanual-gestures defined by the authors. A set of 18 volunteers were randomly split into two groups of 9 participants to avoid the effect of fatigue. In the first group each participant was asked to draw one sample from each gesture of the Graffiti data set

and five samples of bimanual gestures so finally each participial has drawn 27+16*5 gestures. In the second group each participant was asked to draw five samples of each gesture from the $1 data set i.e. 16*5 gestures were collected from each one. The authors randomly picked the data collected from four participants in each group as training and remaining data for test. And held-out data was kept for final evaluation. The recognition rates were 92.7 % and 96.2 % for recognizing one -handed and two-handed gestures.

5. A new muti-touch recognition system

We introduce in this section the new recognition system that we have developed. This work complements the work of [Rahmoun, 2012]. The outcome of last year work was a methodology for modeling multi-touch gestures. This year, our mission was to put this idea in practice a choose an embedding strategy. We will now detail the major building blocks of this system, the content and the function of each block and how all these components cooperate in order to fill the task of recognition.

*5.*1 The general architecture of the system

In our new multi-touch recognition system, we can distinguish tree major blocks which are respectively: *The Graph Building Block*, *The Graph Embedding Block* and finally the *Recognition Block*. We will now present the function of each one of these blocks.

1) *The Graph Building Block*

This block assumes the task of representing a multi-touch gesture. It tries to model as good as possible the richness of these kind of gestures. To achieve this goal, we have decided to adopt an approach where we represent the gestures by spatial and temporal labeled graphs. Each primitive will be presented by a set of nodes in the graph and the relations between these primitives will be represented by labeled edges. We will give a more details about this process in the graph modeling sub section *(5.2)*.

2) *The Graph Embedding Block*

This block proceeds to the embedding of the graph issued by the *Graph Building Block* into a one dimensional vector. The purpose of this process is to try to keep a simple representation of the graph while avoiding the loose of much of the information contained in the primary structure, i.e the graph. This simple one-dimensional vector representation is required to operate the third block which is the *Graph Recognition Block*. A complete description of this block will be detailed in the embedding strategy sub section *(5.3)*.

3) *The Graph Recognition Block*

As we have seen, the output of the *Graph Embedding Block* was a simple one-dimensional vector which is just what this recognition block needs to operate. Taken the vectors entries, this block can perform the classification task using a well known statistical tools (classifiers) like an SVM or NN.

The general flow of the information inside our system is illustrated in (*Figure [9]*).

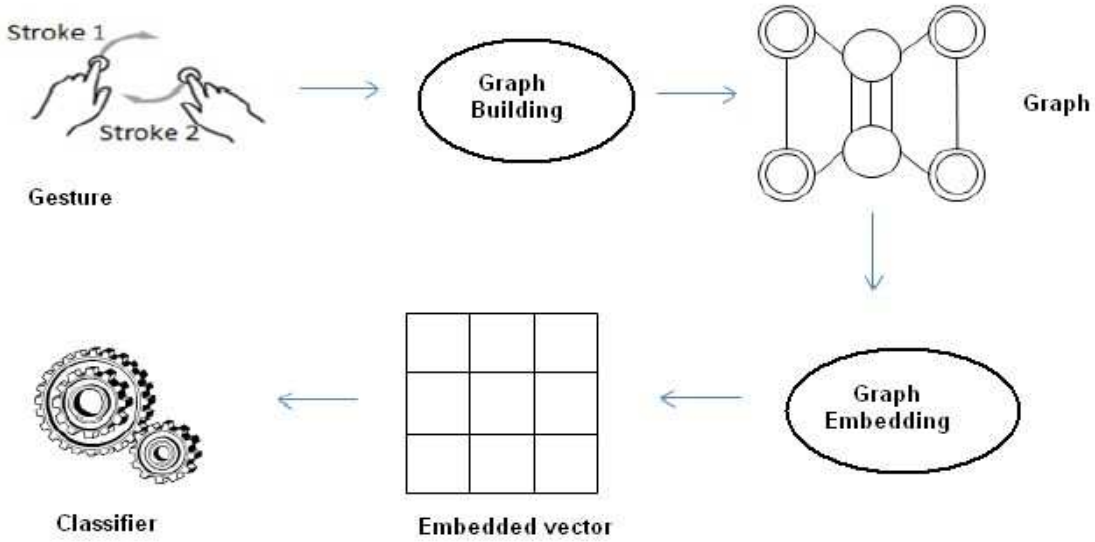


Figure 9: Multi-touch system

5.2 Graph modeling

A multi-touch graph is rich in information, either spatial or temporal information. As we have seen in the introduction section of the multi-touch gestures, each gesture is composed of many primitives. A spatial information tells us the position of each single primitive with reference to the other primitives of the same gesture, whereas a temporal information describes the temporal order between the different primitives and the duration of each one. A third important information is about the shape of each primitive which allows us to differentiate for example between a straight line and a curve. All these three level of information should be taken in consideration when modeling multi-touch gestures to guaranty a complete representation of the information.

We consider representing a gesture by spatial and temporal relational labeled graph. We are going to present now a basic representation which considers a gesture as a simple graph structure, and after that we are going to integrate into this basic model, the spatial information, the temporal information and finally the shape of the strokes.

A basic representation using graphs

Most of the graph representation methods that we can find in the state of the art relies on the idea of representing each primitive with one graph node, and the relations between the primitives with graph edges. In our approach we consider representing each primitive by three different nodes: one node represents the primitive as a global form and the two remaining nodes for representing the extremities (begin and end) of the primitive. This representation allows us to model the spatial and temporal relations between two primitives as global forms but also between their extremities. The (*Figure [10]*) illustrates a simple representation of a multi-touch gesture with a graph.

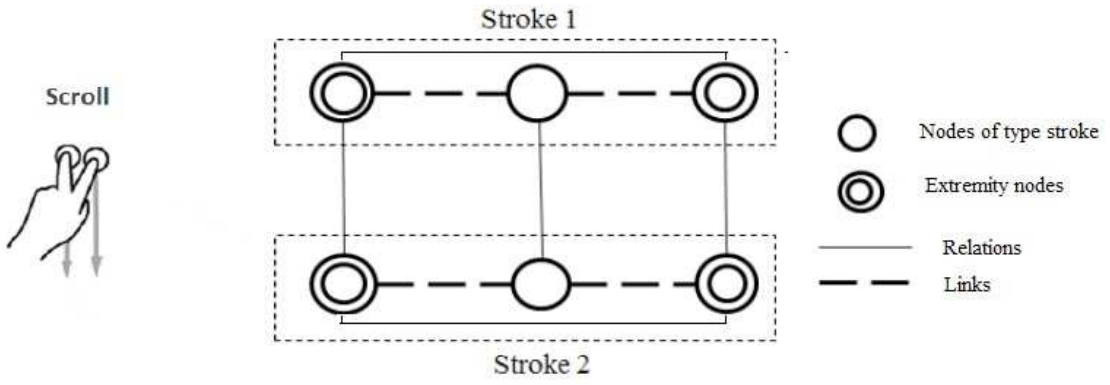


Figure 10: Basic Multi-touch graph

The structure of the graph changes depending on the dynamic of the gesture. Le number of nodes per stoke is fixed to three, whereas the number of edges is not fixed. The edges of type "links" are present in each graph because they link nodes of the same primitive, but those of type "relations" may or may not exist depending on the existence of the relations. Theses relations may be either of temporal or spatial nature.

To complete the preceding representation, which is just a raw modeling of the gesture "scroll", we need to set up the labels for the edges and nodes in order to provide a compete characterization of its graph. For this purpose we should model two types of information which are the temporal information and the spatial information.

Temporal information modeling

Here we consider each primitive as an event. The time which spans from the starting of one primitive until the time when this primitive finishes represent a temporal interval. Allen's relations [Allen,83] is a set of temporal relations which was introduced by James F. Allen. These relations give us a model for a complete characterization of temporal intervals. The (*Figure [11])* shows the set of Allen's relations and the T*able-1* describes the meaning of each relation.

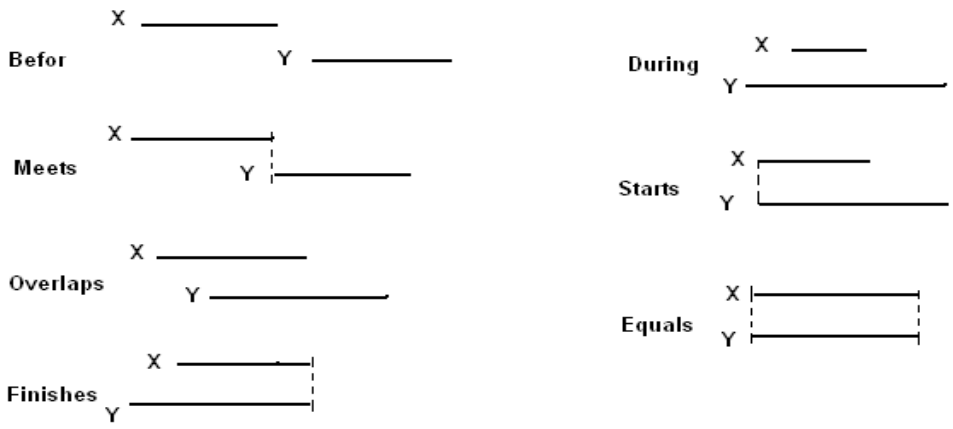


Figure 11: Allen's relations

Relation name	Semantic
Before	If the two events are totally independent from each other
Meets	If the ending of one event coincides with the beginning of the other
Overlaps	The two events overlap
Finishes	The two events end at the same point
During	One of the two events occurs while the second is being performed
Starts	The two events begin at the same point
Equals	The two events begin and end at the same time

Table 1: Semantic of allen relations

Taking the gesture "scroll" as an example, we discover the “overlaps” temporal relation (see *Figure[12])*

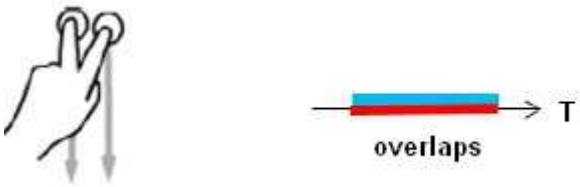


Figure 12: Scroll temporal information modeling

Spatial Information Modeling

Several positioning techniques have been introduced in order to model the disposition of different primitives of the gestures. The directions, angles and distances are such examples that could be used. In our work we use the notion of bounding box. The positioning of two strokes is done using the extremities of their bounding boxes. For each gesture we perform the projection of bounding boxes of its primitives along the x-axis and the y-axis and we use Allen’s relations to position the projections. We then consider two types of relations which are: the X-axis relation and Y-axis relation. The (*Figure [13])* shows the X and Y relations of the gesture “scroll”.

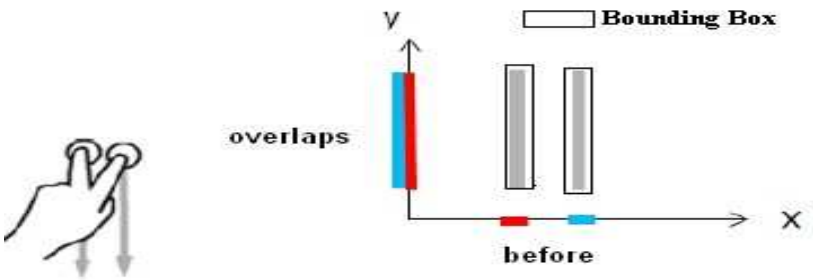


Figure 13: Scroll spatial information modeling

So far we have seen different types of information that can be integrated in a multi-touch graph in order to guaranty a complete characterization of one gesture, and obtain a high level of discrimination between gestures. We now take an example of the gesture "scroll" and try to look at the model graph that results when considering all types of relations that can be discovered.

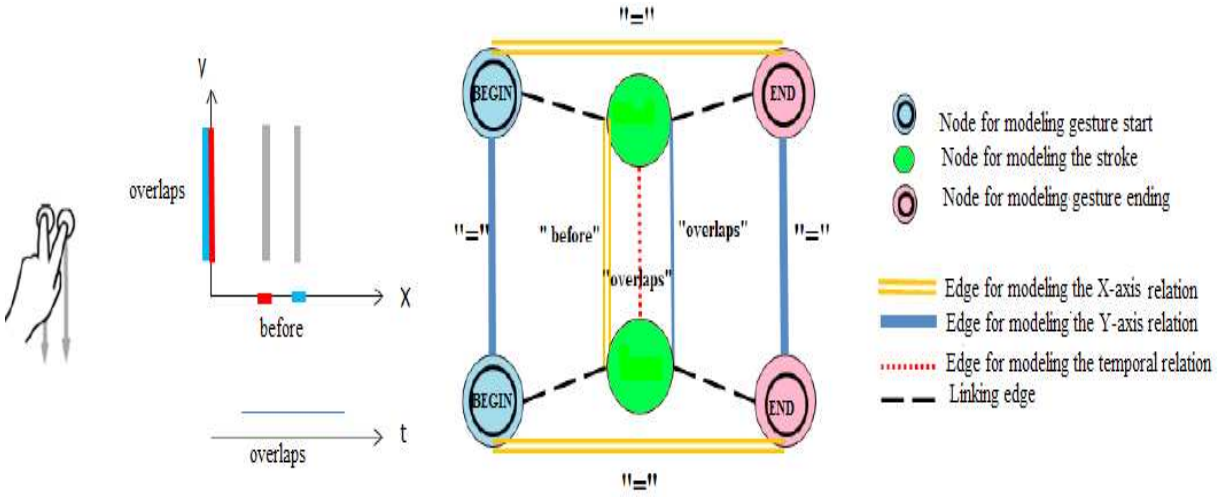


Figure 14: Scroll gestures modeling-1

And here are the details of the relations:

- The gesture scroll contains two primitives so we have six nodes in the graph: respectively two sets of *Begin-Stroke (green circle)-End*

-The two primitives start and end at the same point with ref. to Y-axis: in the graph we have the *Equal* relation between *Begin-Begin* and *End-End.*

-When considering the bounding boxes of each primitive, we have the following relations: *Overlaps* (for Y-axis and Time-axis) and *Before* for X-axis.

Stroke Shape Modeling

We are going now to introduce our new contribution to modeling a multi-touch gesture. This new contribution consist in modeling the shape of the strokes. This step was necessary in order to differentiate gestures which have the same multi-touch graph but different stroke shapes. An example of these kind of gestures is: "Tap" and "Hold" (see *Figure [23]*).

The different primitives of multi-touch gestures can have different shapes. One gesture can be a combination of many segments of straight lines, many cycles, or a mix of the two or more other types of primitives. To be able to recognize the different shapes a learning phase should be done. This phase can be done in a manner totally independent of our system. This learning phase consists in clustering a set of primitives, using the K-means algorithm, which yields a cluster prototypes. These cluster prototypes will be integrated into our system in order to label the nodes strokes of new candidate gestures.

The node stroke labeling of the new gestures is done by computing the distance of each stroke (primitive) of a candidate gesture to this clustering. This can be done by computing the distance to each single cluster prototype and the label of the primitive will be the label of the cluster whose centroid distance to the primitive is the smallest. Examples of distances that can be used in this phase are the euclidean distance [Euclidean] or mahalanobis distance [Mahalanobis].

The gesture “scroll” is composed of two primitives of type vertical lines (*V-LINE*), this information is modeled on the nodes stroke (green nodes) of its graph (see *Figure [15]*)

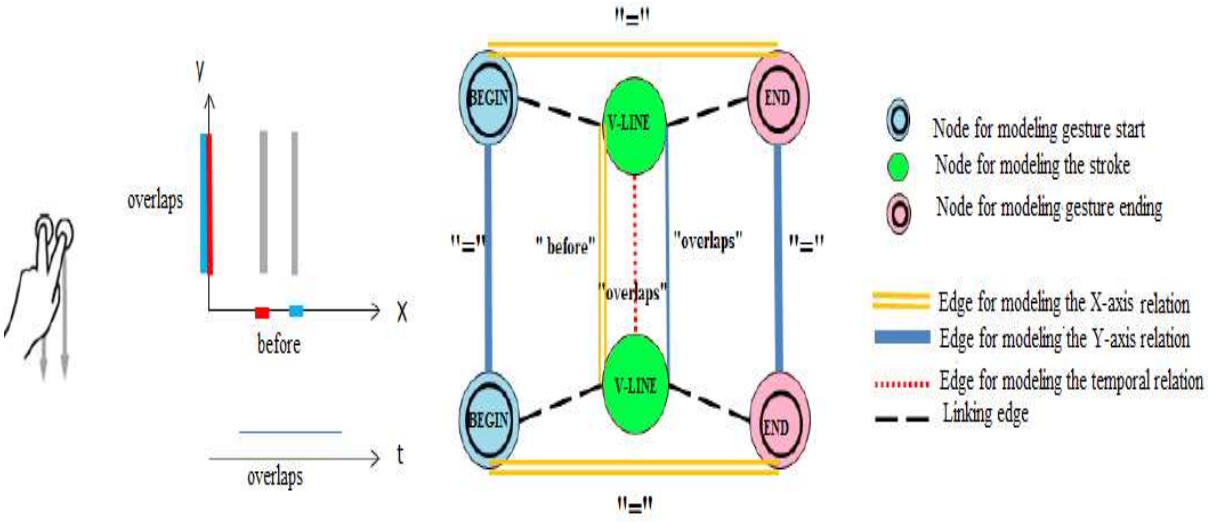


Figure 15: Scroll gesture modeling -2

Once we have a model which does better represent a multi-touch gesture. We have to proceed to embed it into a one dimensional vector for getting a simple representation. Remember that this step is required before using the machine learning tools for recognizing these gestures. We now present the embedding strategy that we use.

5.3 Embedding strategy

To embed a graph into a one-dimensional vector we adopted an approach which was introduced in the work of [Sidere et al., 2009] for recognizing symbols. In this approach we consider the topology information and the labeled information as well during the projection process. The topology tells us the existence of connection between the nodes (primitives), and the labels tell us the type of connection. The basic idea is to search topological structures in the graph and count the frequency of each one. These structures are obtained from a no-isomorphic graph network (see *Figure [16]*). This network represents a set of graphs formed of one node to graphs with N edges, where N is the maximum number of edges. This network is constructed in an iterative manner where in each iteration an edge is added to the previous sub graph if it is possible, otherwise a node is added. The (*Figure [16]*) shows the network of no-isomorphic graphs until the rank of size 4. The dot points show the construction process.

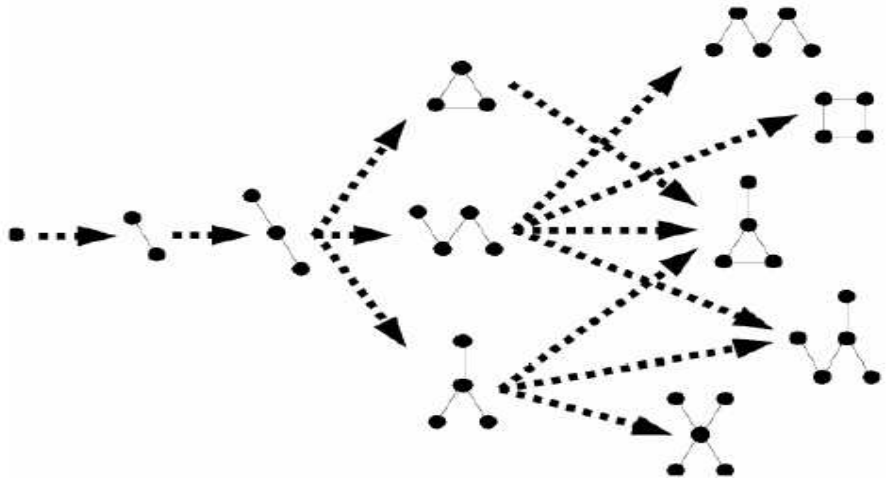


Figure 16: No isomorphic graph network

In a later work the authors proposed an improvement of their approach which has better improved their results by considering the label information as well during the projection process. Instead of just counting the occurrences of sub structures we now count also the labels of nodes and edges found in the mapping of the sub structures inside the graph of the gesture. We then obtain a matrix whose columns are represented by the sub structures/patterns and whose rows represent a frequency of occurrence of each pattern and all the labels of nodes and edges (see *Figure [17]*). Given a pattern P_J and a row label L_I, the value of the projection matrix at $[L_I, P_J]$ is the number of occurrences of this label inside each sub graph P_K found in the graph which is isomorphic to the pattern P_J. In other words if we are interested in counting the value of the projection matrix at the row label Li using the pattern Pj and the graph we want to embed is G, then this value is given by:

$$MatPrj[Li, Pj] = \sum frequency(Li) \in Pk \text{ where } Pk \in G, Pk \text{ isomorph with } Pj$$

Table 2: Embedding formula

Instead of using substructures which are independent of the training data set as did [Sider et al.,2009], in our approach we consider using substructures which are dependent of the data set to ensure a strong connection between our graphs and these substructures. These patterns can either be set empirically or detected automatically. Finally, once the embedding matrix is built, a one dimensional vector is extracted by considering the matrix as one-dimensional vector, this vector represents an input for training the classifiers to recognize gestures.

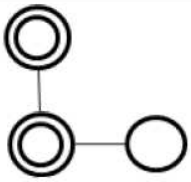
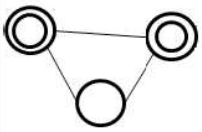
Patterns / Labels				 Pj	
Frequency in the graph					
Label_node 1 Li				MatPrj [Li , Pj]	
.....					
Label_node n					
Label_edge 1	0		0		
.....	0		0		
Label_edge m	0		0		

Figure 17: Embedded vector

5.4 Classification

For the recognition of the multi-touch gestures this step works only on their simple representation, ie the embedded vectors. During the training process we will build a model from the training examples where the projection information and the labeled information are provided. This model will be used later to predict the class of an unknown gesture. Many tool boxes for machine learning can be used in this step and many algorithms, like SVM or K-NN, can be tested. This training process is illustrated in *Figure[18]*.

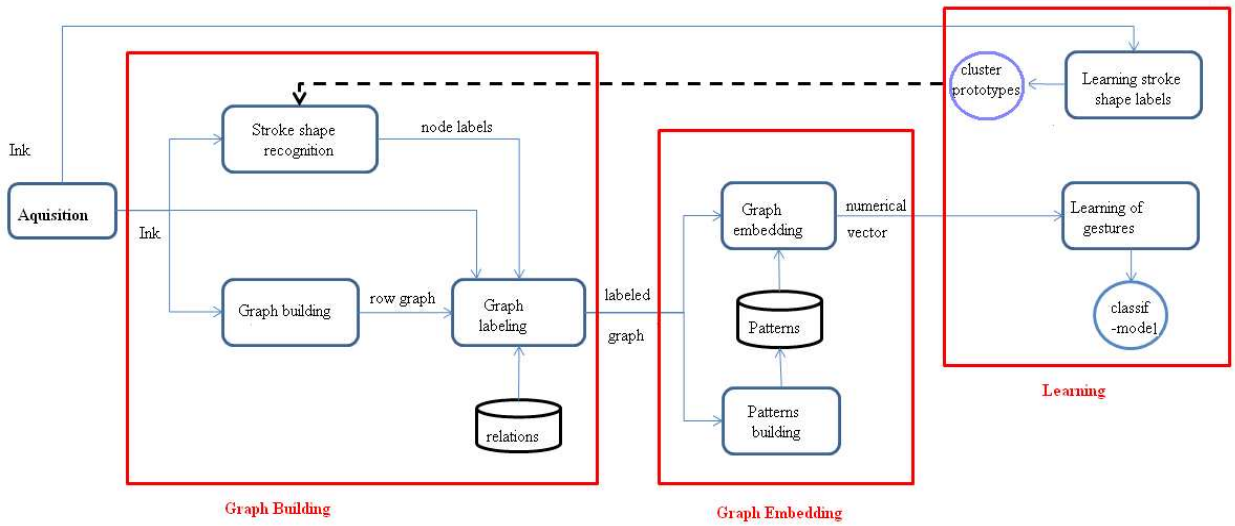


Figure 18: Learning Process

For predicting the name of an unknown gesture, we proceed as in the training process by building the graph of this candidate gesture. And after that, we use the classification model built in the training process for predicting the name of this gestures. The *Figure [19]* illustrates the prediction process.

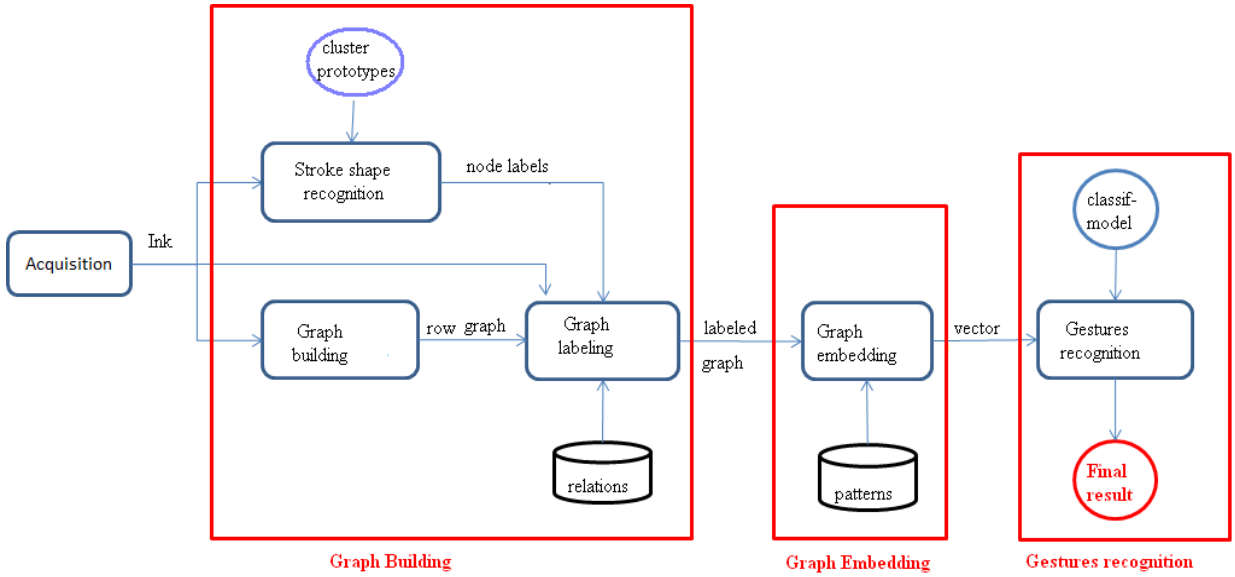


Figure 19: Prediction Process

In the appendix we give the general multi-touch recognition algorithm which describes these learning and prediction processes in a more formal way.

6. Experiments

6.1 Protocol

We have implemented our system using the C++ language. For building the graphs we used LEDA (see appendix) and implemented Ullman Algorithm [Ulman, 76] to find graph and sub graph isomorphism. The training and testing was done using the machine learning tool box RapidMiner (see appendix).

The gestures data set

To test our system, we have selected a set of ten multi-touch gestures (*Figure [23]*). These gestures were selected from the "Open source Gesture Library"[2]. The data was collected at the *IntuitDoc* team using a multi-touch surface (*Figure [20]*) thanks to the participation of five volunteers. Each person was asked to draw ten samples per gesture for ten different gestures, so we have one hundred samples per person. Our data set contains then five hundred gestures.

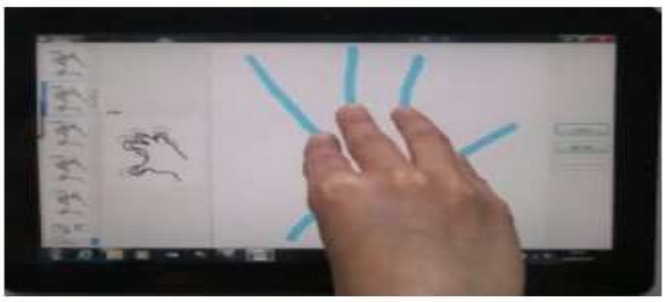


Figure 20: IntuiDoc multi-touch surface

Data base partition

Traditionally, there exist three ways to divide a data base:

- *Multi-writer partition (MtW)* : The same writers participate in the data used for the training and the data used for testing.
- *Omni-Writer partition (OmW)* : The system is trained on the data of some writers and tested using the data of other writers. This represents a real case evaluation of the application. In other words, we evaluate the capability of the system to recognize the gestures of an unknown person.
- *Mono-writer partition(MnW)* : The system is trained and tested using the data of the same person. This testing methodology is the most basic for a data base and gives generally good results. In fact, it is easy for a classifier to recognize an unseen data when it comes from the same source as the training data.

2 http://gestures.com/features.open-source-gestures/ : multi-touch gesture library

Evaluation Protocol

To evaluate our system with mono-writer and multi-writer mode we used a 10-fold cross validation because of the lack of data in this mode. In the mono writer mode, we give the recognition rate by computing the mean recognition rate of each single writer. And for the omni-writer mode we used a 5-cross validation protocol, each time the training was done on the data of four writers and testing on the data of the remaining writer. We repeated this procedure five times and we computed the mean recognition rate. As for classifiers, we used the reference classifier SVM (with an RBF kernel) with a grid-search strategy to determine the parameters which gave us the best results.

6.2 Results

We have done two kinds of experiments. In the first one, for each gesture we built a multi-touch graph without labeling the nodes of type stroke. The results we obtained at the end of the first experiment were not so good, this is why we have done a second type of experiments where this time we integrate the labels of the nodes stroke. We will now present the two experiments in details.

First Experiment

As we have said in the previous paragraph, in this experiment we built a multi-touch graph without labeling the nodes stroke. An example of a generated graph in this first experiment is the (*Figure [21]*) which represents the gesture "Flick". The embedding was done using tree different patterns which we selected manually. To avoid resulting vectors with zero entries, these patterns were very frequents inside each gesture of the data base.

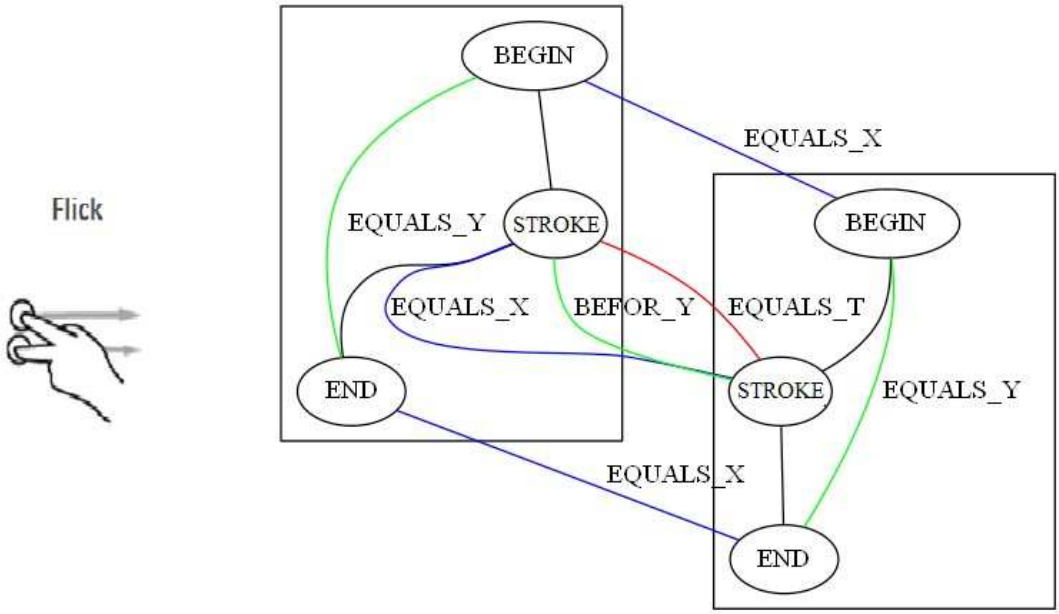


Figure 21: Graph of the gesture Flick without stroke labels

After doing many tests by choosing different patterns during the embedding and optimizing the parameters of the classifier with a grid-search strategy we obtained the results in (Table-3).

	System	Protocol	Accuracy	Std
Graph with no labels on the node stroke	SVM	MnW CV	90.80 %	(4.58 %)
		MtW CV	91.80 %	
		OmW T+V	91.00 %	(4.43 %)

Table 3: First experiments results

T+V : Training / validation CV : Cross-Validation

The Multi-writer mode gave us better results as in this mode there are much data to train upon it. After that comes the omni-writer mode and finally the mono-writer mode with the worst recognition rate.

These results are week but encouraging. In fact, we have observed some confusions between gestures that tend to have the same multi-touch graph but are performed differently on the multi-touch surface. Examples of these kind of gestures are (“hold” and ”tap”) and (“ring” and “scroll”). The resulting contact of “hold” is much thicker than “tap” and “ring” contains two arcs while “scroll” contains two vertical lines. This is what motivate us to integrate the shape information as explained in (section 5.2) inside the multi-touch graph during the second experiment.

Second Experiment

After the integration of the shape information inside a multi-touch gesture graph we obtain a graph whose stroke nodes are labeled with some label which represents the family (cluster) of this stroke. The affectation of a label to some stroke is done in two steps. In the first step, we learn the stroke labels by clustering some primitives from the training set using the K-means algorithm. The optimal parameters for our clustering were k=8 in combination with mahalanobis distance. The parameter k is in fact dependent on the data set and the goal is to obtain a good clustering, where all the primitives in the same cluster have the same shape and two primitives from different clusters are different. The second step is for predicting the labels for the strokes of a new candidate gesture. For each primitive of this new gesture we compute its distance to each cluster using the same distance (mahalanobis) and the cluster label whose centroid distance to this primitive is the smallest is affected to this primitive label. The (*Figure [22]*) represents the new graph of the gesture “Flick” after labeling the nodes stroke. C4 is the the cluster witch represents the family of the primitives of type horizontal line. This interpretation of C4 was deduced after observing that the primitives of type horizontal line were gathered in the cluster C4.

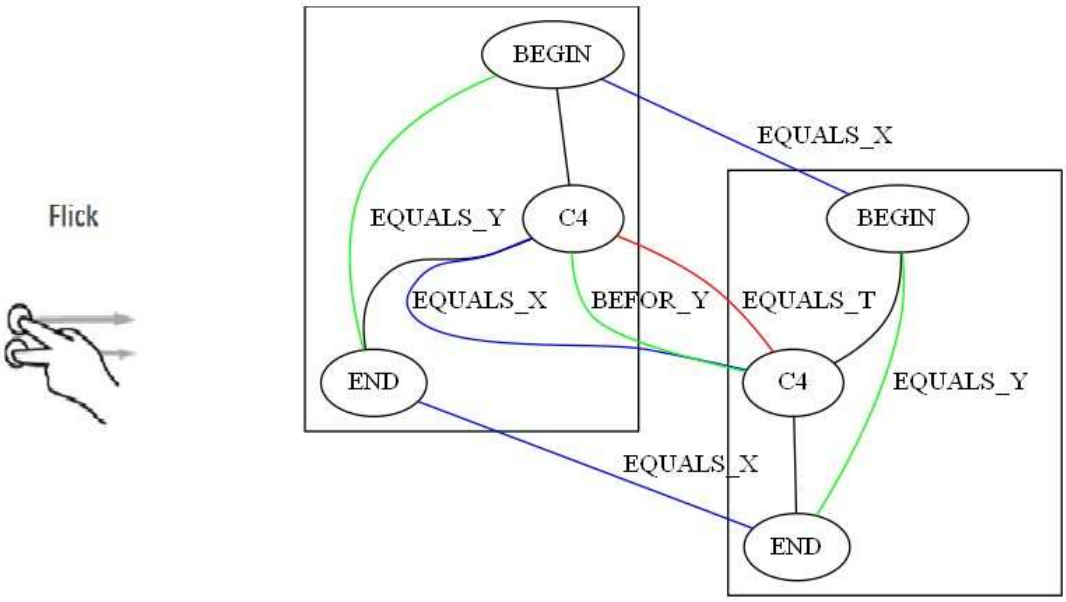


Figure 22: Graph of the gesture Flick with stroke labels

The embeddings of each multi-touch gesture during this second experiment is done using the same patterns used for the first experiment, and the results we have obtained are summarized in (Table-4).

	System	Protocol	Accuracy	Std
Graphs with labels on the node stroke	SVM	MnW CV	94.20 %	(3.76 %)
		MtW CV	95.60 %	
		OmW T+V	94.60 %	(3.44 %)

Table 4: Second experiment results

Discussion of the results

The results obtained in this second experiment as we can see from (Table-4) are much better then those of the first experiment. We reduced the error by approximately 50 % in each mode. This improvement in the recognition rate is in fact the result of our refinement of the multi-touch graph of each gesture by considering the shape of each primitive. This information allowed us the better discriminate some gestures which can have the same graph without considering the primitives shape information. Example of these gestures are (“tap” / “hold”) and (“scroll”/”ring”).

These experiments as we said in the beginning of this section were done using the data of five volunteers. The recognition rates for each person in the mono-writer mode are not the same. In fact, we recorder a recognition rate of 97 % in case of two writers, and for the others a little bellow the average recognition rate which was 94.20 % (from Table-4). From these results, we can deduce that until now we can not state that our system does perform well but we need much more data in order to evaluate its performance.

However, even if we have noticed a very good recognition rates for some writers, other writers results were not very good. This can be explained by the fact the graph construction is not a straight forward task as we have seen during this thesis in the graph modeling sub section. In fact, there are many experimental factors which can influence our results. These factors are the time and space margin. The time margin is measured is milliseconds and tells us the time interval within which two events are considered as occurring in the same time, and space margin is measured in pixel which tells us the space interval within which two points are considered as having the same position in space. These two margins can influence very heavily the type of Allen's relations that can be discovered between two primitives, and the resulting graph will be slightly different.

General conclusion

In this work, we have been interested by the subject of multi-touch gestures recognition under sensitive interfaces. Those interfaces gained an important place in today's applications. They allow a human-machine interaction using basic gestures with limited number of fingers. Examples of these gestures are: scrolling down in a web document and switching between two pages of a book by sliding our finger in the left-right/right-left direction.

Our goal was to develop a multi-touch gestures recognition system which is capable of recognizing any multi-touch gesture. To achieve this goal, we proceeded by steps. In the first step, we proposed to model any multi-touch gesture by temporal a spatial relational labeled graph. Our choice of graphs because they allow to model all the richness of the multi-touch gestures, but also because graphs are a well studied data structures which will facilitate their manipulation. In the second step, we faced the problem of the gestures recognition. The task of gestures discrimination is then reduced to graphs discrimination which lends us to graph matching.

Graph matching is a hot topic in the field of pattern recognition. Several strategies for resolving the graph matching problem have been reported in the state of the art with applications in many real case problems. We have chosen an approach called graph embedding which consists in mapping each graph into a one-dimensional vector. Comparing graphs is then reduced to comparing simple vectors which is a very straight forward task using statistical classifiers. But the big challenge is in the embedding of graphs.

For embedding graphs into one-dimensional vectors, we adopted an approach which was reported in [Sidère et al.,09]. The idea consists in counting the occurrences of sub structures, also called patterns, in the graph to be embedded. The number and the quality of these patterns are very important variables in the embedding equation. The authors of this strategy used patterns which are independent from their data base. In our approach we propose to act differently by choosing patterns which are frequents in the gestures data set.

The results we obtained confirm that we are in the right way toward having a robust multi-touch recognition system. The recognition rates for some writers were very good. However, there are some experimental parameters which need to be tuned in order to increase our system reliability. Examples of these parameters are the problem of temporal and the spatial margin. So it will be interesting to do a battery of tests with different parameters and observe the performance of our system, eventually with increasing the data set with new gestures. Finally we need also to set up protocols in order to compare our system with systems dealing with multi-points gestures based on classical approaches.

Appendix

Gestures data set

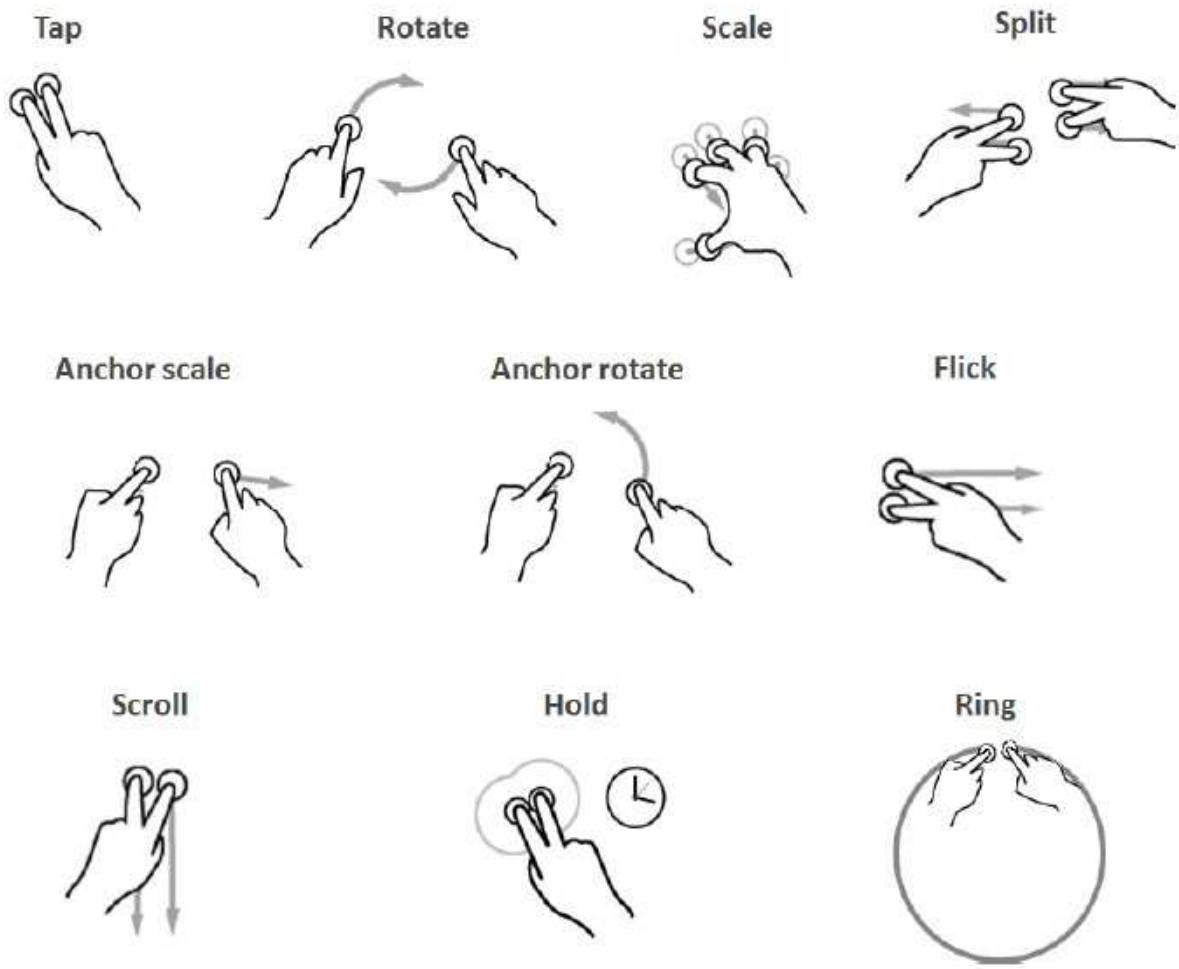


Figure 23: Gestures data set

General multi-touch recognition algorithm

The multi-touch gestures recognition is done essentially in three steps:

Step 1: Stroke Shape Learning /* To Learn the labels of the nodes of type stroke */

```
Input: a set of training primitives (strokes) : Stroke_Set
Output: cluster prototypes: Clustering_Protos

vector_strokeSet ← empty;
for each s: Stroke_Set do
{
   vector ← s.getFeatures(); /* Compute a suitable set of features*/
  vector_strokeSet.add(vector);
}
/* Build the cluster prototypes */
Clustering_Protos ← CLUSTERING(NbClsusters,vector_strokeSet );
```

Step 2: Training /* Build the classification model from the training set */

```
Input: a set of gestures to train on: Train_set
output: classification model : Classification_Model
Graph-V-Set ← empty;
  For   each gesture: Train_set do
 {
    /* Building the graph using Allen relations, and cluster prototypes for labeling stroke nodes*/
      graph_i ← buildGraph(gesture, Allen_rel, Clustering_Protos)
      /* Embed the graph into a one dimensional vector using a set of patterns/ sub structures*/
      graph-i.Embedding ← Embeddings (Pattern_set, graph_i)+gesture_label
      Graph-V-Set .add(graph-i.Embedding);
 }
 Classification_Model ← CLASSIFICATION.TRAIN(Graph-V-Set);
```

Step 3: Recognition /* Prediction of the class of an unknown gesture*/

```
Input: a set of unknown class gestures: Prediction_set, and a training model:
Classification_Model
  output: each input gesture will be assigned a class (the predicted name)

 For   each gesture: Prediction_set do
 {
    /* Building the graph using Allen relations, and cluster prototypes for labeling stroke nodes*/
   graph_i ← buildGraph(gesture, Allen_rel, Clustering_Protos)
   /* Embed the graph into a one dimensional vector using a set of patterns/ sub structures*/
   graph_i.Embedding ← Embeddings (Pattern_set, graph_i);
   graph_i.Name←CLASSIFICATION.PredictClass(Classification_Model , graph_i.Embedding);
   gesture.setName( graph_i.Name);
}
```

Developpement Tools

Os: Windows 7 Entreprise / Ubuntu 12.04(only for testing some components)
IDE: Visual studio 2010 (Windows)/ CodeBlock(Linux)

C++/STL: The process chain was completely implemented in the C++ programing language because of its robustfulness.
http://www.cplusplus.com/

Leda

For graph construction and manipulation we have used the Leda Graph library free edition. This version contains many useful data structures for graph processing like doing breadth first search and computing short paths in a graph.... but unfortunately lacks some important algorithms like graph an sub graph isomorphisms which was very useful in our work. Te achieve this goal we have implemented the Ullman algorithm in C++ [Ullman, 76].
http://www.algorithmic-solutions.com/leda

GraphVis (Graph Visualization software)

Is an open source library for graph visualization initiated by the At&T labs research. It allows an easy representation of graphs : adding colors, fonts, shapes ..
http://www.graphviz.org

Rapidminer

Is an open source environnement for machine learning, data mining , text analysis. According to *Kdnuggets* which is a data mining newspaper, this software ranked first as a data mining/analytics tools used for real projects in 2010.
http://www.rapid-i.com/.
Beside, it is also a very extensible software:
http://geoimagermp.gforge.inria.fr/.

Inkml(Ink Markup Language)

The data of the gestures was available under the Inkml format which is a data format for representing ink entered with an electronic pen (stylus) or finges. This data format has a structure of an XML file.
www.w3.org/TR/InkML/

Xerces

Is an apache's library for parsing, validating and manipulating XML documents. We have used this library to parse the inkml files for extracting the information about the strokes of the gestures.
http://xerces.apache.org/

OpenCv (Open source computer vision library)

Is a computer vision and a machine learning library, we used this library for performing the K-means clustering using mahalanobis distance.
http://opencv.org

Bibliography

[Sidère et al., 2009] Nicolas Sidère, Pierre Héroux, Jean-Yves Ramel "Vector Representation of Graphs : Application to the Classification of Symbols and Letters" 10th International Conference on Document Analysis and Recognition, ICDAR '09. P 681-685, (2009)

[Luqman, 2012] Muhammed Muzzamil Luqman: "Fuzzy Multilevel Graph Embeding for Recognition , Indexing and Retrieval of Graphic Document Images". Phd dissertation at Université de Francois Rabelais Tours, France. (2012)

[Rahmoun, 2012] Somia R., " Composition manuscrite de gestes graphiques sur des interfaces tactiles multipoints", Master thesis at Université de Rennes1, France. (2012)

[Ghedamsi, 2013] Boussad G., "Multi-touch gestures recognition using the strategy of graph embedding", Master bibliography. (2013)

[Inokuchi et al., 2000] A. Inokuchi , T..washio and H. Motoda : "An apriori-based Algorithm for mining Frequent Structures from Graph Data". Proc PKDD 2000, sept 13-16, 2000 , Lyon , France.

[Quershi, 2008] Quershi J. ," Reconnaissance des formes et des symboles graphiques complexe dans les images des documents", Phd dissertation, Université de Francois Rabelais Tours, France. (2008)

[Interface, 2013] www.educause.edu/eli (on line)

[Gestures, 2013] : www.lukew.com/touch (on line)

[Lu et al., 2012] Hao Lu. And Yang Li : Gesture Coder : "A Tool for Programming Multi-Touch Gesture by Demonstration", CHI'12, Austin, Texas, USA, (2012).

[Wu et al., 2006] Mike Wu, Chia Shen, Kathy Ryall, Clifton Forlines, and Ravin Balakrishnan "Gesture Registration, Relaxation , and Reuse for Multi-point Direct-Touch Surfaces", Horizontal Interactive Human-computer systems. First IEEE International Workshop. P. 8, (2006).

[Bettens and Todoroff, 2009] Frédéric Bettens, Todor Todoroff. "Real-Time DTW-based gesture recognition external object for MAX/MSP and Puredata". Proceedings of the SMC 2009- 6th sound and music computing conference, 23-25 july 2009, Porto-Portugal.

[Bevilacqua et al., 2007] Frederic Bevilacqua, Fabrice Guédy, Nobert Schnell, Emmanuel Fléty, Nicolas Leroy : "Wireless sensors interface and gesture-flower for music pedagogy". Proceedings Conference on New Interfaces for musical Expression (NIME07), New York, USA, (2007).

[Chan et al., 2000] Kam-Fai Chan and Dit-Yan Yeung, "An efficient syntactic approach to structural analysis of on-line handwritten mathematical expressions," *Pattern recognition*, vol.

33, p.375-384, (2000).

[Macé, 2007] Sébastien Macé. "Composition manuscrite interactive et interprétation à la volée de documents structurés en-ligne". Phd disseration at *l'Institut Nationale des Sciences Appliqués de Rennes,France, (2007).*

[Xu et al., 2004] Xiaogang Xu, Zhengxing Sun, Binbin Peng, Wiangyu Jin, and Wenyin Liu.s., "An online composite graphics recognition approach based on matching of spatial relation graph," *International Journal on Document Analysis and Recognition*, vol. 7, p 44–55, (2004).

[Damaraju, 2008] Sashikanth Damaraju, Andruid Kerne "Multitouch Gesture Learning and Recognition System", *IEEE Workshop on Tabletops and Interactive Surfaces, p 102-104 , (2008).*

[Anquetil et al.,2009] Sébastien Macé, Eric Anquetil, "Eager Interpretation of On-Line Hand-Drawn Structured Documents: The DALI Methodology", Pattern Recognition, vol 42, P 3202-3214, (2009).

[Messmer, 1995] Messmer, B. "Efficient Graph Matching Algorithms". PhD dissertation, University of Bern, Switzerland, (1995).

[Agrawal et al., 1994] Agrawal, R. and Srikant, R. 1994. "Fast algorithms for mining association rules". In Proc. of the 20th VLDB Conference, pp.487-499, (1994).

[Bunke, 1997] Bunke, H. 1997. "On a Relation between Graph Edit Distance and Maximum Common Subgraph", *Pattern Recognition Letters*. pp. 689 – 694, (1997).

[Umeyama, 1988] Umeyama, S. "An Eigen Decomposition Approach to Weighted Graph Matching Problems". *IEEE PAMI*, 10, pp. 695 – 703, (1998).

[Borgwardt, 2007] Karsten Michael Borgwardt, "Graph Kernels", Phd dissertation , Ludwig-Maximilians-Universität , Munich, Germany, (2007).

[Gärtner et al., 2003] Gärtner, T., Flach, P., and Wrobel, S . "On graph kernels: Hardness results and efficient alternatives". In Schölkopf, B. andWarmuth, M. K., editors, Proc. Annual Conf. Computational Learning Theory, pages 129–143. Springer, (2003).

[Kashima et al., 2003] Kashima, H., Tsuda, K., and Inokuchi, A.. "Marginalized kernels between labeled graphs". In Proc. Intl. Conf. Machine Learning, pages 321–328, San Francisco, CA. Morgan Kaufmann, (2003).

[Mahé et al., 2004] Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., and Vert, J.-P. "Extensions of marginalized graph kernels". In Proceedings of the Twenty-First International Conference on Machine Learning, pages 552–559, (2004).

[Horvath et al., 2004] Horvath, T., Gärtner, T., and Wrobel, S. "Cyclic pattern kernels for predictive graph mining". In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD), pages 158–167, (2004).

[Pekalska and Duin, 2005] Pekalska, E. and Duin, R. P.W. "The Dissimilarity Representation

for Pattern Recognition : Foundations And Applications." World Scientific Publishing, (2005).

[Tsai et al., 79] W.H. Tsai and K.S. Fu. Error-correcting isomorphism of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man, and Cybernetics (Part B)*, 9(12):757–768, (1979).

[Tsai et al., 83] W.H. Tsai and K.S. Fu. Subgraph error-correcting isomorphisms for syntactic pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics (Part B)*, 13:48–61, (1983).

[Wong , 90] E.K. Wong. Three-dimensional object recognition by attributed graphs. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition: Theory and Applications*, pages 381–414. World Scientific, (1990).

[Borgwardt et al., 2005] K. Borgwardt, C. Ong, S. Sch¨onauer, S. Vishwanathan, A. Smola, and H.-P. Kriegel. "Protein function prediction via graph kernels". *Bioinformatics*, (2005).

[Watkins, 99] C. Watkins. Kernels from matching operations. Technical Report CSD-TR-98-07, Royal Holloway College, (1999).

[Petri et al, 2007] K. Borgwardt, T. Petri, H.-P. Kriegel, and S. Vishwanathan. An efficient sampling scheme for comparison of large graphs. In P. Frasconi, K. Kersting, and K. Tsuda, editors, *Proc. 5th. Int. Workshop on Mining and Learning with Graphs*, (2007).

[Ullman, 76] J. R. Ullman, "An algorithm for subgraph isomorphism", J. Assoc. Comput. Mach. Vol. 23,n°1, p. 31-42 (1976).

[Cordilla et al, 2000] L. P. Cordella, P. Foggia, C. Sansone and M. Vento, "Fast graph matching for detecting CAD image components", in Proc. 15th Int. Conf. Pattern Recognition, pp. 1034-1037, (2000).

[Conte et al, 2004] D. Conte, P. Foggia , C.Sansone and M.Vento: Thirty years of graph matching in pattern recognition . International Journal of pattern reecognition an artificial intelligence Vol. , 18, No. 3, 265-298, (2004).

[Zager, 2005] Laura Zager , Graph similarity and Matching , Master thesis at MIT, June, (2005).

[Wobbrock, 2007] Wobbrock, J. O., Wilson, A. D., and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In Proc. UIST 2007 , ACM Press, 159–168, (2007).

[Kristensson et al ., 2012] Per Ola Kristensson, Thomas F.W. Nicholson, and Aaron Quigley: “Continuous Recognition of One-Handed and Two-handed Gestures using Full-body Motion Tracking Sensors”, IUI, Lisbon, Portugal, (2012).

[Santosh, 2011] Santosh “Character recognition based on DTW-Radon” , ICDAR p. 264-268, (2011).

[Luqman et al., 2010] Muhammed Muzzamil Luqman, Thierry Brouard and Jean-Yves Ramel "Graphical symbol recognition using Graph Based Signature and Bayesian Network Classifier", (2010).

[kmeans] J.B.MacQueen "some methods for classification and analysis of multivariate observations, Proceedings of the 5-th Berkely symposium on mathematical statistics and probability ", Berkely, University of California Press, 1:281-297 (1967).

[Allen,83] James F.Allen "Maintaining Knowledge about temporal intervals - CACS", University of Louisiana, (1983).

[Mahalanobis] Prasanta Chandra "On the generalized distance in statistics" Proceedings of the National institute of sciences of india 2(1):49-55, (1936).