Motion Data Preprocessing in Robotic Applications

Peter Benický, Ladislav Jurišica, Anton Vitko

Slovak University of Technology in Bratislava
Faculty of Electrical Engineering and Information Technology
Institute of Robotics and Cybernetics
Ilkovičova 3, 81219 Bratislava, Slovakia
e-mail: peter@benicky.info, ladislav.jurisica@stuba.sk, anton.vitko@stuba.sk

Abstract: There are numerous approaches to processing images of moving bodies. As to robotic applications call for real time algorithms, the redundant data obtained from the input image should be filtered out and implemented with deep knowledge of hardware specifications. The paper demonstrates a new approach to real-time motion classification. Partial problems resolved in this paper cover the fast motion detection, edge detection, corner detection and finally motion classification. The motion, edge and corner detectors are optimized with respect to effective use of the memory and excluding redundant data.

Keywords: robotics, motion, real time, identification, in memory database

1. INTRODUCTION

If an adequate mathematical model of a system is not available, the designer should make recourses to abstract methods. The image processing is one of such problems. The image processing has been realized by means of artificial intelligence implemented on neural networks of various kinds. However, the neural processing of image data is time consuming and very often it does not meet real time requirements. In real time applications the processor should process 24 or more images per second. Therefore this paper is focused on solving the problem of motion detection and classification by different means then those used so far. The suggested approach tries to resolve the problem of improving quality of detection so as the detected edges may be used for subsequent corner detection and fast motion identification or classification tasks.

Real time object tracking in robotics applications is extremely important in the area of visual servoing where the presence of time delay causes a significant problem. The performance of object tracking can be increased using Kalman Filter (Liu et al., 2012).

Among the existing edge detection methods the Canny method is probably the most popular one and it has various implementations. One of its fast implementation is in OpenCV library (OpenCV), representing an assembly of optimized Computer Processing Unit (CPU) implementations. Another implementation can be achieved by Graphical Processing Unit (GPU), using the programming model (Compute Unified Device Architecture or CUDA) released by NVIDIA. Using this implementation the Canny edge detection becomes more than 20% faster than assembly optimized CPU implementation. (Luo and Duraiswami, 2008). The GPU allows for processing even larger amounts of data in real time, for example the transformation from 2D into 3D representation using range imaging sensors (Wasza et al., 2011). The corner detection by GPU resulted into significant speed-ups (Teixeira et al., 2008). Another approach of real-time edge detection using CPU rests in computing a set of anchor points on the image and drawing edges between them (Topal et al., 2010). Such approach produces 1-pixel wide edges with real connectivity in real time (10msec for a VGA image on a 2GHz CPU in single thread).

The work (Wang, 2006) is focused on the performance evaluation of the edge detection in the tasks of recognition of a human behavior using infrared lighting (for night vision). The results are automatically in the grayscale format. By removing some mathematical operations, like finding square roots and reduced number of divisions, the performance becomes ten-times faster than conventional Sobel detector. It can be used for night monitoring of the patients behaviors in a hospital. The approach described in (HANA) is based on using fast Random Access Memory (RAM) in all data storage applications. It was implemented in HANA (product of SAP Company). The HANA is basically “in memory database” with column or row storage making applications much faster.

The authors are not aware of any work dealing with the one column store “in memory” approach to the edge detection of a moving object which would minimize redundant and useless information for achieving high processing speed. That is just what is required in robotic applications. It can be reasonably expected that “in memory” implementations (not only databases) can play a significant role in the endeavor of reaching high performance. These facts have motivated us to adopt such an approach in developing a real time processing system for robotic applications.

2. MOTION DETECTOR

To achieve motion detection of acceptable quality the detecting algorithm should be considered together with its implementation. Note that using a slow detection algorithm the final result is of poor quality even for a good implementation. To substantiate this, let us calculate (for each element of the array $\_array$) the value of a testing function, for example the function (1). To achieve more accurate measurements the algorithm is executed 10000 times. Note that the algorithm must be compiled in the release mode.

```
for (int k = 0; k < 10000; k++)
 for (int i = 0; i < 255; i++)
  for (int j = 0; j < 255; j++)
    _array[i, j] = f(i,j);
```

Fig. 1. Experiment for performance testing.

For instance, the execution time for calculating the two-parametric function (1) $i, j \in \langle 0 \quad 255 \rangle$ by the specific hardware (Pentium with $1{,}6GHz$ microprocessor, RAM $512MB$) was $10{,}87$ seconds. Let us note that our goal is to achieve higher performance by the algorithm rather then by the hardware.

$$f(i,j) = \frac{i}{i+j+1} \tag{1}$$

Therefore let us rewrite the function (1) into an array representation with already pre-calculated values and suppose that in the run time no calculations, except for reading from the two dimensional array, are performed. This led to the execution time $6{,}7$ seconds using the same hardware. After rewriting the two dimensional array into one dimensional array $f(k.i+j)$, where k is a number of elements in one row and using one-dimensional array of pre-calculated values, the execution time decreased to $3{,}49$ seconds. With a correct implementation, the algorithm may be 3 times faster. This is especially important when considering real time motion detection.

However, our main goal was to obtain characteristic features that would better describe input data. The characteristic feature means, for instance, the curvature of the detected motion. For detection of the edges, i.e. locations where the contour's curvature is greater than a given threshold the contours of the detected areas must be known.

3. EDGE DETECTOR

Standard methods like Laplace edge detector (Fig. 2) do not provide contours that could be applicable in the tasks of curve detection. For this purpose one needs contours of one pixel wide and without unreasonable line discontinuity. As to the solution should meet the real time processing requirements, the processing time should be considered as well.

Fig. 2. Input to the edge detection (left) and the result of the detected edges by the standard Laplace edge detector (right).

Since the inputs to the curve detector are binary data (binary picture) we used the method based on masks was considered, which is faster than any standard of edge detection method.

1	2	3
4	5	6
7	8	9

Fig. 3. Matrix of size 3x3 as a reference template for the mask based method.

The edge detection of a binary picture with the mask based method is defined as follows: the pixel "5" (Fig. 3) is a part of the edge (line) marked as a black point if the pixel has at least one white pixel in its direct neighborhood ("2", "4", "6", "8") (the white pixel is not a part of the detected edge) or there is at most one black pixel among all eight pixels around it. ("1", "2", "3", "4", "6", "7", "8", "9"). However, the mentioned rules do not provide contours of sufficient quality as required by the task the subsequent corner detection. To ensure higher quality contours the set of rules should be widened. New rules were found empirically using the set of input binary images. Using this approach the 89 fix patterns were found. Using all these patterns resulted into improved robustness of the edge detection and provided higher quality of the pixel-wide edges.

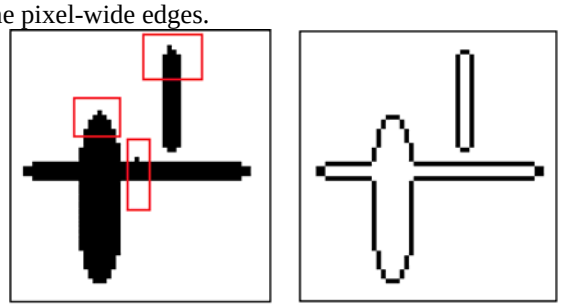

Fig. 4. Input image (left) and detected edge with correction by pattern fixing images (right).

Another advantage of this method (besides the quality of detected edges), rests in the possible optimization during implementation, which is aimed at obtaining the real time processing system. The number of 89 fix patterns can be considered as 9 bit information per pattern, meaning that this information can be coded by 32 bit integer (in C++ language) for one fix pattern.

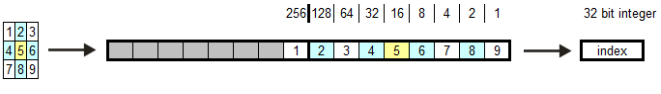

Fig. 5. Binary pattern image can be represented by a 32bit integer where 9 bits are used.

In this way 89 integers were obtained, each representing one binary edge describing pattern. To cover all combinations of 9 bit information an array of size 2^9=512 is needed. (Access to one dimensional array is faster than to multidimensional array).

```
int n = matrix[i - 1, j - 1];
n += matrix [i - 1, j - 0] << 1;
n += matrix [i - 1, j + 1] << 2;
n += matrix [i - 0, j - 1] << 3;
n += matrix [i - 0, j - 0] << 4;
n += matrix [i - 0, j + 1] << 5;
n += matrix [i + 1, j - 1] << 6;
n += matrix [i + 1, j - 0] << 7;
```

Fig. 6. Implementation of 3x3 matrix into integer that will be used as index.

Finally, after getting a 3x3 matrix from the input binary image it is converted into 32 bit integer (Fig. 6) and the information whether the pixel "5" is edge or not will be simply decided by looking into the array of integers.

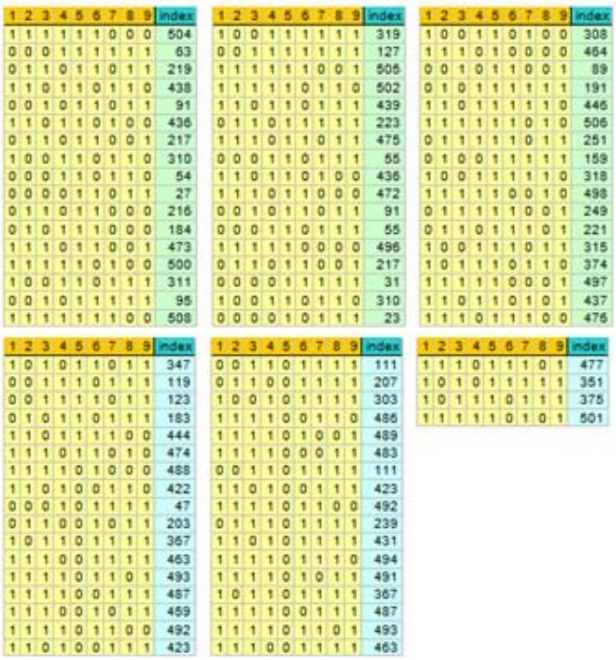

Fig. 7. Fix patterns and 9 bits information per pattern and its index.

/logical operation is required since the look up table of integers is prepared before starting the entire algorithm. Finally, by that approach the high quality moving contours (continual lines of a pixel-wide) can be quickly obtained.

Fig. 8. Input image (left) and the result of detected edge with correction by references images on real input data (right).

When looking at the edges in Fig. 8, it is obvious that data are not invariant w.r.t. rotation. The approach described is also applicable on the exact motion identification from a fixed position of the observer. A picture similar to the Picasso's one line hand sketch of a bull can be obtained. By a multi-erosion of the input picture all partial surfaces can be connected together into one. It is done on expense of the lost of unimportant details.

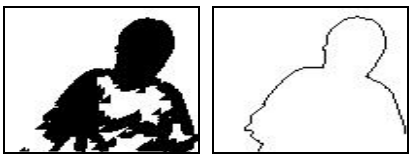

Fig. 9. Motion detection with multi-erosion (left) and detected longest edge (right).

Having all parts connected the longest edge will describe the object in the same way as Picasso's hand sketches do. Then having the longest pixel-wide edge describing the moving object as one unit, the rotation and zoom invariant characteristic features can be calculated. As a means of the curves detection the M2003 algorithm was used (Mayed et al., 2012). To obtain invariant features, the object center is calculated by expression (2).

$$x_c = \frac{1}{T}\sum_{k=1}^{n-1} x_h[k],\ \ y_c = \frac{1}{T}\sum_{k=1}^{n-1} y_h[k] \tag{2}$$

Where $[x_h, y_h]$ is h-th point of the edge. The relative distances can be then calculated by expression (3).

$$d_R[k] = \frac{d[k]}{\frac{1}{N}\sum_{i=1}^{N} d[i]} \tag{3}$$

Where $d[k]$ is k-th absolute distance of the corner from the object center. It is calculated in accordance with (4).

$$d[k] = \left\{\sqrt{[x[k]-x_c]^2 + [y[k]-x_y]^2}\right\} \tag{4}$$
$$k = 1,\ldots,n-1$$

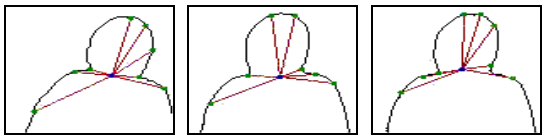

Fig. 10. Invariant motion descriptors are defined as the distance between the detected corner and the centroid of the moving object.

When the human finds itself inside of a new scene, he or she first considers the biggest moving object. The above mentioned approach describes the biggest moving object (or the linked group of objects) by one line that describes the contour by invariant motion descriptors. The approach can be used even in situations where besides the main moving objects also other smaller moving objects are in the background as shown in Fig. 11.

Fig. 11. Detected motion of hand (in the middle) with complex background of moving trees. Detected corners with relative distances between centroid and corners on the longest detected edge (right).

Fig. 12. Test experiment of taking flowerpot with flower and its movement from left to the right.

The algorithm was used to obtain invariant relative distances of the longest detected motion line. Since the algorithm implementation of the algorithm was done with respect to the known properties of the hardware used it is relatively fast. In order to evaluate it in similar experiments a simulation of the movement of the flowerpot with flower (Fig. 12) from the right corner to the centre of the window after which the hand was removed without flowerpot was done. The invariant relative distances calculated by expression (3) for action shown in Fig. 12 are described in Fig. 13.

Fig. 13. Detected relative distances of moving flowerpot.

The invariant relative distances of such action are shown in Fig. 14 (left). Now let us imagine that the camera had been recording a hand moving towards the flowerpot but at a certain moment the hand was returned back without grasping the flowerpot. The invariant relative distances of this action can be seen in Fig. 14 (right). Finally, the same action was simulated but it was done much faster. The invariant relative distances are shown in Fig. 14 (middle). It can be seen that there is certain similarity between Fig. 13 and Fig. 14 (middle).

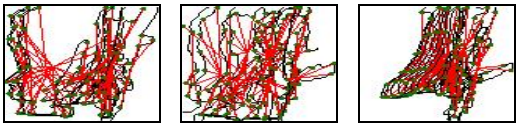

Fig. 14. Detected relative distances of other similar actions.

Invariant motion descriptors may be created easily without an excessive computational burden. They are also useful for the object identification or classification.

4. STORING THE INVARIANT MOTION DESCRIPTORS

Having the invariant motion descriptors known, the next important task is to use them effectively in terms of their storing and reading in a database. To work with the database effectively means using the correct fields indexing and table relations according the data. Also the principle of parsimony should be applied, meaning that data should not be redundant and not more accurate than it is required. In our case it means that the variable representing invariant motion descriptors should not be stored as double types. For that reason they were converted into the integer types.

$$y_{\text{integer}} = Round\left(k.x_{real}\right) \tag{5}$$

To calculate parameter k one needs to know the minimal and maximal value of x_{real}, which (in our case) will be represented by minimum and maximum value of relative distance from the corner of the detected edge to its centre calculated by the expression (3).

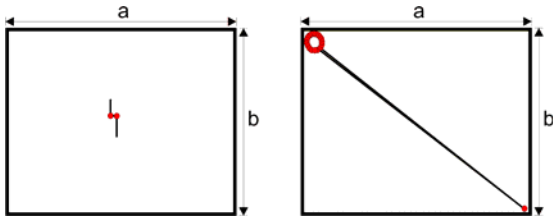

Fig. 15. Minimal relative distance calculated by expression (3) (left) and maximal relative distance calculated by expression (3) (right).

From Fig. 15 (left) it is obvious that minimal relative distance calculated by (3) is equal to 1. In order to simplify expression (5) let us suppose that the parameter k is within the following interval of integers.

$$0 \le k \le 65536 \tag{6}$$

To calculate parameter k the maximum relative distance calculated by expression (3) will be considered. Let' us imagine a theoretical situation shown in Fig. 15 (right), where an infinite number of points are located in a circle of radius 1 pixel in one corner of the picture and just one point is located in the opposite corner, thus we can achieve the maximum relative distance calculated by expression (3). This maximum relative distance will be represented by integer value of 65536. The maximum relative distance calculated by expression (3) based on the situation shown in Fig. 15 (right) will be calculated as the longest absolute distance of an edge point from the edge centroid and the average absolute distance of all edge points from the edge centroid as mentioned in expression (7).

$$d_{R\max} = \frac{d_{\max}}{\frac{1}{N}\sum_{i=1}^{N} d[i]} = \frac{\sqrt{a^2+b^2}}{\lim_{n\to\infty}\left(\frac{\sqrt{a^2+b^2}+\sum_{i=1}^{n}1}{n+1}\right)} \tag{7}$$

The integer parameter k will be calculated from the condition (8).

$$\frac{\sqrt{a^2+b^2}}{\lim_{n\to\infty}\left(\frac{\sqrt{a^2+b^2}+\sum_{i=1}^{n}1}{n+1}\right)}.k \overset{!}{=} 65536 \tag{8}$$

The expression (8) can be simplified by calculating the limit as mentioned in expression (9).

$$\begin{aligned}&\lim_{n\to\infty}\frac{\sqrt{a^2+b^2}+\sum_{i=1}^{n}1}{n+1}=\\&=\lim_{n\to\infty}\frac{\left(\frac{\sqrt{a^2+b^2}}{n}+1\right).n}{\left(1+\frac{1}{n}\right).n}=\\&=\frac{(0+1).n}{(1+0).n}=1\end{aligned} \tag{9}$$

Hence, the parameter k will be calculated in accordance with the expression (10).

$$k=\frac{65536}{\sqrt{a^2+b^2}} \tag{10}$$

In case of using non-integer type of data the parameter k should be recalculated with respect to the range of values of the using data type.

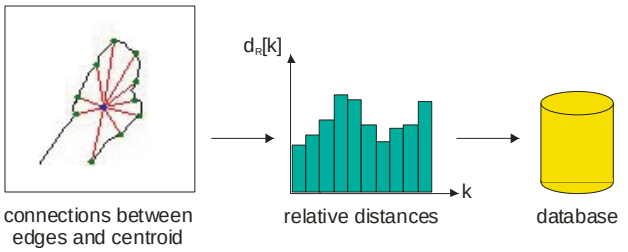

Fig. 16. Process of writing invariant descriptors of moving hand into database of knowledge.

The main reason for the need of effective use of data types is using smallest memory so as to allow for the fastest computer memory, namely the RAM memory. The database which is exclusively located in RAM is known as “in memory database”. It is much faster than the database which is only partly stored in RAM, meaning that the rest of data is stored on a hard disk. Having “in memory database” with one dimensional arrays the extremely high performance can be achieved.

5. DESIGN OF DATABASE MODEL

As to the database model can contain large amounts of data and their processing is required in real time, in the design stage it is necessary to consider various requirements (e.g. low data redundancy, correct use of data types, correct draft indexes) with regard to known application database tables. Also utilization of all means which would secure higher performance of the database system, for instance so called storing procedures should not be omitted.

The suggested structure of the movement defining table is displayed in Table 1. Some space is reserved for additional descriptors of the contours of the motion contours.

Fig. 17. Amounts of dynamic pixels and information on the total length of edges may be obtained during the preparation of input data.

During data preparation it is possible to get a few useful descriptors which are readily available and easily calculated. For instance, during the calculation of dynamic pixels one can implement a counter of such pixels. This, so-called a number of dynamic pixels, describes the “size” of the movement. This descriptor is, however, dependent on the distance of the object from the camera.

Table 1. Movement defining database table named “MotionDefinitions”.

Column name	Data type
Motion	Integer data type
Frame	Integer data type
Edge	Integer data type
Corner	Integer data type
CornerDistance	Integer data type
CornerCount	Integer data type
CornerDistanceCount	Integer data type
ParameterN	Integer data type

However, if a certain degree of invariance should be ensured, we can put the number of dynamic pixels to the ratio of the length of edges as indicated in the column name "Edge" (Table 1) which was already calculated because of the selection of the longest edge where the length of detected edge was calculated. This descriptor, for the purpose of draft tables will be named as "ParameterN". Note that by a similar simple way it is possible to get more descriptors (for example, the overall curvature of the edges, calculated as the sum of the curvature at each point which is situated on the edge).

From the Table 1 it is evident that each of the trained movements (the movement registered in the table "MotionDefinitions") is represented by a unique ID number in the column "Motion". Every movement is composed of several pictures "Frame" and every frame has few edges "Edge". Finally, the edge contains several corners "Corner". Then for each of the "Motion", "Frame", "Edge", "Corner" combinations there exists a relative distance between a corner

on the edge and the edges' Centre of gravity. The relative distances are shown in the raw "CornerDistance".

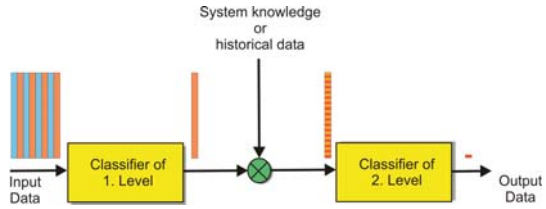

Fig. 18. Multilevel classifier.

Due to using the multilevel classifier (Fig. 18) it is convenient to define the descriptors also on the level of the images, i.e. for a combination of "Motion" and "Frame" values.

6. MOTION IDENTIFICATION

At the first level identification the similar images are identified on the basis of global descriptors describing an image as a whole. The motion identification is based on finding images with similar global descriptors. Similarity of global descriptors can be calculated as the percentage of the deviation from the model in accordance with (11).

$$\chi[\%] = \left| \frac{D_{pattern} - D_{in}}{D_{in}} \right| .100 \qquad (11)$$

Symbol D_{in} means a descriptor calculated on the basis of input data and stored in the knowledge database.

For the sake of simplicity we only consider the longest edge and two descriptors describing the image as a whole. One descriptor represents number of detected corners on the longest edge and the second one represents the sum of all relative distances between corners and the centre of gravity of the longest edge. Using this approach, one can use the following SQL command for obtaining all similar images from the knowledge database. Similar images are those where all descriptors at the image level have smaller percentage of the deviation then empirically determined deviation for the given descriptor.

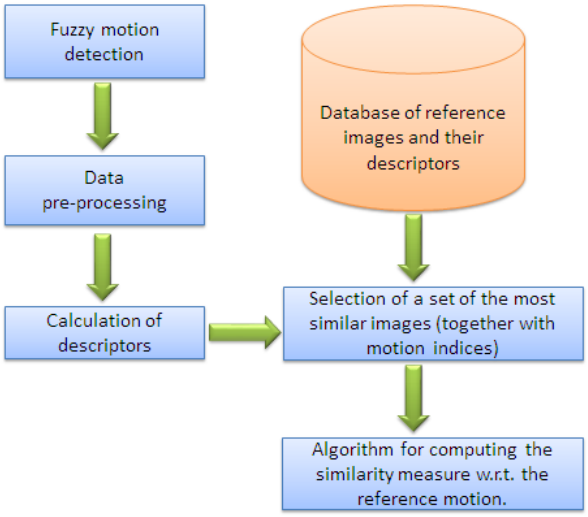

Fig. 19 Block diagram of the input data identification

As an example let us consider the input image with 10 detected corners and with the sum of all relative distances is equal to 2545.

```
SELECT MotionID, Frame, Edge
FROM MotionDefinitions
WHERE
(abs(CornerCount-10)/10)<0.10
AND
(abs(CornerFeatureCount-
2545)/2545)<0.2
GROUP BY MotionID, Frame, Edge
```

Fig. 20. The SQL command for identification all similar images.

Now we should obtain (from the knowledge database) all similar images for which the deviation of the descriptor of the number of corners (*CornerCount*) is less than 10% and the deviation of the sum of relative distances is less than 20%. The SQL command shown in Fig. 20 is not effective because the SQL server must calculate the deviation of the descriptors for all records. The command may be optimized so as the SQL server does not need to do any calculations, except for the simplest mathematical operation, namely the comparison.

```
SELECT MotionID, Frame, Edge
FROM MotionDefinitions
WHERE
CornerCount between 9 and 11
AND
CornerFeatureCount
between 2036 and 3054
GROUP BY MotionID, Frame, Edge
```

Fig. 21. Optimized SQL command for identification of all similar images.

The optimized SQL command (Fig. 21) will constitute the identifier of the first level, i.e. (at the level of images.) It will identify those pictures that have global descriptors in the permitted value ranges. In general, one can assume that for any couple of the similar images their descriptors must be similar as well. This is a necessary condition for the existence of a similarity between the images. However the opposite is not true.

By using the optimized SQL command one can get similar images. After that the images can be ordered in accordance to the similarity based on the number of corners. Ordering based on the sum of the relative distances of the detected corners may be the same for different combinations of the corners. For this reason the identified images will be ordered by the number of corners only. In this way the similar images will be ordered from the most probable image the lowest probable image.

It is necessary to note that even this ordering does not necessary correspond the actual ordering. For instance, the image with other number of detected corners may be more similar with the input image as the image with an identical number of detected corners. It has been found by the experimental testing procedure that the images that had the same number of detected corners or those that had virtually the same number of detected corners as the input image were actually similar.

```
SELECT MotionID, Frame, Edge,
Min(Abs(CornerCount - 10))
      as Deviation
FROM MotionDefinitions
WHERE
CornerCount between 0 and 11
AND CornerFeatureCount
      between 2036 and 3054
GROUP BY MotionID, Frame, Edge
ORDER BY Deviation
```

Fig. 22. Optimized SQL command identifying all similar images which are ordered in accordance to the degree of similarity.

Table 2. Table of comparisons of input image with images stored in database.

Input motion ID	Input image	Input edge	Identified movement ID	Identified imae	Identified edge	Deviation in number of crners	Deviation in sums of relative distances
2	96	1	1	5	1	0	0
2	96	1	1	9	1	0	0
2	96	1	1	20	1	0	0
2	96	1	1	22	1	0	0
2	96	1	1	28	1	0	0
2	96	1	1	40	1	0	0
2	96	1	1	54	1	0	0
2	96	1	1	68	1	0	0
2	96	1	1	74	1	0	0
2	96	1	1	77	1	0	0
2	96	1	1	85	1	0	0
2	96	1	1	88	1	0	0
2	96	1	1	90	1	0	0
2	96	1	1	93	1	0	0
2	96	1	1	99	1	0	0

The Table 2 shows descriptors of the image 96 of input motion with index 2 that was calculated as similar motion to the motion stored in database as motion with index 1. Images from the knowledge database that have been identified as being similar to the input image numbered as 96 are ordered in accordance to deviation w.r.t. number of the detected corners. In particular, these pictures are from the knowledge database of the motion of indexed as 1, that is those with serial numbers 5, 9, 20, 22, 28, 40, 54, 68, 74, 77, 85, 88, 90, 93, 99. In this case the permitted value of deviation of the number of detected corners was set to zero (zero tolerance of detected corners).

The images presented in Fig. 19 can be considered as outputs of the first level classifier. These pictures can be also analyzed at the second level, i.e. with respect to the deeper details, for instance at the level of individual edges, corners and relative distances (Table 3).

Table 3. Table of relative distances of corners with the longest edge in image that it is stored in the database.

Motion ID	Corner Relative Distance	Frame	Edge	Frame Feature ID	Corner Count	Corner Feature Count
1	145	88	1	2	10	4510
1	386	88	1	5	10	4510
1	398	88	1	6	10	4510
1	427	88	1	7	10	4510
1	430	88	1	11	10	4510
1	465	88	1	10	10	4510
1	521	88	1	8	10	4510
1	522	88	1	4	10	4510
1	571	88	1	3	10	4510
1	645	88	1	9	10	4510

From set of pictures that were obtained from first level classifier can be analysed the picture number 88 from knowledge database (Table 3).

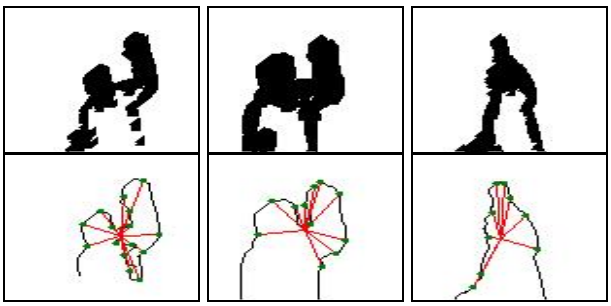

Fig. 23. Object with detected corners (left), the same object turned by β angle (in the middle), and finally the same object turned by γ angle (right) with detected corners.

There is no need to consider an identification numbers of detected corners in order to calculate the similarity between the pictures based on individual corners and theirs relative distances from the center of gravity of the object. That is because the same object turned by β or γ angle can have different values of relative distances from the center of gravity for individual identification numbers of detected corners. Therefore, the objective will be finding the maximum number of relative distances from the set of data in input image that will have similar relative distances as the reference picture stored in the database will have. It means that the objective is to find the most relative distances from the set of data in input image, which will have similar relative distances as the reference image stored in the database. For example the similarity between input image number 96 and reference image number 88 stored in knowledge database as motion number 1 is calculated below. The relative distances of the input image are shown in Table 4 and that of the reference image are shown in Table 3.

Table 4. Relative distances of corners with the longest edge in input image.

Input Motion ID	Input Corner Relative Distance	Input Frame	Input Edge	Input Frame Feature ID	Input Corner Count	Input Corner Feature Count
2	226	96	1	9	10	4510
2	316	96	1	4	10	4510
2	346	96	1	10	10	4510
2	371	96	1	6	10	4510
2	416	96	1	5	10	4510
2	451	96	1	8	10	4510
2	551	96	1	11	10	4510
2	592	96	1	7	10	4510
2	600	96	1	3	10	4510
2	641	96	1	2	10	4510

As in the case of the first level classifier, which is based on global descriptors (number of corners detected, the amount of relative distances), also the second level classifier based on each relative distances, the percentage of acceptable deviation of the relative distances of the reference image is

used. Please note that there are several solutions within second level classifier.

Finally the algorithm for similarity calculation may be summarized as below:

1. For the relative distance of the input image $d_{Rv}[i]$, it searches for such relative distance between relative distances of reference image, which, regarding its length is the most similar in accordance with the formula (11).
2. For the first greatest relative distance of the input image and the first greatest relative distance of the reference image we calculate percentage deviation. This calculated percentage is stored in order to calculate average arithmetic percentage deviation later.
3. Step 1 is repeated until all relative distances in the input image or the reference image are evaluated.
4. Average percentage deviation is calculated as the average value of all average deviations as calculated in step 1.

From the above algorithm it is clear that the number of mathematical operations will be higher, which will result into higher demands for computer time. To reduce the number of operations the simplified algorithm can be used:

1. Both the input and reference image descriptors are sorted by values of relative distances.
2. For the longest relative distance of the input image and longest relative distance of the reference image percentage error is calculated and saved for later calculation of the arithmetic average percentage deviation. The percentage deviations are computed sequentially for each pair of relative distances with the same order position (position in input image and position in reference image) until all pairs of relative distances are taken into account.
3. The average percentage error as the average of percentage deviations calculated in step 1 and 2.

The advantage of the simplified algorithm is that it is less time demanding. The result of the algorithm is the value of the average percentage deviation, which expresses the degree of similarity between the input image and the reference image.

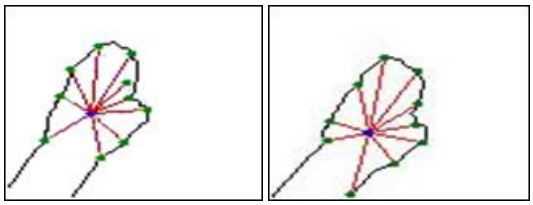

Fig. 24. The contour of the moving hand stored in the database of already learned movements (left) and the contour of the moving hand detected as a similar contour in the input image (right).

Given the fact that we identified the movement and not only a similarity of a single input image with the reference image, a measure of the motion similarity of should be calculated. For expressing degree of similarity of motion, it is necessary to express the percentage deviation between the input motion and the reference motion. The algorithm is searching for such a sequence of consecutive images of the reference motion that will produce the maximum similarity of the input motion with the reference motion. It is therefore an optimization problem, which may resemble the so-called "traveling salesman problem", in which a path is looking for which secure reaching the extreme of optimization function when moving from one place to another. In the problem in hand the task is to find a sequence of reference images, which would secure maximum agreement between the input and reference motion.

7. CONCLUSION

The rotation, zoom invariant motion descriptors based on the curves of the longest edge of a moving object and calculated in real time are inevitable data for better real time identification and/or classification of the moving object. The presented approach can be easily implemented. Its computational complexity is low due to the fact that the memory is exclusively used for motion and edge detection. Besides, the way of excluding the redundant and useless information while useful information is converted into the appropriate data ranges were described. Due to that the only a small portion of memory is required for implementation of the "in memory database" with fast response. Note that when a motion comprising tens or even hundreds images is to be processed, a sophisticated methods were traditionally implemented to identify the similar motions in knowledge base. Finally the approach to implementation of a multilevel classifier exhibiting better performance than the one-level classifier was presented. It is able to evaluate image details even in the case of low similarity.

ACKNOWLEDGMENT

The support of the grant VEGA 1/0177/11 is dully acknowledged.

REFERENCES

HANA. http:// www.saphana.com

Liu Ch., Huang X. and Wang M. (2012). Target Tracking for Visual Servoing Systems Based on an Adaptive Kalman Filter, *International Journal of Advanced Robotic Systems*, Volume 9,

Luo Y., Duraiswami R. (2008). Canny Edge Detection on NVIDIA CUDA, *Computer Vision and Pattern Recognition Workshops*, 1-8. Anchorage, USA.

Marji M., Siy P. (2003). A new algorithm for dominant points detection and polygonization of digital curves, *Pattern Recognition* Volume 36, Issue 10, 2239-2251, ISSN: 0031-3203.

OpenCV. http://opencv.org

Teixeira L., Celes W., Gattass M. (2008). Accelerated Corner-Detector Algorithms, *British Machine Vision Conference*. Leeds, UK.

Topal C., Akinlar C., Genc Y. (2010). Edge Drawing: A Heuristic Approach for Robust Real-Time Edge

Detection, *International Conference on Pattern Recognition*, 2424-2427. Istanbul, Turkey.

Wang Ch. W. (2006). Real Time Sobel Square Edge Detector for Night Vision, *Springer*, 404-413. Berlin, DE.

Wasza J., Bauer S., Hornegger J. (2011). Real-time Preprocessing for Dense 3-D Range Imaging on the GPU: Defect Interpolation, Bilateral Temporal Averaging and Guided Filtering, *Computer Vision Workshops*, 1221 – 1227. Barcelona, ES.