Multistroke Mouse Gestures Recognition in QReal metaCASE Technology

Maria Osechkina
Mathematics and Mechanics Faculty, SPbSU
Saint-Petersburg, Russia
osechkina.masha@gmail.com

Yuri Litvinov
Mathematics and Mechanics Faculty, SPbSU
Saint-Petersburg, Russia
yurii.litvinov@gmail.com

Timofey Bryksin
Mathematics and Mechanics Faculty, SPbSU
Saint-Petersburg, Russia
timofey.bryksin@gmail.com

Abstract - **Approaches for multistroke mouse gestures support are considered in this paper. Presented way of gestures implementation is oriented on creation of new elements on a scene or diagram in meta-CASE system. We propose to generate examples of mouse gestures for elements and to allow user create elements by fast mouse move. We present numerical comparison of gestures recognition algorithms before and after training by k-means algorithm. The proposed approach is implemented in QReal meta-CASE-system.**

Keywords – recognition, multistroke gestures, meta-CASE-system, k-means algorithm

I. INTRODUCTION

World practice has gained considerable positive experience with domain-specific visual techniques for solving problems of industrial software development. In comparison to the traditional approach of "manual" coding we observe 3-10 times increase in productivity on average [4].

There is no significant gain in programmers productivity when using general purpose visual languages such as UML, and therefore domain-specific visual languages become more and more popular. Meta-CASE-systems are designed for quick creation of languages for particular subject areas and generation of tool support for these languages. One example of meta-CASE-systems is QReal [5, 6], which is developed at Software Engineering chair of St. Petersburg State University. To make rapid development of new visual editors in QReal possible a meta-editor was developed. It allows to create metamodels of new visual languages by describing objects on a diagram and associations between them, and defining visual representation of these elements. Then the created metamodel is compiled into a dynamic library and is plugged in QReal in run-time.

The effectiveness of each tool is determined by how easy and fast it performs operations which this tool is intended for. In modeling some of the most frequent operations with objects and relationships in diagrams are their creation and deletion. In many of existing CASE-tools to create a desired object on a diagram one must first either find it on a toolbar or select it from a menu, and then specify position on a diagram where this element sould be placed. Many toolkits provide ability to create an element drag-and-dropping it from a palette. The problem is that the number of diagram types and objects in the palette of each diagram can be quite large (for example, 13 types of diagrams in UML 2.4). It is not always possible to leave on a palette only items that are specific to current diagram, because sometimes there is a need to use elements from different diagrams, for example, for rapid prototyping. Even in case of such basic operations like creation of new element, developer has to make a set of purely mechanical actions and also remember in which tab of a palette or which menu holds desired item. He or she has to constantly switch from thinking about the hierarchy of created models to particular questions of usage of selected tool. We believe that the process of adding items to diagrams could be faster and easier if it involved some alternative ways of man-machine interaction.

As an example of such approach of user interface optimization we consider mouse gestures recognition. The main idea is to associate some actions with specific mouse movements performed with some modifier (for example, with mouse button pressed). These actions are executed immediately after the gesture is recognised. It is desirable that there will be no restrictions on the direction of mouse movements, number of clicks, and the sequence of obtained strokes. In case of CASE-system the action associated with gesture could be creation of element on a diagram at a position where the gesture has been performed. For more similarity with natural process of drawing we want to use recognition of multistroke gestures. This approach does not limit the way a gesture is being drawn.

User interfaces based on mouse gestures recognition are widely used not only in CASE-tools, but also in other areas, such as online handwriting recognition in a variety of text editors (for example, for texting in smartphones), creation of objects on forms, diagrams and scenes in different applications, navigation in some web browsers, controlling the character in some computer games. Some utilities (such as StrokeIt, gMote, etc.) even allow you to add support of mouse gestures recognition into an arbitrary application. Moreover, there are CASE-systems that use similar approach, for example, Visual Paradigm. However, support of mouse gestures recognition in such programs is limited a number of fixed gestures. That is unacceptable for scalable systems like meta-CASE-systems. The paper

discribes an experience of implementing mouse gestures recognition mechanism in an application where the complete set of gestures is not known a priori and can be expanded any time. In addition, for most of known algorithms a long process of training is required and a training set is comparable in size with a test set. Because of a requirement of gestures set extensibility we need to reduce time of creation of training sets to a minimum.

II. DEFINITIONS AND ALGORITHM DESCRIPTION

In general, gesture recognition algorithm works as follows: there is a list of ideal gestures, each of which is associated with an action. Ideal gesture is a gesture pattern that is matched to a user action that should be executed after performing this gesture. User performs a gesture holding a mouse button pressed. After a specific signal (e.g. a timeout, a keypress or release of mouse button in case of single mouse stroke gestures), the gesture should be recognized. For that a list of ideal gestures is searched for a gesture similar to the performed one. If such gesture is found, the action that is associated with this gesture should be executed.

Analysing the ideas behind pattern recognition algorithms it becomes clear that before starting recognition we have to build ideal gestures for each action. In QReal this action is creation of a new element on a diagram. It would be convenient if user gesture would correspond to graphical representation of created object. We decided to generate list of ideal gestures based on this consideration. Graphical representation of each object in QReal is stored in XML files in SVG-like graphical format. This format consists of basic elements like line segments, circles, arcs etc. For line segments are defined by coordinates of their ends; circles — by the top-left and bottom-right corners of a circumscribed square with sides parallel to coordinate axes; arcs are defined by top-left and bottom-right corners of a square circumscribed around the circle, a part of which this arc is, and two angles identifting the start of this arc and rotation to the end of this arc. For constructing ideal gesture it is enough to present described simple figures as gesture strokes. Stroke is performed by mouse movement between pressing and releasing of mouse button. Since number of strokes, order of stroke drawing and mouse movement direction has not to be taken into account in the context of recognition problem, ideal gesture can be represented as a list of lines that correspond to segments, circles, and arcs that form object's graphical representation, without specifying beginnings and ends of these lines and the order of their drawing. When user moves mouse, application receives a signal that the mouse is moving with information about its current position. Since these signals are discrete, line drawn by user, as it is received by application, is a polyline, not a continuous smooth curve, as it may seem to the user. This polyline consists of short segments, which ends' coordinates were obtained receiving signals from the hardware. For constructing ideal gesture strokes we have to represent primitive shapes consisting of short line segments similar to polylines received from the hardware. We have to calculate average speed of mouse movement empirically — average distance between two points, coordinates of which were obtained by successive signals from the hardware. Note that this value can be considered as signals rate on the assumption that signals are sent at regular intervals. Segment is presented as stroke by dividing it into equal segments of appropriate length. A circle is represented by an inscribed polygon with a sufficiently large number of sides. Arc is represented by a polygin inscribed in an arc with equal segments.

III. FEATURE SET

Comparison of lists of points is difficult, since actual list of points received from mouse depends on the hardware, speed of the mouse, stretch of a figure and other factors that are not necessary and sometimes excessive to consider for successful recognition. It is convenient to calculate some features, e.g. numerical measures that correspond to an object and represent its essential properties. We introduce a feature set for which comparison is made. In a space formed by feature sets we introduce a function F with two arguments, where arguments are the feature sets. Let M be the space of feature sets.

$$F : M \times M \to \mathbb{R}$$

This function determines the similarity between two sets of attributes. For example, it may be a probability of the user drawing a particular gesture, or the distance between feature sets. For every ideal gesture a feature set is calculated for later comparison.

IV. MOUSE GESTURES RECOGNITION ALGORITHM

In general, algorithm of mouse gestures recognition can be divided into several stages.

1) Identification of strokes in mouse path. At this step a list of stroke point coordinates is being built. These points are obtained from hardware signals like mouse position when the button was pressed, coordinates of mouse movement with button pressed and position of mouse button release.

2) Path filtering. As a result of user's hand shakes a gesture can become distorted. It is desirable to eliminate this distortion before proceeding to gestures comparison. We have to smooth those parts of the stroke that the user meant to draw as straight lines, and don't smooth those, there the user meant to change movement direction. As a general rule, when a user moves the mouse slowly, he typically wants the movement of the mouse pointer on the display to follow the exact path of the mouse, whereas when he or she

moves the mouse more quickly, the main concern is typically where the pointer ends up, not the exact path to get there. Guided by this consideration, smoothness of the mouse path can be based on mouse velocity.

3) Building of feature vector for user's gesture.

4) Selection of an object. We compute the similarity function for feature set related to user gesture, and each of the feature sets that are related to ideal gestures. In our algorithm this function will serve as the similarity distance. We will choose the ideal gesture according to the results, in our case this is ideal gesture, which is associated with the set of attributes to which the distance is at minimum. If there are several ideal gestures which are close to minimum distance, we shall provide user the ability to choose desired object from them himself. If this distance is less than some threshold, the gesture is recognized and we generate corresponding object.

5) Executing of an action. In case of QReal, when an action is executed, new object have to be created on a diagram. The object is generated in such a way that its center coincides with the center of a rectangle circumscribed around user gesture, with sides parallel to the coordinate axes.

There are training algorithms that allow to adjust recognition algorithm's parameters, including a feature set (or vector) that corresponds to the ideal gestures, in order to improve recognition performance. But for such algorithms we need training sets – a database of user gestures where for each gesturewe know a class to which this gesture relates. Creation of such set is mainly manual process that takes time. In QReal editors for new visual languages can be created very fast, and we do not want to slow down the process of language creation by a need of creation of gesture training database manually. As a general rule, in known solutions the number of items in training set is comparable to the number of items in test set. After some experimenting we decided that training is a necessary step, so we would like to reduce training sets size and time spent on training to minimum. To accomplish this it is desirable that percentage of recognized gestures is high even without training.

V. RASTERIZATION

It is obvious that gesture position should not affect recognition. In addition, in QReal you can stretch elements on the diagram along the coordinate axes, so stretch of the gesture should not affect recognition either. For example, the ellipse and the circle should be recognized as the same object. We introduce the feature set, which is invariant to translation and scaling of a gesture: at first, rectangle with sides parallel to coordinate axes is circumscribed around gesture, then the rectangle is divided into equal cells by lines parallel to the coordinate axes. The number of cells is fixed in advance. Then we create a list of cells, which are intersected by the gesture. Coordinates of cells that contain points of the gesture are considered as a feature. A final list of coordinates is considered as a feature set. We call it basic features set. The introduced feature set is invariant to scale and translation: if we stretch or move a gesture, the corresponding feature set will remain unchanged. Having such a set constructed we got rid of unnecessary information about the number of strokes, the sequence of their drawing and direction of mouse movement. Vast majority of user gesture properties that are independent of the above factors, is also persisted for basic feature set, since a basic feature set is essentially a rasterized gesture with low resolution.

VI. FEATURE SETS AND DISTANCE

We would like to introduce the distance between the basic feature sets, which has some geometrical meaning. Let us analyze how we can interpret the similarity of gestures. Note that two gestures can be called similar if for each object (line segment, circle, cell) from the first gesture we can pick similar object from the second gesture so that they are close enough. I.e. for each cell from the first basic feature set we can indicate close cell from the second basic feature set and vice versa. Based on this consideration Hausdorff distance d_H was chosen as distance between the basic feature sets. Let X and Y are basic feature sets, r is distance in the cell space.

$$d_H(A, B) = \max\{X, Y\}$$
$$X = \max_{a \in A} \min_{b \in B} r(a, b)$$
$$Y = \max_{b \in B} \min_{a \in A} r(a, b)$$

I.e. distance r is introduced in the space of cells. Distance from each cell of the first basic feature set to corresponding nearest cell of the second feature set is calculated and vice versa — for each cell of the second gesture distance to the nearest cells of the first gesture is calculated. Hausdorff distance between the basic sets is maximum of calculated distances between the nearest cells.

*A*s distance between cells we considered the Euclidean distance ($l2$), maximum of absolute differences between the coordinates ($l\infty$), sum of absolute differences of the coordinates ($l1$). The best recognition result is achieved by maximum of absolute differences between the coordinates.

Note that basic feature set cardinality is not constant, so for different user gestures feature sets may have different numbers of elements.

There are training algorithms that involve correction of feature vectors corresponding to the ideal gestures. k-means is one of such algorithms. This algorithm finds center of mass for the elements of space, in which the training is executed. Center of mass for a set of vectors is a vector which coordinates are the arithmetic

mean of the corresponding coordinates of vectors from the training set. It is difficult to imagine the addition of basic feature sets and the division by a scalar keeping the geometric meaning of a basic feature set, since the cardinality of the basic feature set is not fixed.

To simplify the training it is desirable to deal with feature vectors, that are feature sets of fixed cardinality, for which the order of the elements is essential. We propose several ways of constructing a feature vector for the basic feature set. In all of the proposed ways we have to construct a matrix, which is interpreted as feature vector. In all cases, the size of the matrix corresponds to the number of cells across the width and height of the rectangle circumscribed around the gesture, for which this basic feature set is built.

The matrix of distance to the basic feature set

M is a matrix constructed by basic feature set and has dimensions $w \times h$, where the dimensions of the matrix are described above. Element of the matrix *M[i, j]* is equal to the distance *r* from cell with coordinates (*i, j*) to the nearest cell of feature set. Thus we have matrix of distances to basic feature set. Note that we can unambiguously reconstruct the basic set of features by constructed matrix. Basic feature set consists of those and only those cells, coordinates of which are equal to coordinates of matrix elements that are equal to 0. The matrix *M* can be represented as feature vector from space $R^{w \cdot h}$ by writing down all of its lines into one. The distance between the feature vectors is equal to norm of the vectors difference. Introduced function (let us call it *F*) is actually the distance between the feature vectors and between basic data sets.

We prove that for the introduced function the properties of distance in space of basic feature sets are true.

A, B – feature sets,

M(A) – matrix corresponding to set *A*

M(B) – matrix corresponding to set *B*

1) $F(A, B) \geq 0$ - is obvious from the properties of norm.

2) $F(A, B) = 0 \Leftrightarrow A = B$

Thus vector norm is equal to 0 if and only if the vector is *(0..0)*, that is $F(A, B) = 0 \Leftrightarrow M(A) = M(B)$

Since the matrix can be uniquely reconstructed by the basic feature set, the basic feature sets are also equal.

$\Leftarrow$ Clear

3) $F(A, B) = F(B, A)$ This equality follows from the properties of the norm.

$$F(B, A) = \|M(B) - M(A)\| =$$
$$= |-1| \cdot \|M(A) - M(B)\| = F(A, B)$$

4) Correctness of the triangle inequality property follows from the norm definition.

Proposition

r is the distance between cells, that is used for calculation of Hausdorff distance between the basic feature sets. Let the same distance *r* be used as the distance between cells for constructing distances to the gesture. The norm in space of feature vectors is the maximum of absolute elements values ($l\infty$), the sum of absolute elements' values (*l1*) or the square root of the sum of the elements' squares (*l2*). Under these condition Hausdorff distance between the basic feature sets is equivalent to the introduced distance between the basic sets.

Proof

Equivalence is transitive. The $l\infty$ (maximum of absolute elements values) is equivalent to *l1* (the sum of absolute elements' values) and to *l2* (the square root of the sum of the elements' squares). I.e. for proof of proposition it is suffice to prove the equivalence of the Hausdorff distance and $l\infty$.

Let *M1* be the matrix that correspond to the first user gesture, *M2* be the matrix that correspond to the second gesture, *FS1* be the basic feature set for first gesture and *FS2* be the basic feature set for second gesture, *dist(FS1, FS2)* be the introduced distance.

$$dist(FS1, FS2) = \max(|M1_{i,j} - M2_{i,j}|) \geq$$
$$\geq \max_{M1_{i,j}=0 \vee M2_{i,j}=0} |M1_{i,j} - M2_{i,j}| =$$
$$= D_H(FS1, FS2)$$

D_H is Hausdorff distance for basic feature sets.

We got that

$$dist(FS1, FS2) \geq D_H(FS1, FS2)$$

We shall prove that

$$dist(FS1, FS2) \leq D_H(FS1, FS2)$$

We consider elements of the first and the second matrix with coordinates *(i, j)*. By construction of matrix next identity is true:

$$M1_{i,j} = min\{r(O, a) \mid a \in FS1\} ,$$

where *O* is the cell with coordinates *(i, j)*. Let minimum is achieved at point *A1*.

Similarly, $M2_{i,j} = min\{r(O, a) \mid a \in FS2\}$

Let minimum be achieved at point *A2*.

Let $r(O, A2) = b \leq r(O, A1) = c$. The case if $r(O, A2) \geq r(O, A1)$ is analogue.

Suppose that there exists a cell *K1* from the first basic feature set, such that $r(K1, A2) < c - b$.

Then by the triangle inequality

$$r(K1, O) \le r(O, A2) + r(A2, K1) < \\ < b + (c - b) = c$$

It is a point *K1* of the first basic feature set, the distance from which to *O* is less than the distance from *O* to *A1*. It contradicts the selection of *A1* as the cell on which the minimum is achieved. It turns out that for each point *A* of the first basic feature set $r(A2, A) \ge c - b$.

It follows from the definition of Hausdorff distance that $c - b \le D_H(FS1, FS2)$. Due to the fact that the cell *O* was selected randomly, $D_H \ge |M1_{i,j} - M2_{i,j}|$ for all *i,j*. Then $D_H \ge \max |M1_{i,j} - M2_{i,j}|$

We got that

$$D_H(FS1, FS2) \ge dist(FS1, FS2) \ge \\ \ge D_H(FS1, FS2)$$

The distances are equivalent, as required.

Note

When vector norm is considered as the maximum of element absolute values, the Hausdorff distance is equal to the introduced distance.

The number of cells in the rectangle

The idea of following algorithm is calculation of the number of cells in a rectangle of $m \times n$, where *m* varies from 1 to the rectangle height, *n* varies from 1 to the width of the rectangle. Element with coordinates (*i, j)* is equal to the number of gesture cells, that are contained in the rectangle $[0, i] \times [0, j]$. If the gesture intersects the same cell several times we count only one intersection. As in the previous case, the matrix is considered as a feature vector from the space $R^{w \cdot h}$, distance between vectors is defined as norm of the vector difference. Note that by construction of the matrix we can unambiquously reconstruct the basic feature set.

Let *e[i, j]* be equal to 1 if there is a cell (*i, j)* belonging to the basic feature set, otherwise *e[i, j]* is equal to 0. Let *J* be constructed matrix, then $e[i, j] = J_{i,j} + J_{i-1,j-1} - J_{i,j-1} - J_{i-1,j}$. We assume that elements of *J* are equal to 0 outside of gesture rectangle. Note that introduced function is indeed the distance between the feature vectors and between basic feature sets, as there is bijection between the space of constructed feature vectors and the space of basic feature sets. Proof is analogue to case of matrix of distance to basic set.

The number of cells in zone

Feature vector used in this algorithm is similar to previous, but we count cells in the zone, not in the rectangle. *J* is matrix $w \times h$. We define element *[i, j]* of matrix as follow: let *d* be a number of cells in the rectangle $(i-1) \times w$, i.e. in the rectangle, which lies entirely under the cell (*i, j)*, *k* is the number of cells in the zone from a number *(i,j)* from *[i, 0]* to *[i, j]*, $J_{i,j} = k + d$. Note that the basic feature set can be unambiguously recovered from the resulting matrix. Then if we introduce a function of distance between the matrices similar to two previous cases, this function will be the distance between the basic feature sets.

Note that all these feature vectors and distance can be combined by considering a new feature set. The distance between combined feature sets is defined as a linear combination of distances with the non-negative coefficients, at least one of which is strictly positive. This linear combination can be considered as distance between combined feature sets.

VII. TRAINING

We can improve gestures recognition with training: if we have user gesture base, and we know which object should be generated for each gesture, we can correct the parameters of recognition algorithm so that other gestures performed by user will be recognized with greater accuracy. There are a lot of algorithms that allow to improve recognition through training set, for example, Neural networks, the method of *k* nearest neighbors, Bayesian method or k-means algorithm.

The neural network is a system of interconnected processes, that are neurons. Training of neural network involves calculating of coefficients for connections between neurons. But a large training set is necessary for successful recognition via neural network. For example, sometimes number of items in a training set is 1.5 times greater than number of items at a test set [2]. As already mentioned, we would like to reduce the training set's size as much as possible, otherwise the generative approach to gestures creation will be meaningless.

We can use the k nearest neighbors method for training [1]. For a gesture feature set we need to find k nearest neighbors among the feature sets that correspond to user gestures in the training set. The gesture is referred to the class to which the majority of the k neighbors is referred. k nearest neighbors is a time-consuming problem in case of multi-dimensional spaces. In addition, in case of a neural network it is sufficient to store only weights of connections after training, and it is unnecessary to store whole training set unless the need of retraining. But this method does not allow to save memory or to compress training set in any way.

Bayesian method [1] is designed for a variety of symptoms, among which there are no statistical dependencies. For each gesture's feature the probability of this gesture belonging to the given class is calculated basing on the training set. Then the total probability of the gesture belonging to the class is calculated. The most probable class is selected as a result of recognition.

Weakness of this algorithm is limitation to the feature set (lack of statistical dependencies).

Idea of k-means algorithm [3] is to correct feature set corresponding to ideal gestures and the maximum threshold distance between feature set corresponding to user gesture and corresponding to the ideal gesture. Feature vector corresponding to ideal gesture is replaced by a center of mass of feature vectors that are corresponded to the gestures of the training set for given object. Threshold distance is recalculated so that distance from the center of mass to each feature vector of training set is less than threshold distance. In this case, after training we can store only corrected feature set that is corresponding to the ideal gesture, and corrected threshold distance.

In case of a neural network, the percentage of recognized gestures would be high after training with a small training set, if only we guessed optimal values of coefficients of connections between neurons good enough. E.g. we have to create a large training set to use neural network. It is unlikely that after addition of a new ideal gesture the training set could be quickly expanded for it. In contrast, with k-means algorithm it is hoped that it wouldn't require too many elements in the training set. The method of k nearest neighbors, as mentioned, require additional memory cost. And Bayesian method [1] is designed for special feature sets. Therefore, we have selected k-means algorithm for training.

Typically, the center of mass of a few vectors is calculated as sum of these vectors divided by the number of points. The problem of feature sets of non-fixed cardinality is that it is difficult to determine sum of feature sets and division by a scalar, so that geometric meaning of feature set was kept and training gave results.

For construction of center of mass of feature sets of non-fixed cardinality the following properties of center of mass were used:

1) Center of mass of one element is equal to this element.
2) Let M_n be center of mass of a set of n elements. We add element *k* to the set.

Let $d = dist(M_n, k)$ be the distance between the center of mass and the added element.

Let M_{n+1} be the new center of mass.

$$dist(M_{n+1}, M_n) = \frac{d}{n+1} = d0_M$$

$$dist(k, M_n) = \frac{d \cdot n}{n+1} = d0_k$$

We construct a point, that is located at distance $d0_M$ from point M_n, and at a distance $d0_k$ from point *k* (Property 2). In general, such point may not exist in a metric space. Then we construct a point, the ratio of the distances to which from M_n and *k* is the closest to desired ratio. Several points may satisfy this requirement, it is important that in this case the algorithm should return at least one such point.

Suppose we have two basic feature sets at a distance *d* from each other, and we have to build a basic feature set at distance $\frac{d}{n}$ from the first set, and $\frac{d \cdot (n-1)}{n}$ from the second. For calculating distance between two basic feature sets we consider pairs of cells that are at the minimum distance from each other. For each cell from the first basic feature set we find the nearest cell from the second base set, and for each cell of the second basic set nearest cell from the first basic set is found. We consider obtained pairs of cells as endpoints to construct the required set on a plane. Then we get a point, that divides the segment in a given ratio of $\frac{1}{n}$ from the first cell, if the first cell is referred to the first feature. Otherwise ratio is equal to $\frac{(n-1)}{n}$. The resulting points are interpreted as festures, and they can have non-integer coordinates. Basic feature set is consisted of points, distance to which from two given sets is equal to required value.

Center of mass of a set of *n* basic feature sets is constructed by induction. One basic feature set is equal to its center of mass. Let the center of mass of the set of *k* basic feature sets is already constructed. We add another basic feature set and calculate the distance *d* between it and the center of mass. By above-described method we construct new center of mass so that the distance from it to old center of mass is equal to $\frac{d}{k+1}$, and the distance from new center mass to the added element is equal to $\frac{d \cdot k}{k+1}$.

During the training we should change not only the center of mass, but the maximum distance from the center of mass of the class to a feature sets for the gestures of a given class. It is necessary to correct the maximum distance so that the sphere $B_n(M_n, D_n)$ would belong to the sphere B_{n+1} with center M_{n+1} and radius D_{n+1}. Added feature set should belong to the new sphere. Thus $D_{n+1} \geq \frac{d \cdot n}{n+1}$. Let $b_0 \in B_n$, B_n is feature set. By triangle inequality

$$dist(b_0, M_{n+1}) \leq dist(b_0, M_n) + dist(M_n, M_{n+1})$$

We recalculate maximum distance based on above inequalities

$$D_{n+1} = max(\frac{d \cdot n}{n+1}, D_n + \frac{d}{n+1})$$

VIII. EXPERIMENTS

For testing we selected ten objects that are used in QReal diagrams. The set of ideal multistroke gestures was generated by graphical representation of these objects (pic 1 – pic 10). Six people created training and test sets to test the effectiveness of different algorithms. There are1545 gestures in test set and 100 gestures in training set.

Number of cells in circumscribed about the user gesture rectangle that is used to construct the basic feature set is equal to 81. Recognition results are shown in the table 1.

The dependence of the recognition from number of cells in the rectangle (similar to image resolution) for combination of two algorithms is shown in the table 2. Note that after some threshold recognition accuracy does not increase with the size of the rectangle, but the speed drops due to the complexity of the algorithm. So we selected rectangle sizes equal to 81.

Percentage of recognized user gestures is very different for different objects. Recognition depends on similarity between different ideal gestures: if two ideal gestures are similar enough, user gesture that is corresponding to one ideal gesture can be confused with another. Algorithm based on matrix of number of cells in rectangle has a low general percentage of recognition, but some gestures that are poorly recognized by other algorithms, are well recognized by it. It was decided to combine feature vector from this algorithm with the feature vector that gives the best summary recognizability. Coefficients for linear combination of distances were chosen empirically, they are equal to 0.2 and 0.8 for algorithm based on matrix of number of cells and algorithm based on matrix of distance to the basic feature set respectively. Results of gesture recognition by combination of feature vectors are given in the last two rows of table 1.

Increasing of the percentage of recognized gestures depends not only on the training set, but the method of constructing feature vector. For example, feature vector based on matrix of number of cells in the zone shows improvement in recognition of 19% after training. While feature vector based on matrix of number of cells in rectangle shows recognition improvement of only 6%, but it shows better recognition percentage before training.

IX. CONCLUSION

Several algorithms of multistroke mouse gestures recognition have been tested. Necessary condition for all these algorithms is recognition independence of the number and order of gesture strokes and mouse movement direction.

After testing of algorithms without training we concluded that recognition without training doesn't satisfy the user. k-mean training algorithm has been applied for feature vectors of fixed dimension and was extended to the feature set of non-fixed cardinality. However, the initial recognition threshold and speed of training allows developers to provide the ability of training application online. Retraining after insignificant expansion of ideal gesture set is unnecessary. Initial detection threshold is the percentage of gestures recognized without training, the speed of training is the ratio of gestures recognized with training to gestures recognized without training. At the moment, the best recognition percentage after training is 91% on the test set of 1545 user gestures.

REFERENCES

[1] Cesar F. Pimentel, Manuel J. da Fonseca, Joaquim A. Jorge, «Experimental evaluation of a trainable scribble recognizer for calligraphic interface» // Graphics recognition: algorithms and applications, eds. Dorothea Blostein, Young-Bin Kwon, Ontario, Canada, 2001, pp. 81-85

[2] Don Willems, Ralph Niels, Marcel van Gerven, Louis Vuurpijl, «Iconic and multi-stroke gestures recognition» // Pattern recognition, Vol 42 Issue 12, pp. 3303-3312, December, 2009

[3] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; Piatko, C. D.; Silverman, R.; Wu, A. Y., «An efficient k-means clustering algorithm: Analysis and implementation». IEEE Trans. Pattern Analysis and Machine Intelligence Vol. 24 Isuue 7, pp 881–892, July, 2002

[4] Kelly, S., Tolvanen, J.-P., «Visual domain-specific modeling: benefits and experiences of using metaCASE tools», DSM Forum, URL http://www.dsmforum.org/papers/Visual_domain-specific_modelling.pdf

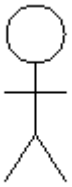
[5] Timofey Bryksin, Yuri Litvinov, Valentin Onossovski, Andrey N. Terekhov, «Ubiq Mobile + QReal - a technology for development of distributed mobile services», unpublished

[6] Timofey Bryksin, Yuri Litvinov, «QReal metaCase technology overview», unpublished

Pic 1: AcceptEventAction

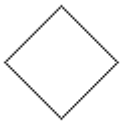
Pic 2: AcceptTime EventAction

Pic 3: ActivityPartition

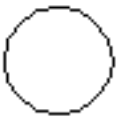


Pic 4: Actor

Pic 5: Comment



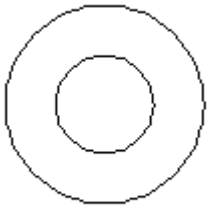
Pic 6: Decision Node



Pic 7: InitialNode

Pic 8: InputPin

Pic 9: SendSignalAction



Pic 10: FinalState

Algorithms	Picture										All gestures
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
User recognition	160	202	177	84	152	187	210	161	148	64	1545
Hausdorff distance	143	179	165	34	21	174	205	6	76	57	1060
Hausdorff distance + training	140	171	167	61	125	181	200	83	113	60	1301
Matrix of distance to basic feature set	141	176	171	39	21	178	204	19	80	58	1087
Matrix of distance + training	141	197	168	81	123	181	194	97	129	60	1371
Cells number at rectangle	45	144	169	10	85	51	193	8	82	33	820
Cells number at rectangle + training	79	112	171	28	112	99	164	70	41	39	915
Cells number at zone	112	169	166	5	15	61	205	7	44	25	809
Cells number at zone + training	122	114	174	44	133	142	185	69	62	49	1094
Combination of feature vectors	144	194	177	20	92	161	205	17	86	59	1155
Combination of feature vectors + training	140	197	175	74	149	180	195	126	115	59	1410

Table 1: Recognition results

Rectangle width and height, cell	**32**	**40**	**50**	**60**	**70**	**81**	**90**
Recognition, %	60	69	72	73	74	75	74

Table 2: Dependence of recognition on rectangle size