Recognition of Hand Postures Based on a Point Cloud Descriptor and a Feature of Extended Fingers

Submitted: 23rd December 2015; accepted 3rd February 2016

Dawid Warchoł, Marian Wysocki

DOI: 10.14313/JAMRIS_1-2016/7

Abstract:
Our work involves hand posture recognition based on 3D data acquired by the KinectTM sensor in the form of point clouds. We combine a descriptor built on the basis of the Viewpoint Feature Histogram (VFH) with additional feature describing the number of extended fingers. First, we extract a region corresponding to the hand and then a histogram of the edge distances from the palm center is built. Based on quantized version of the histogram we calculate the number of extended fingers. This information is used as a first feature describing the hand which, together with VFH-based features, form the feature vector. Before calculating VFH we rotate the hand making our method invariant to hand rotations around the axis perpendicular to the camera lens. Finally, we apply nearest neighbor technique for the posture classification. We present results of cross-validation tests performed on a representative dataset consisting of 10 different postures, each shown 10 times by 10 subjects. The comparison of recognition rate and mean computation time with other works performed on this dataset confirms the usefulness of our approach.

Keywords: *hand posture recognition, depth cameras, Kinect, point cloud, Viewpoint Feature Histogram*

1. Introduction

Nowadays, people tend to use gesture-based computer interfaces which are present in mobile device applications, computer games, control systems used in television sets, etc. Gesture recognition is therefore one of the most important problems of human-computer or human-robot interaction. Currently available gesture recognition methods are relatively primitive, compared to vision of mammals [4], and offer satisfactory reliability only in controlled laboratory environment. This makes vision based recognition algorithms an important and challenging research area. This paper presents an approach to recognize hand postures, which are often referred to as static hand gestures, based on three-dimensional depth data in a form of point clouds. 3D representation of visual information is more natural to humans than images obtained from standard 2D cameras. It is because we have two eyes that enable us to estimate our distances from particular objects.

Recently introduced Microsoft's Kinect™ sensor, used mainly to control computer games, became a low-cost device that can be used to depth data acquisition. Another worth mentioning 3D imaging devices are time-of-flight cameras which are now becoming more and more affordable to the mass market. Growing popularity of these devices caused researchers' interest in hand gesture recognition using depth cameras. However, most often the depth data is used only for hand segmentation or as an auxiliary information included in feature vectors of classified objects [11], [12], [19], [21]. Depth data combined with color data is used by [20] to classify postures based on Average Neighborhood Margin Maximization Transformation approximated by Haarlets (i.e., Haar wavelet-like features). Another approach to hand posture recognition based on depth data is presented by Keskin *et al*. [7] where Kinect™ data is used to obtain hand skeletons by the Mean Shift Local Mode Finding algorithm. Then a skeleton fitting method is used with the random decision forests to classify depth pixels into hand parts. Skeletal data is also utilized by Jiang *et al*. [3] along with depth data by calculating histograms of points' distances from the hand joints. The mean, variance and symmetry of the histograms are used as features to recognize hand gestures.

Two of the most interesting methods of depth-based segmentation data are presented by Oprisescu *et al*. [12] and Dominio *et al*. [2]. In the first of these works the region growing algorithm is used. The growing stops at the boundaries of the region detected using three thresholds: (i) the depth distance of the current point from the seed (the point nearest the camera), (ii) the depth distance of the current point from its neighborhood, and (iii) the luminance threshold based on the intensity time-of-flight image. In [2] palm is separated from fingers and forearm using the largest circle (or ellipse) that can be fitted in the palm region. A center of palm is found by performing Gaussian blur on the binarized arm image and searching for the point with sufficiently large value that is close enough to the point nearest the camera. Proposed by us segmentation method, described in Subsection 3.1, is based on this approach. In the discussed work depth information is also used in the extraction of the classified objects' features. The combined descriptor includes the following feature sets: (i) the distances of the fingertips from the palm center, (ii) the distances of the fingertips from a plane fitted on the palm samples, (iii) the curvature of the contour of the hand region, and (iv) the shape of the palm region. The authors' another work [10] includes the addition of features extracted from the Leap Mo-

tion data. This device provides 3D information as a set of relevant hand points and some hand pose features. The Leap Motion-based descriptor contains three features: (i) the position of the fingertips, (ii) the palm center, and (iii) the hand orientation.

Viewpoint feature histogram (VFH) has been introduced by Rusu *et al*. [16]. It is a descriptor of 3D objects in a point cloud form. Particular VFH components describe geometry and viewpoint of the surface robustly to large surface noise and even missing depth information. In [5] an approach to recognition of dynamic hand gestures based on VFH and time-of-flight data has been described. Our later work includes the recognition of dynamic hand gestures as well as hand postures using both time-of-flight and Kinect™ data [6]. In these works a modification of VFH calculation has been proposed, which consists in dividing the observed scene into smaller cuboidal cells and calculating VFH for each of them. This method increases distinctiveness of the descriptor, especially for objects with subtle differences in shape, which has been proved by experiments resulting in significantly higher recognition rates for a divided scene. The approach is not fully rotation-invariant since while rotating the cloud some of its parts move between neighboring cells. Therefore, a transformation of the hand area should be applied before its division into cells and VFH calculation. We propose a fast way of hand rotation based on the most protruding hand point location.

The feature vector contains one additional feature encoding the number of extended fingers (including thumb) as a discrete value. A different approach to gesture recognition using the number of extended fingers is presented in [9]. The fingers are counted, identified and then the algorithm tracks the fingertips while they arrange to form a posture. In our method, the finger information is not crucial and is used only for discriminatory purposes that can be achieved by properly adjusting the importance of this feature with respect to other features.

The contributions of this paper are: (i) introducing the extended fingers feature, (ii) proposing some modifications to the hand segmentation and rotation methods that result in faster and easier to implement recognition algorithm, (iii) comparing the proposed algorithm (in terms of recognition rate and time) with the approaches from two significant and recent works.

The paper is organized as follows: Section 2 discusses the VFH descriptor; Section 3 describes every step of our proposed posture recognition system; In Section 4 we present the experimental results; Section 5 contains the conclusions and the plans for the future work related to this subject.

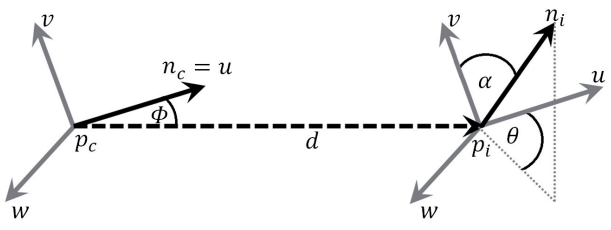


Fig. 1. Values of the surface shape component of the VFH

2. Viewpoint Feature Histogram

VFH is the global descriptor of a point cloud – a data structure representing a multidimensional set of points in a clockwise coordinate system [17]. The system's *x*-axis is horizontal and is directed to the left, the *y*-axis runs vertically and faces up, the *z*-axis coincides with the optical axis of the camera and is turned towards the observed objects. VFH consists of two components: a surface shape component and a viewpoint direction component. They describe geometry and viewpoint of surface created by clouds. The descriptor is able to detect subtle variations in the geometry of objects even for untextured surface, which has been shown experimentally [16].

The first component consists of values θ, $\cos(\alpha)$, $\cos(\Phi)$ and d measured between the gravity center p_c and every point p_i belonging to the cloud (see Fig. 1). n_c is the vector with initial point at p_c with coordinates equal to the average of all surface normals. n_i is the surface normal estimated at point p_i. The angles θ and α can be described as the yaw and pitch angles between two vectors while d denotes the Euclidean distance between p_i and p_c. The vectors and angles shown in Fig. 1 are defined as follows:

$$u = n_c \tag{1}$$

$$v = \frac{p_i - p_c}{d} \times u \tag{2}$$

$$w = u \times v \tag{3}$$

$$\cos(\alpha) = v \cdot n_i \tag{4}$$

$$\cos(\Phi) = u \cdot \frac{p_i - p_c}{d} \tag{5}$$

$$\theta = arctg\left(\frac{w \cdot n_i}{u \cdot n_i}\right) \tag{6}$$

where dot denotes the scalar product and cross denotes the vector product. The viewpoint component consists of a histogram of the angles that the viewpoint direction makes with each normal. The method of VFH calculation has one parameter *nn* denoting the number of points belonging to local neighborhood used to estimate the surface normals. Default histograms consist of 45 bins for each feature of the surface shape component and 128 for the viewpoint component (308 bins in total). Computational complexity of VFH is quadratic with respect to the number of cloud points. The more detailed descriptions of VFH calculation are presented in [15], [18] (PFH and FPFH descriptors) and [16]. In our experiments we process point clouds and calculate VFH using PCL library described in [17].

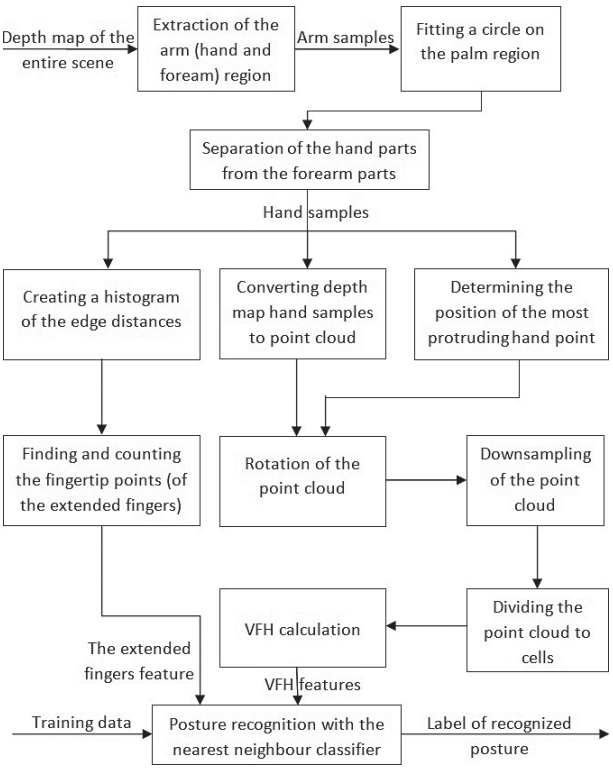


Fig. 2. ***Architecture of the proposed posture recognition system. Blocks represent functions; labels, next to arrows, represent input/output data***

3. Posture Recognition System

Proposed by us posture recognition system is shown as a diagram in Fig. 2. It consists of the following steps: hand segmentation, conversion of the depth map to the point cloud, rotation and downsampling of the point cloud, extraction of the features, and classification.

Our method imposes the following requirements to the person showing postures: (i) the hand should be the object closest to the camera, (ii) the hand should be situated at some distance, greater than a specified parameter T_{depth}, from the other body parts, (iii) at least a small part of the forearm should be visible. Note that all these requirements are satisfied almost every time if the postures are shown in a natural way and if there are no visible unwanted objects situated closer to the camera than the user's hand. The requirement (i) results from the fact that we exploit depth information only.

3.1. Hand Segmentation

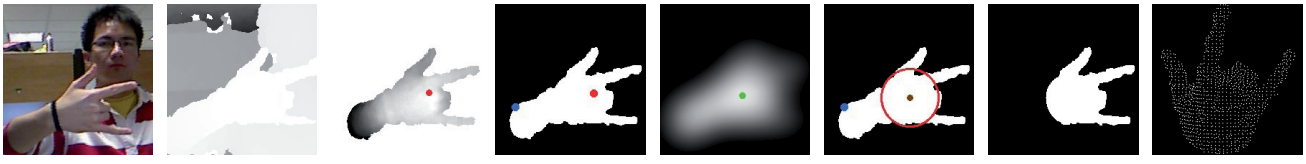
Proposed by us method of recognition begins with the segmentation of the hand. It is based on the approach described in [2] with some modifications introduced in order to make the algorithm faster and simplify it.

The first step of the hand segmentation is the extraction of the arm region (hand with forearm) from the depth map *DM* (see Fig. 3b) which is our input data. To this end, the pixel with the least depth value (closest to the camera) is localized and marked as *na* and its depth in [m] is denoted as d_{na}. Then all the pixels with the depth value greater than $d_{na} + T_{depth}$ are rejected by setting them to 0. As a result we obtain depth map representing arm that is shown in Fig. 3c. In our experiments we set T_{depth} to 0.1 [m] according to the empirical observations and taking the example from [2]. Subsequently, we threshold *DA* in order to obtain a binary image *BA* (see Fig. 3d):

$$BA_i = \begin{cases} 1 & if\ DA_i^z > 1 \\ 0 & otherwise \end{cases} \tag{7}$$

where DA_i^z is the depth value of *i*-th *DA* pixel and BA_i is the *i*-th *BA* pixel. Note that if T_{depth} value is too large, *BA* may contain some torso or hand pixels adjacent to the hand parts. However, if the value is too small, there may be no forearm parts in *BA*. Both cases may result in improper hand segmentation. To remove defects resulting from measurement errors of the imaging device, we propose to perform morphological operations on *BA* preparing it to the next step of the segmentation process. First, the jagged edges are smoothed using opening and closing with a 3×3 structuring element. Then, small gaps within the arm region, that are not larger than the rectangular structuring element of size 35×35 pixels, are filled. This size turned out sufficient for the data acquired by Kinect™ in good lighting conditions.

The next step of the hand segmentation is the selection of the central hand point. For this purpose, we apply the Gaussian filter on *BA* with the kernel of a large size depending on the depth value of *na*: $fe_x = fe_y = round_o(100/d_{na})$ pixels, where $round_o$ is a function rounding the value to the nearest odd integer. With such calculated kernel size the hand is wholly embraced by the kernel. The resulting grayscale image has a global maximum (the brightest point) C_o at the object's center (see Fig. 3e). We must, however,



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 3. Segmentation and rotation of the hand: (a) Color image (not used in our method); (b) Depth map DM; (c) Depth map with the arm region only (na is marked as a red dot); (d) Binary arm image BA (fa is marked as a blue dot); (e) Output of the Gaussian filter on BA (C_h is marked as a green dot); (f) BA with circle C_c (C_f is marked as a brown dot); (g) Binary hand image BH; (h) Rotated and downsampled point cloud

ensure that C_o belongs to the hand (not the wrist or the forearm region). Therefore, we create a set M containing every local maximum not smaller than 85% of the C_o value. Let us denote by $D_{m,na}$ the Euclidean 2D distance (on the xy plane) from the maximum point m to na and by m_v the value (brightness level) of m. As a central hand point C_h we select a maximum from the set M for which the difference $D_{m,na} - m_v$ is minimal. Thus, the hand center is not too far from the point of the least depth na, and it has similar (if not the same) brightness as the global maximum C_o. In [2] the point C_h is selected by minimization of $D_{m,na}$, without using m_v. Our approach to selection of C_h turned out to be more precise which was confirmed by the experiments.

After the calculation of the central hand point, the palm region of the hand is roughly estimated by the largest circle that can be fitted on it. The circle with the center C_h is iteratively enlarged, starting from the radius equal to one pixel, until the hand area inside it, denoted as hc_{area}, is less than 95% of the circle area c_{area}. Then the circle is moved to a neighboring position maximizing the hc_{area}, and finally the radius is increased once more. After this step, we obtain the updated central hand point C_f with coordinates C_{fx},C_{fy} and the central palm circle C_c with radius R_{Cc}.

The last segmentation stage is the rejection of the pixels corresponding to the forearm as well as the eventual unwanted objects. We begin with the pixels not belonging to the arm. Such fragments might not be filtered during the first segmentation step because of not using the color-based skin detection in contrast to [2]. They are usually not adjacent to the hand pixels, and we exploit this fact by applying a region growing algorithm with the seed point C_f. The output of this operation is the arm region without the unwanted fragments that were previously isolated from the arm pixels in the *BA* image. Note that if the algorithm requirements (i) and (ii), mentioned at the beginning of this section, are satisfied, we are sure that, at this point, *BA* contains only the pixels belonging to the arm region. The corresponding pixels are also rejected in the depth map *DA*.

The main difference in relation to the original segmentation method is the separation of the forearm from the hand. Dominio *et al*. [2] propose to first rotate the arm in a way that its main axis, extracted using PCA (Principal Component Analysis), coincides with y-axis and faces up. Then, every arm pixel with x coordinate smaller than C_{fy}-R_{Cc} is considered as forearm pixel and therefore discarded. The direction of the main axis computed in this way is not very precise because it depends on the position not only of the hand, but also of the forearm (which can be different). The approach proposed by us consists in discarding the forearm region first and performing the rotation of the hand later (for the feature extraction purposes), without the forearm pixels. The hand rotation is described in Subsection 3.3. For the separation of the forearm pixels we use region growing method, as for the previous operation. In this case, as a seed point we use the pixel with the largest depth value (furthest to the camera) marked as *fa*. In almost every case this pixel belongs to the forearm. The region growing is applied for *BA* image with drawn circle C_c (see Fig. 3f). The region boundaries are the edges of the forearm and the lower part of C_c. The designated pixels are considered to belong to the forearm region and are therefore rejected forming a binary hand image *BH*. In some rare cases, which may be the consequence of showing the posture while not following the requirement (ii), *fa* may belong to a hand region situated outside the circle C_c. Such a small part is then rejected and the forearm region remains. To give the algorithm another chance to remove the proper pixels, the region growing is repeated when the number of removed points in the last operation is smaller than 5% of the whole arm area in *BA*. As a new seed point, the arm pixel closest to *fa* is selected provided it lies outside C_c. An example of a segmented hand is presented in Fig. 3g.

3.2. Conversion to the Point Cloud

In order to calculate VFH, each depth map has to be converted into the point cloud format. For this purpose, we define the set H of hand pixels belonging to binary image *BH*: $H = \{x \in BH | \ x = 1\}$. The conversion is applied for each *DA* pixel if H contains a pixel of the same coordinates. Therefore, the created point cloud consists only of points belonging to the hand, without forearm and isolated objects. The coordinates of cloud points: PC_i^x, PC_i^y, and PC_i^z were set with respect to the *DA* pixels' depth value DA_i^z based on the perspective projection equations and Kinect™ camera's parameters:

$$PC_i^x = \frac{(DA_i^z + fl) * \left(\frac{DA_{width}}{2} - DA_i^x - 1\right) * ps^x}{fl} \quad (8)$$

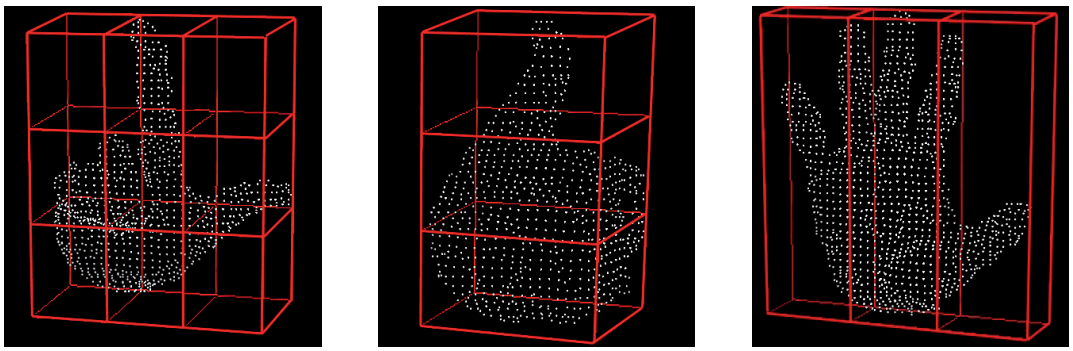
$$PC_i^y = \frac{(DA_i^z + fl) * \left(\frac{DA_{height}}{2} - DA_i^y - 1\right) * ps^y}{fl} \quad (9)$$

$$PC_i^z = DA_i^z \quad (10)$$

where DA_{width} is the number of depth map columns; DA_{height} is the number of depth map rows; fl is the Kinect™ infrared camera's focal length; ps^x and ps^y are the pixel dimensions, width and height, respectively. The values of Kinect™ camera's parameters were taken from *http://kinectexplorer.blogspot.com* and set as follows: fl = 4.73 mm, $ps^x = ps^y$ = 0.0078 mm. After the conversion clouds observed from different angles, look realistic compared to corresponding real life objects.

3.3. Rotation and Downsampling of the Point Cloud

Since the approach of calculating VFH for cells, which is explained in Subsection 3.4, is not invariant to hand rotations around the z-axis, the point cloud has to be properly rotated before its division into cells. To this end, we calculate 3D distance (exploiting depth information) between the central point C_f and each hand pixel H_i. Then we choose the point for



(a) (b) (c)

Fig. 4. Point clouds of postures and their divided bounding boxes (marked by red edges): (a) nine cells; (b) three horizontal cells; (c) three vertical cells

which the distance is maximal and denote it as *Mf*. This point indicates the position of the most protruding hand point which is usually the longest finger's tip when at least one finger is extended in the posture. The point cloud is then rotated in a way that the vector $Mf - C_f$ has the same direction as the *y*-axis.

The obtained cloud is redundantly dense. We do not need such a large number of points to compute a representative VFH. For this reason the cloud is downsampled which reduces the number of points and, therefore, speeds up the process of histograms calculation. This operation is performed by the so-called *voxel grid* filter which creates a 3D voxel grid over the input point cloud data. Every point situated within each voxel (i.e., 3D cuboid) is approximated by its centroid. The voxel dimensions $V_x \times V_y \times V_z$ are the parameters of the filter. We decided to use cubic voxels and set the dimensions as follows: $V_x \times V_y \times V_z$ =0.00439 m, which in our previous work [6] turned out to be the optimal value in terms of postures recognition rate. An example of a rotated and downsampled point cloud is shown in Fig 3h.

3.4. Extraction of the Features

For the classification purposes we collect the feature set based on the VFH descriptor and one additional feature of extended fingers, denoted as F_{vfh} and F_{ef}, respectively.

In order to calculate VFH, we must define the bounding box, i.e., the cuboidal area around the point cloud. Its walls are determined by the points situated furthest in each main direction of the coordinate system: Bx_{min}, Bx_{max}, By_{min}, By_{max}, Bz_{min}, Bz_{max}. The bounding box size is thus matched to the cloud size embracing it entirely with no space between each furthest point and the box wall. To increase the distinctiveness of the descriptor, we divide bounding boxes into several cells of equal sizes and VFH is calculated for each of them. Such a modification is inspired by the method of dividing image into smaller blocks in order to calculate histogram of oriented gradients (HOG) descriptor [1]. It seems reasonable to try to divide areas of interest into different numbers of small regions. Koszowski [8] showed that the number of blocks significantly affects the recognition rate in the case of the HOG-based hand posture recognition. Therefore, our experiments involve the division of point cloud area into: (i) nine cells, (ii) three horizontal cells, and (iii) three vertical cells (see Fig. 4).

To avoid problems with high dimensionality, F_{vfh} does not include full histograms. Instead, VFH calculated for every descriptor value is represented by its mean and standard deviation. We use the two VFH values: $|d|$ and $\cos(\Phi)$ since our previous work [6] indicates that this combination is the best in terms of the hand posture recognition and adding new values or replacing some of them does not yield better results. The experiments undertaken in the mentioned work and other previous works also led us to the conclusion that VFH calculated for the entire point cloud (without the division into cells) was significantly less distinctive, and in this case the postures recognition most often resulted in misclassification, even for the seemingly dissimilar hand shapes. For these reasons we decided to omit the experiments with different VFH values as well as with the VFH calculated for the entire cloud. The F_{vfh} feature set has a cardinality of $N_c \cdot N_v \cdot N_{hs}$, where N_c is the number of cells, N_v is the number of VFH values, and N_{hs} is the number of histograms representative values (in our case mean and standard deviation). Every feature is normalized to the range [0-1], with maximum and minimum values determined on the basis of the training data, and the whole set is added to the feature vector *F*.

The additional feature is the number of extended fingers (including thumb). To extract it, a histogram of the edge distances from the central hand point C_f is calculated. For each hand pixel H_i we determine its 2D distance from the central point $d_{Hi} = ||H_i - C_f||$ as well as the angle γ_{Hi} between the vector $H_i - C_f$ and the vector perpendicular to the *y*-axis with the opposite direction. The calculated set of angles is then quantized with the uniform quantization step $\Delta=6°$ and assigned to the set γ'. All H_i pixels with the same discrete value γ'_{Hi} are grouped together dividing H as follows: $H = \{C_1, C_2, \ldots, C_{60}\}$, where C_j denotes the angular range, i.e., the group of H_i pixels for which $\gamma'_{Hi} = j$, and $j \in \{1,2,\ldots,60\}$. For each group C_j the maximum d_{Hi} value is chosen and added to the histogram *L*:

$$L(j) = \max_i \left(d^j_{Hi} \in C_j \right) \tag{11}$$

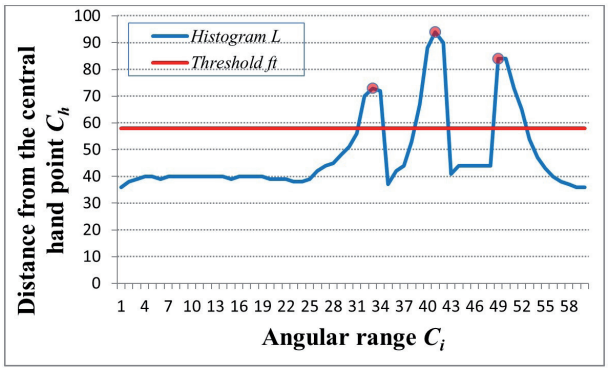


Fig. 5. ***Histogram of the edge distances from the hand center. Local maxima corresponding to the tips of the extended fingers are marked as red dots***

Thus, L contains the distances of a subset of the hand edge pixels.

Then we find every local maximum of L that is not smaller than the threshold ft equal to 145% of the C_c radius. These points represent the tips of the extended fingers. In our method, a maximum is a point that is greater than its left neighbor and not smaller than its right neighbor (the first and the last point are also considered as neighbors). The threshold ft ratio is set to 145% after some experimentation, so that the algorithm can properly identify extended fingers (even relatively short thumbs). Fig. 5. contains the histogram calculated for the posture presented in Fig. 3. The obtained number of extended fingers is normalized to the range [0–1] and added to the feature vector F.

3.5. Classification

For the classification of postures the k-nearest neighbors technique with k=1 was utilized. Each hand posture was represented by the feature vector $F = [F_{vfh}, F_{ef}]$. We noticed that F_{ef} feature was too important for the classifier to treat it like F_{vfh} features whose number is usually much greater. Thus, while calculating the distance of classified object to each pattern from the training set, we multiply the F_{ef} by a weight i_{ef} to increase its importance in classification. The distance $D_{o,p}$ from the object o to pattern p is therefore calculated in the following way:

$$D_{o,p} = \sum_{i=1}^{N_{vfh}} \left(F^i_{vfh,o} - F^i_{vfh,p}\right)^2 + \left(i_{ef} \cdot (F_{ef,o} - F_{ef,p})\right)^2 \quad (12)$$

where N_{vfh} is the number of F_{vfh} features. The best i_{ef} value, in terms of recognition rate, depends on N_{vfh}. We thus performed cross-validation experiments with different number of cells and VFH values which resulted in changing N_{vfh}. This let us roughly estimate,

Table 1. Mean accuracies obtained by 5-folds cross validation tests and mean running times of our approach compared to authors' method from [2]. The tests were performed on the computer (i) - PC with Intel® Core™ Quad, 2.4 GHz CPU

Method	Recognition rate [%]	Mean running time [ms]
F_{vfh}: 9 cells + F_{ef}	95.4	57
F_{vfh}: 3 horizontal cells + F_{ef}	98	
F_{vfh}: 3 vertical cells + F_{ef}	91.1	
F_{vfh}: 9 cells (without F_{ef})	82.6	56
F_{vfh}: 3 horizontal cells (without F_{ef})	88.9	
F_{vfh}: 3 vertical cells (without F_{ef})	83.4	
Four combined depth features (Dist. + curv. + elev. + area) [2]	99	114
Three combined depth features (Dist. + curv. + area) [2]	99	114
Two combined depth features (Dist. + curv.) [2]	98.5	104

through logarithmic regression, the following dependency allowing to calculate i_{ef}:

$$i_{ef} = 0.485 \cdot \ln\left(N_{vfh}\right) + 1.455 \quad (13)$$

We also observed that larger i_{ef} values did not cause much worse recognition results. We may then assume that i_{ef} should be greater than or equal to the value determined by equation (13). Later experiments were performed with i_{ef} estimated as above.

As one can observe, $D_{o,p}$ is calculated based on the Euclidean distance. We also examined the city-block distance as well as k-nearest neighbors classifier with different values of k but the results were not better.

4. Experiments

Our method was evaluated on a dataset first introduced in [14] and used in later works: [13], and [2]. This is a set of 10 postures from which 8 is a subset of American Sign Language finger alpha-

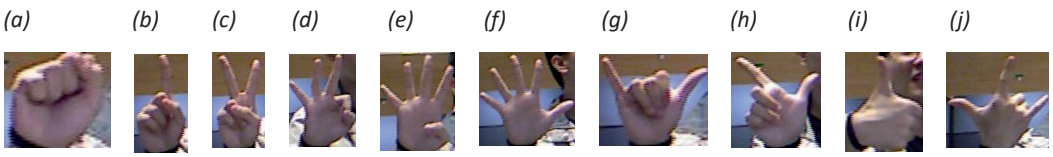


Fig. 6. Postures included in the dataset used for the evaluation of our recognition method

Table 2. Mean accuracies obtained by template matching tests and mean running times of our approach compared to authors' and classic methods provided by Ren et al. [13]. The tests were performed on the computer (ii) - PC with Intel® Core™ Quad, 2.66 GHz CPU

Method	Recognition rate [%]	Mean running time [ms]
F_{vfh}: 9 cells + F_{ef}	71.3	46
F_{vfh}: 3 horizontal cells + F_{ef}	93.8	
F_{vfh}: 3 vertical cells + F_{ef}	88.3	
F_{vfh}: 9 cells (without F_{ef})	49.8	45
F_{vfh}: 3 horizontal cells (without F_{ef})	77.9	
F_{vfh}: 3 vertical cells (without F_{ef})	69	
Thresholding decomposition + FEMD – authors' method [13]	93.9	4 001
Near-convex decomposition + FEMD – authors' method [13]	93.2	750
Shape context without bending cost – classic method [13]	83.2	12 364
Shape context with bending cost – classic method [13]	79.1	26 777
Skeleton matching – classic method [13]	78.6	2 445

bet. Hands are shown on a cluttered background, in a various orientation around the *z*-axis differing up to about 135 degrees. They also differ in scale and articulation. The postures were shown 10 times by 10 subjects (1000 executions in total). Fig. 6 contains an example of each posture from the dataset.

4.1. Evaluation of the Segmentation Method

The segmentation method was evaluated first. We visually analyzed generated binary hand images *BH* for every posture file from the database. Only 5 (out of 1000) images were improperly segmented. In each case some hand parts were deleted and in four cases the forearm parts were present. These errors occur when the furthest arm point *fa* does not belong to the forearm. It is hard to determine whether this is the consequence of not following the algorithm requirement (i) or (ii) by the person showing postures, or this results from a measurement error of the camera.

4.2. Evaluation of the Rotation Method

The next step was the evaluation of the rotation method. The visual analysis of the rotated clouds showed that the point *Mf* did not always correspond to the actual longest extended finger because of little differences between the lengths of some fingers and the camera measurement errors. We checked all 100 realisations of the posture G6. *Mf* corresponded to the middle finger in 93 postures, to the ring finger in 4 postures, and to the index finger – in 3 postures. We also checked 100 realisations of the posture G7. *Mf* corresponded to the little finger in 83 postures, to the thumb – in 15 postures, and 2 postures were improperly segmented (the forearm pixels were not rejected). These inaccuracies do not become a significant problem if we use sufficiently large training dataset and the *k*-nearest neighbors classifier. In such cases, for some posture classes we have 2 or 3 variants of rotated postures according to the finger to which *Mf* corresponds. In most cases the classifier is able to properly find the nearest neighbour among each variant leading to proper classification, which can be seen on the recognition rates presented later in this section.

4.3. Evaluation of the Extended Fingers Counting

The approach to count the extended fingers was also evaluated. 97.6% of all the postures have their fingers properly counted. Analyzing the failures we observed that 62.5% of them were related to the posture G1 (clenched fist). In most cases the thumb does not appear on the background of the fingers but on the left side of the fist. It is also not clenched at all or bent only a little. Sometimes even the forefinger is not entirely clenched which appears unnatural even for a human. Increasing the threshold *ft* prevents such mistakes. However, it also causes the situations where relatively short extended thumbs are not counted, es-

Tab. 3. Confusion matrix for our algorithm with the three horizontal cells case and 5-fold cross-validation tests. The maximum possible value in every cell is 100. For clarity, the cells with value 0 are shown as empty

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	97	2	1							
G2		99						1		
G3			100							
G4				99	1					
G5				1	97	2				
G6					2	98				
G7							94	6		
G8							2	98		
G9							1		99	
G10				1						99

pecially when they are too close to the palm region. The remaining 37.5% of failures result from the segmentation errors or from arranging the extended fingers too close to each other so they appear on the image as nearly adjacent. In such cases an error would occur if the tips of these fingers belong to the same angular range C_j of the quantized histogram L.

In 70.8% of all the miscounts, the obtained number of fingers differs by 1 from the actual number. For each remaining failure, the difference is equal to 2 and is related to the posture G1 (not entirely clenched forefinger and thumb).

4.4. Comparison of Recognition Rates and Mean Running Times

The main evaluation of our method consists in the comparison of recognition rates and mean running times with the results obtained in [2] and [13] for the same dataset. The comparison is shown in Table 1 and Table 2. The results from Dominio *et al*. [2] were obtained for different combinations of their proposed depth features. In addition to their own method, *Ren et al*. [13] provide results for three classic shape recognition algorithms: (i) shape context with bending cost, (ii) shape context without bending cost, and (iii) skeleton matching. Our comparison also includes those scores. Note that Ren *et al*. [13] exploit a black belt, worn by all the people showing postures, in the hand segmentation. Our method does not require to wear such markers and therefore the belt information is not used.

The recognition rates shown in Table 1 were measured in 5-fold cross-validation tests. We divided the dataset to five subsets: one used as a training set and the remaining four as a test set. The division was performed in a way that subjects from the test set were not present in the training set. All the gestures of the first two subjects created the first test set, the gestures of the next two subjects created the second one, and so on. Thus, the ability of the recognition system to generalize over unseen subjects was verified. Dominio *et al*. [2] performed exactly the same cross-validation tests. The i_{ef} value was set according to equation (13). Our best score, obtained for the three horizontal cells, is 98%. It is slightly lower than for the algorithm described in [2] where the best results, obtained for three or four combined depth feature sets, are 99%.

In [13] the recognition rates are measured using template matching method where the first gesture realization of each class is used for creating its template. Therefore, the training set consists of only ten realizations. For comparison purposes, we decided to additionally perform such tests despite the fact that our recognition system was designed to be trained on much larger data. In our tests first gesture realizations from person number 5 were used as templates since they turned out to be the most representative. Using such a small training set does not provide representative F_{vfh} features for each class. Therefore, we multiplied by 1.5 the i_{ef} value, calculated according to equation (13), in order to put more emphasis on the extended fingers feature. Our best score, obtained for the three horizontal cells, outperforms every classic algorithm and the authors' method near-convex decomposition + FEMD and is lower by 0.1% in the case of thresholding decomposition + FEMD.

To prove the usefulness of the extended fingers feature, we also present the results obtained without it. For the three horizontal cells case the recognition rate is less by 9.1% according to the cross validation tests and less by 18.7% according to the template matching tests, compared to the case when the additional feature is applied. As one can see, the best obtained recognition rate is slightly higher than the correctness of counting fingers. It is due to the fact that if the number of incorrectly detected fingertips does not much differ from the actual number (what has been observed in every case during our experiments), the classifier is still able to correctly guess the posture class.

We can assume that the results for the horizontal cells are significantly higher than for the vertical cells because in the first case the method is less dependent on the hand rotations around the y-axis while it depends more on the hand inclinations around the x-axis. Analyzing the postures from the chosen dataset, we can observe that the rotation around the y-axis is a little more variable than the second one. That explains the better results obtained for the three horizontal cells.

Table 3 is the confusion matrix for our algorithm and the three horizontal cells case. The worst result has been obtained for the posture G7 which was confused with G8 six times. This can be explained by the high similarity between them as well as the same number of extended fingers. The interesting fact is that the posture G3 has been recognized correctly in 100% cases, despite that there are three postures with two extended fingers in the dataset. It shows that the VFH descriptor is able to very well differentiate G3 from G7 and G8 for which the F_{ef} feature has the same value. In this case, the upper cell of the point cloud bounding box seems to be the most discriminant because, after the clouds rotation, for G3 it includes two extended fingers while for G7 and G8 only one extended finger is present in this cell and another one is located in the middle or the lower cell.

In the terms of a recognition program's mean computation time, our method turned out to be very fast compared to others. The time refers to the entire recognition process of a single posture. Obviously, this score depends on the computer, on which the program is running. We performed experiments on two computers: (i) PC with Intel® Core™ Quad, 2.4 GHz CPU (released in 2007) and (ii) PC with Intel® Core™ Quad, 2.66 GHz CPU (released in 2009). Ren *et al*. [13] used a PC with the same CPU as in computer (ii). The mean running time differences between our method and the methods presented in this paper are huge. Our recognition algorithm is about 16 times faster than thresholding decomposition + FEMD and about 87 times faster than near-convex decomposition + FEMD. It is worth noting that the usage of much larger training sets, as in cross-validation tests, significantly improves the recognition rates. In authors' methods from [13] it will cause much greater running times because the calculation of Finger-Earth's Move Distance, that have to be done in each template

comparison, is time-consuming. For our method, the difference between processing times for test sets containing 10 depth maps and test sets containing 800 depth maps is below 0.5 ms.

Dominio *et al*. [2] performed tests on a computer with the same CPU as in computer (ii). In this case the mean running time of our method is almost twice lower. We also compared the segmentation and the rotation stages' mean running time. The original method takes about 75 ms while our modified version – 47 ms, which shows the advantage of our modifications. The computation time of the extended fingers feature extraction is very short (not greater than 1 ms on the computer (i) and (ii)).

Our recognition system is able to run in real-time at about 17 frames per second on the computer (i) and 21 frames per second on the computer (ii). We were able to achieve 35 frames per second while running the program on a modern laptop with Intel® Core™ i7, 2.5 GHz CPU. In order to achieve greater speed, it is possible to apply multithreading to the implementation of our algorithm since some tasks can be performed concurrently, e.g., creating a histogram of the edge distances, converting depth map hand samples to point cloud, and determining the position of the most protruding hand point (see Fig. 2).

5. Conclusions and Future Work

In this paper we propose the depth-based posture recognition method using the Viewpoint Feature Histogram and the extended fingers feature. The accuracy of our approach was tested on a challenging dataset with the usage of demanding cross-validation tests in which the classifier recognized postures shown by unseen subjects. The mean running time of the recognition program was also measured. The relatively good results confirm the usefulness of the proposed algorithm.

The recognition system proposed by us can be applied to control and interact with computers and robots. For example, the user is able to give specific commands to the robot showing his hand in a specific configuration. Although the recognition of postures by the system does not depend on rotation around *z*-axis, the algorithm provides an information about how much the hand is rotated. It may be used, e.g., for setting some application or system parameters in a way that recognized gesture corresponds to particular parameter and its rotation angle represents the parameter value.

The extended fingers feature turned out to be very helpful because of the low computation time of its extraction and the accuracy improvement caused by its application. This feature is obviously not autonomous because of its possible ambiguity (there may be many postures with particular number of extended fingers). However, it can significantly increase the distinctiveness of the descriptor if its importance parameter is properly chosen. It can be observed that the extended fingers feature and the VFH features complement each other because the first analyzes the hand contours while the second examines dependencies within the hand surface.

Our method fails to correctly count the number of fingers if they are adjacent to each other in a presented posture (i.e., the letter 'H' of American Sign Language finger alphabet). In such a case each group of adjacent fingers would be counted as one. However, obtaining the actual number of extended fingers is not an end in itself. The method's purpose is to search for sufficiently large local extrema in the shape of the hand posture and to exploit this information in the distinction of posture classes.

The future work in the subject could consist in altering the VFH descriptor. So far only the modification of its calculation was proposed, which proved to be very effective. Searching for other dependencies among the cloud points or the surface normals, that better describe the hand shape, is reasonable.

ACKNOWLEDGEMENTS

The authors thank dr Tomasz Kapuscinski and dr Mariusz Oszust from Rzeszow University of Technology for their valuable suggestions.

AUTHORS

Dawid Warchoł* - Rzeszów University of Technology, Department of Computer and Control Engineering, Faculty of Electrical and Computer Engineering, Wincentego Pola 2, Rzeszów, Poland, 35-959, tel.: (17)8651592.
E-mail: dawwar@prz.edu.pl.

Marian Wysocki - Rzeszów University of Technology, Department of Computer and Control Engineering, Faculty of Electrical and Computer Engineering, Wincentego Pola 2, Rzeszów, Poland, 35-959, tel.: (17)8651583.
E-mail: mwysocki@prz.edu.pl.

*Corresponding author

REFERENCES

[1] N. Dalal, B. Triggs, "Histograms of oriented gradients for human detection", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego: IEEE, 2005, 886–893. DOI: 10.1109/CVPR.2005.177.

[2] F. Dominio, M. Donadeo, P. Zanuttigh, "Combining multiple depth-based descriptors for hand gesture recognition", *Pattern Recognition Letters*, vol. 50, 2014, pp. 101–111. DOI: 10.1016/j.patrec.2013.10.010.

[3] F. Jiang, C. Wang, Y. Gao, S. Wu, D. Zhao "Discriminating features learning in hand gesture classification", *IET Computer Vision*, vol. 9, 2015, no. 5, 673–680. DOI: 10.1049/iet-cvi.2014.0426.

[4] P. Garg, N. Aggarwal, S. Sofat, "Vision based hand gesture recognition", *World Academy of Science, Engineering and Technology*, vol. 49, 2009, no. 1, 972–977.

[5] T. Kapuściński, M. Oszust, M. Wysocki, "Recognition of signed dynamic expressions observed by ToF camera", *Signal Processing: Algorithms,*

Architectures, Arrangements, and Applications (*SPA*), Poznań, 2013, 291–296.

[6] T. Kapuściński, M. Oszust, M. Wysocki, D. Warchoł, "Recognition of hand gestures observed by depth cameras", *International Journal of Advanced Robotic Systems*, 2015. DOI: 10.5772/60091.

[7] C. Keskin, F. Kirac, Y.E. Kara, L. Akarun, "Real time hand pose estimation using depth sensors". In: *IEEE International Conference on Computer Vision Workshops*, Barcelona, Spain, 2011, 1228–1234. DOI: 10.1109/ICCVW.2011.6130391.

[8] G. Koszowski, "Two-pass algorithm for hand pose gesture recognition". In: *XIX National Conference of Discrete Processes Autom*ation, Zakopane, Poland, 2014.

[9] Y. Li, "Hand gesture recognition using Kinect", *Software Engineering and Service Science* (ICSESS), Beijing, 2012, 196–199. DOI: 10.1109/ICSESS.2012.6269439.

[10] G. Marin, F. Dominio, P. Zanuttigh, "Hand gesture recognition with Leap Motion and Kinect devices". In: *IEEE International Conference on Image Processing* (ICIP), Paris, 2014, 1565-1569. DOI: 10.1109/ICIP.2014.7025313.

[11] J. Molina, A. Escudero-Viñolo, A. Signoriello, M. Pardàs, C. Ferràn, J. Bescós, F. Marqués, J. M. Martínez, "Real-time user independent hand gesture recognition from time-of-flight camera video using static and dynamic models", *Machine Vision and Applications*, vol. 24, 2013, no. 1, 187–204. DOI: 10.1007/s00138-011-0364-6.

[12] S. R. Oprisescu, "Automatic static hand gesture recognition using ToF cameras". In: *European Signal Processing Conference*, Bucharest, 2012, 2748–2751.

[13] Z. Ren, J. Junsong, Y. Meng, Z. Zhang, "Robust part-based hand gesture recognition using Kinect sensor", *IEEE Transactions on Multimedia*, vol. 15, 2013, no. 5, 1110–1120. DOI: 10.1109/TMM.2013.2246148.

[14] Z. Ren, J. Yuan, Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera". In: *19th ACM International Conference on Multimedia*, Scottsdale, 2011, 1093–1096.

[15] R. B. Rusu, N. Blodow, M. Beetz, "Fast point feature histograms (FPFH) for 3D registration". In: *IEEE Conference on Robotics and Automation*, Kobe, 2009, 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[16] R. B. Rusu, G. Bradski, R. Thibaux, "Fast 3D recognition and pose using the Viewpoint Feature Histogram". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), Taipei, 2010, 2155–2162. DOI: 10.1109/IROS.2010.5651280.

[17] R. B. Rusu, S. Cousins, "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference*, Shanghai, 2011, 1–4. DOI: 10.1109/ICRA.2011.5980567.

[18] R. Rusu, Z. C. Marton, N. Blodow, "Learning informative point classes for the acquisition of object model maps". In: *Conference on Control, Automation, Robotics and Vision*, Hanoi, 2008, 643-650. DOI: 10.1109/ICARCV.2008.4795593.

[19] D. Uebersax, J. Gall, M. Van den Bergh, "Real-time sign language letter and word recognition from depth data". In: *IEEE International Conference on Computer Vision Workshops* (ICCV Workshops), Barcelona, 2011, 383–390. DOI: 10.1109/ICCVW.2011.6130267.

[20] M. Van den Bergh, L. Van Gool, "Combining RGB and ToF cameras for real-time 3D hand gesture interaction". In: *IEEE Workshop on Applications of Computer Vision* (WACV), Kona, 2011, 66-72. DOI: 10.1109/WACV.2011.5711485.

[21] Y. Wen, H. Chuanyah, Y. Guanghui, W. Changbo, "A robust method of detecting hand gestures using depth sensors". In: *IEEE International Workshop on Haptic Audio Visual Environments and Games* (HAVE), Munich, 2012, 72–77. DOI: 10.1109/HAVE.2012.6374441.