*Key words:*
*mouse gestures*
*recognition*
*k-Nearest Neighbours*

Paweł HOFMAN[1]
Maciej PIASECKI[1]

# Efficient Recognition of Mouse-based Gestures

The notion of a mouse gesture is defined in the paper as a simplified form of the notion used in literature. For such gestures, an algorithm of efficient and accurate recognition is proposed. Two different classifiers are tested: one based on neural network and the second one based on the k-Nearest Neighbours algorithm. The achieved high accuracy can be ascribed to the proposed algorithm of preprocessing the rough sequences of points into the normalised input sequences for a classifier.

## 1. Introduction and Assumptions

The limits of the human-computer interaction based on keyboard, and mouse modalities are clearly visible nowadays. Most of the proposals going beyond this dominance need introduction of new computer peripherals. However there are still possibilities of new modalities in the styles of interaction based on standard channels, e.g. a kind of gestures performed with the mouse. In general, *a gesture* is a sequence of interactions with the application, which represents one of the specified symbols Tyson et. al. [6]. Such gesture does not need to be a continues line on the screen or even consists of interactions of one modality. This definition

[1]Institute of Applied Informatics, Wrocław University of Technology,
ul. Wybrzeże Wyspiańskiego 27, Wrocław, Poland.

originates from the pen-based interfaces and is more convenient for them, than for mouse-based ones. It is hard to distinguish in the flow of mouse movements the beginning of an unconditioned gesture.

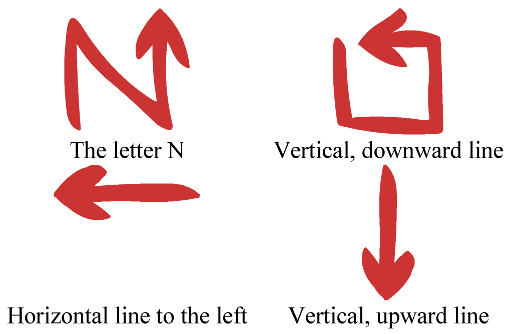
For the mouse-based interaction, we decided to simplify the definition, while still keeping the notion usable: *a mouse gesture* is a continues, directed sequence of the mouse cursor movements with the clearly distinguished start and end, e.g. by the use of the predefined combination of mouse or keyboard keys. In our work, gestures are marked by pressing the right button. The usability of the assumed notion of gesture was assessed during experiments described in Hofman [3], performed according to the standards, e.g. Newman and Lamming [4].

According to the definition, our goal is to recognise and classify a directed sequence of the cursor movements done with the right button pressed as belonging to one of the established classes. The classes are not precisely specified, as the instances can vary in in their shapes and sizes. We assume that the classes are always given by sets of examples expressing user's intentions. We also distinguish the directions of the segments of gestures.

For high usability of gestures-based interface, three basic features must be preserved: *accuracy*, *efficiency* (of recognition), and *adaptability* to the possibilities and needs of the individual user. Accuracy is understood as the percentage of properly recognised gestures in relation to the intention of the user performing them. Efficiency should be enough for the use on an average computer. Adaptability means easy registration of the own classes of gestures of the given user. The quality of the solution should be tested during usability assessment.

## 2. Gestures Classes

Usable gestures can not be complicated drawings, as they must be performed on the limited area of the screen and relatively easy to be repeated with a significant similarity. During several pre-experiments, see Hofman [3], we observed that users propose relatively simple shapes as possible gestures, e.g. the examples presented on the Fig. 1 (the arrows show directions and do not belong to the symbols). Finally, we distinguished 22 different classes as the basis for the further experiments. The chosen set includes: straight lines of different directions, square, circle, simple spiral, a some letters possible to be drawn continuously, i.e. N, M, X, Y, Z, V, W.



Rysunek 1: Examples of classes of gestures.

## 3. General Scheme of Processing

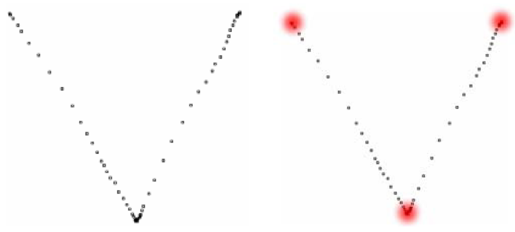
The possible beginning of a gesture is signalled by pressing of the right mouse button. From that moment, until the release of the button, each change in the location of the cursor is registered and added to a vector of points — the rough data. We do not store the time span of the sequence, contrary to Tyson et. al. [6] or Yang & Xu [7]. During the experiments the users intentionally performing gestures were practically drawing them at once. The order of points in vectors defines directions.

As each vector of rough data has practically different length, vectors must be somehow transformed to the assumed length to facilitate classification (most classifiers work on sequence of data of the same length). It is the key task of the *pre-processing phase*. Next, transformed vectors are delivered to a classifier.

## 4. Preprocessing Phase

Firstly, we need to distinguish a simple press of the right button (e.g. use of the context menu) from a gesture. It is done on the basis of the size of the bounding area of the movements. The diameter of 10 pixels has been experimentally set as the threshold.

A simple reduction of the number of points can be based on the identification of clusters of points e.g. Boukreev [1]. However, we can deform the gesture shape

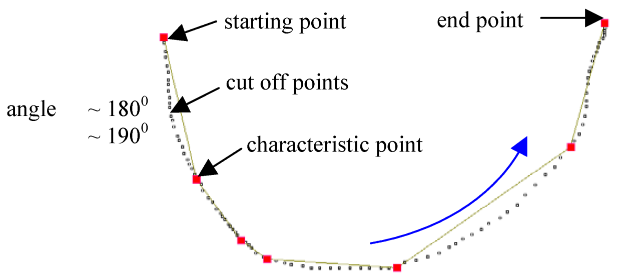

Rysunek 2: Characteristic points for a 'V' shape gesture.

in that way. Instead, one should look for *the characteristic points* that define the significant changes in the direction of a line, see the Fig. 2. To do this, a heuristic *algorithm of identification of the characteristic points* is proposed. It checks whether the angle between the subsequent segments is close to $180^0$. If it is so, the segments are joined. Moreover, if some of the remaining points form a tight group, all of them are removed except one. The algorithm of identification is given below:

```
procedure IdentifyCharPnts(in Points[], out Result[] ){
  LastCharPnt = Points[1];
  addToList ( Result, LastCharPnt );
  for( i = 2 to length(Points)-1 ){
    if ( (significant change in the angle between the segments:
      <LastCharPnt,Points[i]> and  <Points[i],Points[i+1]>)
       and min. distance between: LastCharPnt and Points[i] ){
       LastCharPnt := Points[i];
       addToList (Result, LastCharPnt); } }
  addToList(Result,Points[length(Punkty)]);
  if ( length(Result)> RequestedNum )
    recursively add points to the longest segments in Result
  if ( length(Result)> RequestedNum )
    recursively remove points from Result with the shortest sum
    of the segments, but preserve the points: start and end}
```

On the basis of the experiments, the *significant change in the angle* has been defined as greater than $20^0$. The constraint of the *minimal distance* protects against creation of spatial clusters of points. We want to keep only the points defining the shape of a gesture.

After removing all the points except the characteristic ones, the resulting number can be still different from the one requested by the classifier. In the second part of the algorithm two simple procedures are applied in order to: add some points to the longest segments and remove some points having the shortest segments joined respectively. This simple algorithm works remarkably well in the given domain, and its heuristic parameters were easy to be found experimentally. Its work is illustrated by the Fig. 3



Rysunek 3: The algorithm of spare points removing at work.

As in Boukreev [1] and TyHuNe:90 Tyson et. al. [6], the preprocessing phase is completed by transforming the resulting vector into a new Cartesian space, where the size is constant ($50 \times 40$) and the centre is located in the geometrical centre of the gesture.

## 5. Classifiers

During the development, we have tested two different classifiers. Initially we have applied a standard *back propagation neural network* (NN), e.g. Russel & Norvig [5]. However, as the accuracy was disappointing (see the Tab. 1), we were looking for another classifier. We started with the most simple one i.e. the *k Nearest Neighbours* algorithm (kNN). Surprisingly, it worked so well for the given problem, that is has become the main classifier and NN has been used in the experiments only as a reference point.

The applied NN possesses one hidden layer, the number of the inputs is equal to the number of points in gestures, and the number of the outputs is equal to the actual number of classes of gestures (it means that an addition of a new class causes restart of NN learning). The output values above a defined threshold signal the recognised classes. Finally, the highest value is chosen as the answer.

The data delivered to NN can be encoded as a sequence of:

- cosines values of angles between the subsequent segments,
- changes in the angle between the OX axis and the subsequent segments,
- and polar coordinates of points, transformed to the assumed subspace.

The main advantage of kNN is the lack of a learning phase. Recognition is based on the comparison of a gesture with stored patterns. Thus, the size of a set of patterns must be kept in some limits. The input data are just the sequence of points prepared during the preprocessing phase, and the format of patterns is exactly the same.

Here, kNN works classically and compares an input object $o$ with each pattern and finds the $k$ nearest objects, applying some distance measure. Next, one of the $k$ found objects is chosen as the response. Three different distance measures have been tested:

- standard cosine measure in $m$-dimensional space,
- sum of Euclidean distances between the corresponding points of both objects: the tested gesture and a pattern,
- sum of subsequent distances calculated as Manhattan distance in pixels (between points) in the same way as above.

It is worth to emphasise that the last two metrics are calculated with respect to the order of the sequence. It was motivated by the observation: two sequences of points are more similar, when they are closer each other along the whole shape. All of the applied metrics produce results close to 0 for similar objects.

Tablica 1: The best results achieved by neural network using A12-12 parameters.

| gestures | control points | | |
|---|---|---|---|
| coding | 8 [%] | 16 [%] | 24 [%] |
| cos sequence | 30.56 | 36.11 | 21.78 |
| sequence of changes | 33.33 | 66.67 | 66.67 |
| biegunowy | 11.11 | 83.33 | 85.00 |

## 6. Results

The first learning set was prepared by applying random modifications to singular representants of each class, but the results of learning were poor. Secondly, we have collected learning examples (patterns for kNN) from users, and the results presented in the Tab. 1 and Tab. 2 are based on these data. The following sets of learning examples have been used:

- A12 — basic set consisting of 12 classes of gestures,
- A12-12 — each class extended with examples added by users,
- B22 — extended set of 22 patterns,
- B22-13 — B22 with 13 examples added for each class.

The results are presented in the tables. Surprisingly, even the worst result achieved by kNN is much better than the best result of NN. The significant increase in the number of classes has not caused a significant decrease in accuracy or efficiency of kNN.

## 7. Conclusions

Careful simplification of the notion of mouse gesture has enabled to construct a simple and efficient algorithm of recognition. In the same time, the limited

Tablica 2: The results achieved by the k-neighbours algorithm.

| sets of gestures | control points | | | | | |
|---|---|---|---|---|---|---|
| | 8 | | 16 | | 24 | |
| | Euk[%] | Man[%] | Euk[%] | Man[%] | Euk[%] | Man[%] |
| A12 | 86.11 | 80.56 | 94.44 | 91.67 | 94.44 | 91.67 |
| A12-12 | 97.22 | 97.22 | 100.00 | 97.22 | 97.22 | 97.22 |
| B22 | 81.82 | 80.3 | 90.91 | 90.91 | 90.91 | 89.39 |
| B22-13 | 96.97 | 95.45 | 100.00 | 100.00 | 96.97 | 96.97 |

form of gestures do not limit its properties as a interaction tool. The performed usability tests, see Hofman [3], proved high usability of the solution and eagerness of the users to use the new modality. The recognition mechanism works fast, there are almost no errors in practical recognition, and because of the use of kNN, the user can easily define his new classes of gestures. In the case of significant personal difference in drawing particular gestures, the user can easily exchange the initial patterns for new ones, improving accuracy. The achieved success is mainly due to the preprocessing algorithm. The preprocessing algorithm is so simple, because of the properly defined properties of the gesture modality according to the needs of users.

## Literatura

[1] Boukreev, K.: Mouse Gestures Recognition. CodeGuru, `www.codeguru.com` (December 2001).

[2] Edmonds, A., Vaskovic, P.: Optimoz Mouse Gestures. `http://optimoz.mozdev.org/gestures/` (June 2005).

[3] Hofman, P.: Selected Issues of Artificial Intelligence in the Construction of User Interface to a CASE System. MSc Thesis, Wroclaw Uniwersity of Technology, (2005).

[4] Newman, W., Lamming, M.: Interactive System Design. Addison-Wesley (1995)

[5] Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, Prentice Hall, (1995)

[6] Tyson, H., Hudson, S., Newell, G.,: Integrating Gesture and Snapping into a User Interface Toolkit. In Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST), Snowbird, Utah, October 3-5, (1990).

[7] Yang, T., Xu, Y.,: Hidden Markov Model for Gesture Recognition. Carnegie Mellon University, (1994)