Hand pose estimation using HOG features from RGB-D data

Constantina Raluca Mihalache
Faculty of Automatic Control and Computer Engineering
Technical University "Gheorghe Asachi"
Iasi, Romania 700050
Email: raluca.mihalache@tuiasi.ro

Bogdan Apostol
Faculty of Automatic Control and Computer Engineering
Technical University "Gheorghe Asachi"
Iasi, Romania 700050
Email: bogdan.apostol@tuiasi.ro

***Abstract*—Visual based recognition of hand gestures has been an active research field in recent years due to its efficiency in helping us achieve a more natural human-computer interaction. This paper presents a new approach to hand pose estimation using combined visual and geometric information obtained in a synchronized format from a RGB-D sensor. Firstly, we track the contour of the hand and recognize the fingertip positions. Then, Kernel Principal Component Analysis is used for selecting the most relevant elements from the histograms of oriented gradients feature vectors obtained on both color and depth data. We define an observation model based on the found fingertip positions and the selected principal components and we feed it as input for a Support Vector Machine classifier. Experimental results for the proposed method show that good detection percentages can be obtained with a small training dataset of real hand images and depth masks.**

I. INTRODUCTION

With the advances in video capturing techniques in recent years the Human Computer Interaction (HCI) [1] research area has gained growing interest. Accurate visual based recognition is an extremely challenging problem when it comes to tracking and recognizing fluid motion of articulated body parts. The hand is one of the most complicated parts for pose recovery as it has over 25 degrees of freedom [2] and irregular shapes. Controlling the computer with hand gestures seems to be the natural way of communication between human and machine.

Many techniques have been used to capture hand pose information, including different input devices like instrumented gloves, colored markers or accelerometers [3]. These methods suffer from freedom restriction of the human movement and are usually quite expensive. One of the well-studied alternatives to the input devices is estimating 3D hand poses in a non-intrusive manner on images or video sequences obtained from single or stereo cameras. The release of the Kinect device in 2010 has brought new affordable ways of capturing color images as well as depth information for a given scene. This has opened new possibilities for a wide range of applications in computer vision and pattern recognition such as sign language recognition, gestural communication and device automatic controlling.

Differences in background, illumination conditions, translation, rotation and scale of the human hand are some of the difficulties that one is faced with when trying to accurately segment the hand shape and track it in consecutive frames.

978-1-4799-2228-4/13/$31.00

In this paper, we focus our attention on estimating the hand pose using combined visual and geometric information from a RGB-D sensor.

We make an initial segmentation of the objects in the image based only on depth, and track the contour of the segmented objects in a binary validity mask. Algorithms such as [4] and the more recent [5] and [6] have been proposed for contour extraction. Curves along the contour of the segmented hand are found using a k-curvature algorithm similar to the one used in [7]. Fingertip positions are then selected based on a bisect orientation of the k-vectors.

Feature extraction from a grayscale image is a well-studied domain and has known some successful methods like Harris-Corner detection [8], Scale Invariant Feature Transform (SIFT) [9] and Histograms of oriented gradients (HOG) [10] that can characterize the local appearance of an object by aggregating gradients. In this paper, we compute HOG feature vectors on both color and depth data. Because hand depth data is very close to each other we apply a Cumulative Distribution Function (CDF) on the received raw data. This helps us create a well-defined grayscale depth image on which we can apply HOG feature extraction.

Kernel Principal Component Analysis (KPCA) method is used for selecting the most relevant features from the HOG vectors [11]. This technique is a kernel expansion of the linear Principal Component Analysis (PCA) with a non-linear manifold learning ability. It has already been shown that KPCA has a better performance than linear PCA [12]. KPCA also reduces the dimension of the input data by mapping the input higher dimensional space into a lower dimensional feature space [13] that is feed to our Support Vector Machine (SVM) training algorithm.

We propose a new approach to SVM training patterns by using the first principal components of HOG RGB-D data and the detected fingertips positions. We adopted a supervised learning method for the SVM classifier so that we split the training set into multiple classes corresponding to the number of considered hand poses. The SVM classifier is trained offline using a small set of training data and tests with online data show promising results of up to 89% success detection rate.

The outline of this paper is as follows: Section II presents data interpretation, contour tracking and finger recognition algorithms; Section III explains the details of calculating HOG feature vectors on both visual and geometric data; Section IV

describes the selection of the most important features from HOG RGB-D data; Section V proposes a method of building training patterns for SVM classifier; Section VI outlines the experimental results of the method and Section VII concludes with a summary and discussion.

II. CONTOUR TRACKING AND FINGER RECOGNITION

A. RGB-D Data

Estimating the hand pose involves finding and tracking the hand contour in the captured data. Detection of the components of a hand model, such as the center of the palm and the position of the fingertips are also equally important. This assumes the fact that we can obtain a good segmentation of the hand in the received image.

RGB-D data combines visual (RGB colors) and geometric information (depth) in a synchronized format that allows us to do some tasks that were difficult before. Real time object segmentation is one task that can be done faster and with a higher efficiency using both depth and color information. In our algorithm we use a video stream input and the corresponding depth stream for each captured frame obtained from a Microsoft Windows Kinect sensor.

Kinect was originally released in November 2010 by Microsoft as a HCI device that permits the user to interact with the XBox 360 gaming console using only gestures and spoken commands. Since its release the sensor has become increasingly popular also in the research field mainly because of the reduced cost of such a device compared to similar depth capturing devices such as stereo cameras.

We obtain the video stream from Kinect as an array A_{rgb} of size 640×480 = 307200 RGB pixels at 30 Hz. The depth data stream can be obtained in three different array sizes 80×60, 320×240 or 640×480, and have values corresponding to distances in millimeters for each pixel that range from 1 to 10.000mm. In this paper we will use an identical size array A_d with a 640×480 = 307200 resolution for the depth data. However, there is enough detail to distinguish the contour of the hand to also use a smaller depth resolution of 320×240, which will reduce the number of calculations needed in the algorithm.

B. Hand Segmentation

Similar to [15] we define the observation model O_h that feeds the rest of the algorithm as:

$$O_h = \Big(H^*(I_c); H^*(I_d); F\Big) \tag{1}$$

where I_c and I_d are two arrays containing the 2D maps of the segmented image of the hand and the corresponding valid depth map, H^* is the selection of the most important HOG features and F represents an array of the fingertips positions.

A requirement of our algorithm is that the user is in the near to far range of the Kinect camera and that the front-most object facing the sensor is one or more hands. As hand pose recognition is usually used for HCI it is only natural that the hands of the user be in front of the body and in the proximity of the sensor. This assumption does not impose a limitation to the algorithm.

Let there be $d_{min}(x, y)$ the closest point, and thus the one that has the smallest value in A_d. We have to reduce the whole data to a 3D bounding box that encloses the hand. First we will limit depth value based on an empirical obtained threshold Δ that is calculated as a percent of the remaining depth difference between the closest point d_{min} and the furthest point d_{max}. All values that are not in the interval $[d_{min}, d_{min} + \Delta]$ where $d_{min} + \Delta < d_{max}$ are discarded. The bounding box containing the hand can be visualized in Fig. 1 where we have drawn the bounding box over the depth data.

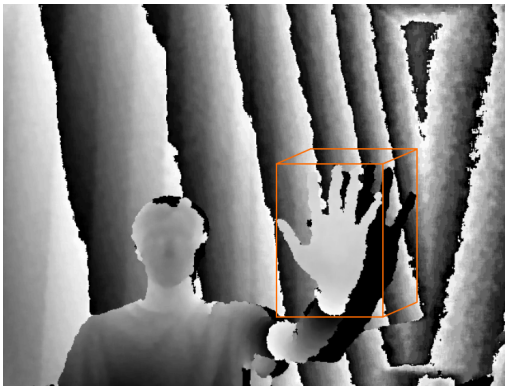


Fig. 1. Hand bounding box for considered depth interval in depth image obtained from Kinect sensor.

The obtained depth values from A_d contain the valid pixels that must be considered from A_{rgb} and thus for each valid value the segmented color image will contain the corresponding color pixel information. However, the data obtained from depth may have unclear contours or missing pixels due to the noise introduced by the Kinect depth measurement.

Optimization of the obtained binary matrix which contains the shape of the hand is done through two common morphological operations used in image segmentation. Thus we expand the contours of the segmentation by applying a dilation operation and then contracting the obtained contours with an erosion operation, applied on a binary image generated by the valid data in A_d. The first operation fills small areas where information might be missing and the second one keeps the size of the contour while generating smoother contours. By applying the filtered and optimized A_d on the original color information A_{rgb} we obtain the final segmentation of the hand I_c. The image I_c is of variable size, depending on the size of the hand segmentation.

The depth map values corresponding to the segmented hand pixels are then transformed in a grayscale image, noted by us with I_d, where the light values are closer and the darker grays are further away from the camera. Because points are closer to each other near the average depth μ_d we used for mapping and representing in grayscale value a cumulative distribution function $\phi_{\mu\sigma^2}(d)$ centered in μ_d. Thus closer points to the average value μ_d will be more differentiated on the grayscale. Fig. 2 presents the scatter plot of an original depth information in the left and the scatter plot of the coresponding CDF in the right. The segmented hand and the depth mask obtained have the same size.

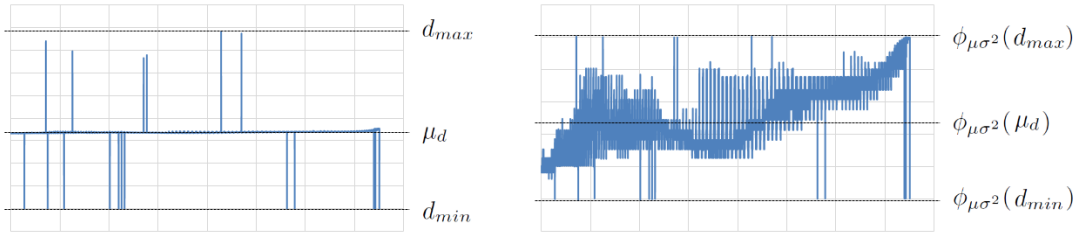
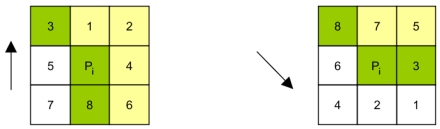


Fig. 2. Scatter plots of depth data: Raw Kinect array in the left plot; Normalised CDF values array in the right plot.

C. Contour Tracking and Fingertip Recognition

Hand contour tracking in the valid depth data obtained from the previous step involves classifying valid positions in either contour or interior pixels. A contour pixel is a valid pixel that has at least one adjacent pixel that is not valid. On the other hand an inside pixel has all the adjacent pixels valid.

Contour tracking algorithms in binary images, such as [4], [5] and the more recent [6], [14] have been proposed. The algorithms work by scanning in different directions with starting from different positions to find the first pixel in the contour. Scanning the depth image from the bottom and up for a valid contour pixel or from the middle of the image [14] yields the same result for us because one of the advantages of our method is that the validity map is in fact a bounding box for our hand and for each line in the map there is at least one pixel on the contour.



(a) (b)

Fig. 3. Two out of 8 possible search directions for contour tracking. (a) Last valid pixel was found in the up direction. (b) Last valid pixel was found in the down-right direction.

After finding the first valid contour pixel we begin searching in the 3×3 neighborhood of the pixel and adding contour pixels in an ordered list. For each pixel that we add in the list we also keep the last direction that the pixel was found in comparison with the previous pixel. We take into consideration 8 directions with the first in the direction of the last pixel and then the others by always expanding by one. We describe this for two examples in Fig. 3 where we show the search order for different search directions. The direction for the first pixel is set to up. To avoid the adding of the same pixel multiple times in the list we use a visited array and a backtracking algorithm for continuous contours.

To recognize fingertips along the contour of the hand we have to first find all the curves in the contour list and then classify them in fingertips or valleys. The method we used in this paper is a variation of the k-curvature algorithm similar to the one used in [14].

The k-curvature algorithm chooses for each point P_i in the contour, two points P_{i-k} respectively P_{i+k} and defines two vectors $\mathbf{P_iP_{i-k}}$ respectively $\mathbf{P_iP_{i+k}}$ and calculates the angle ω between them. The point P_i is considered a valid curve if ω has a smaller value then an empirical chosen value. The k value and ω threshold value are defined by experiments. For this paper k is set to 22 and ω is set to 0.8726 radians, or 50 degrees. The optimal angle was determined by calculating an average of measured angles in the depth map.

To find out if a detected curve is a fingertip or a finger valley we calculate the bisect between the $\mathbf{P_iP_{i-k}}$ and $\mathbf{P_iP_{i+k}}$. The curve is a fingertip only if the bisect points to the interior of the hand, see Fig. 4.

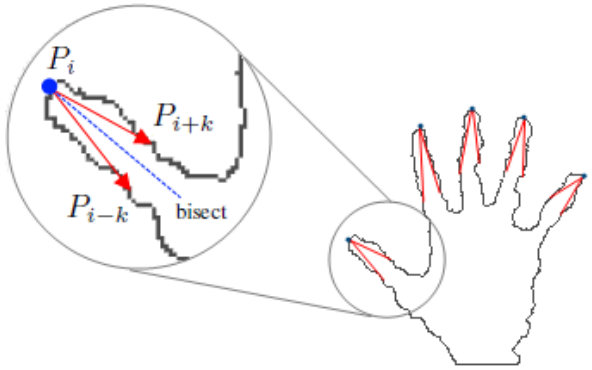


Fig. 4. k-curvature algorithm applied for contour detection and choosing fingertips by bisect orientation.

The list of all points P_i that are found as fingertips on the contour compose the F element in our observation model.

III. HOG ON RGB-D DATA

N.Dalal et al. [10] proposed the extraction of the HOG features from each location in a grid that is placed over the image. For each such cell in the grid, orientation histograms of edge intensities are calculated for the local region. The idea of the algorithm is that this local intensity gradients can describe well an object by its shape and appearance without the need of knowing the original positions of the calculated gradients.

In this section we describe how we extract the HOG features from the grayscale image of the RGB segmented hand I_c and also from the grayscale image obtained by remapping depth values I_d. Thus we obtain two HOG feature vectors that will be used as input for the SVM classification algorithm.

We quantize all the gradient directions in the image into 9 valid directions as shown in Fig. 5. The directions are calculated over small overlapping spatial regions called *cells*. The cells are obtained by dividing the grayscale image into an evenly spaced grid.

Each cell gets assigned a histogram which accumulates gradient directions in 9 corresponding bins. As the cells are equal in size the total number of gradients that composes each cell will be always the same. The cells are then organized

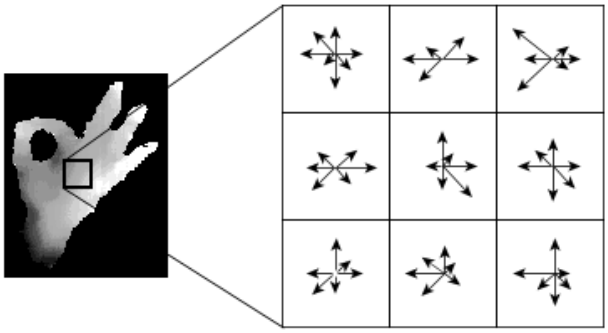


Fig. 5. Extraction process of HOG features from normalized depth image on a 3×3 block.

in rectangular blocks, and the HOG feature of the block is formed by concatenating the cell HOGs. Finally, the HOGs corresponding to the blocks are composed in an overall image HOG vector. However, because the input images differs in size from frame to frame, as the segmented hand might change position or orientation, the overall HOG feature vector will also be different in size. For the tests in this paper we used for each image (color and depth) an overall HOG feature obtained by splitting the image into 6×6 pixel cells, organized in blocks composed of 3×3 cells.

IV. Kernel Principal Component Analysis of RGB-D HOG

Thousands of HOG features are extracted from RGB-D data and probably many of them are redundant. For reducing the dimensionality of the feature vectors we used the Kernel Principal Component Analysis method. This technique is a kernel expansion of the linear PCA which has a non-linear manifold learning ability. It has already been shown that KPCA has a better performance than linear PCA [12]. An important advantage of KPCA is its ability to ignore the noise from input data while the noise from test data is removed by projecting it onto the manifold [11]. KPCA feature extraction was applied in solving many problems, including pattern recognition, regression estimation and noise reduction.

In this section we will briefly describe the KPCA algorithm. Given an input data set $X = x_1, x_2, \ldots, x_n$, KPCA performs a non-linear transformation via a non-linear mapping function $\phi(x) : x \to k(x, .)$ of X to a high dimensional feature space S and then computes linear PCA on the mapped dataset $\Phi = \phi(x_1), \phi(x_2), \ldots, \phi(x_n)$, where $k(.,.)$ is a predefined kernel function.

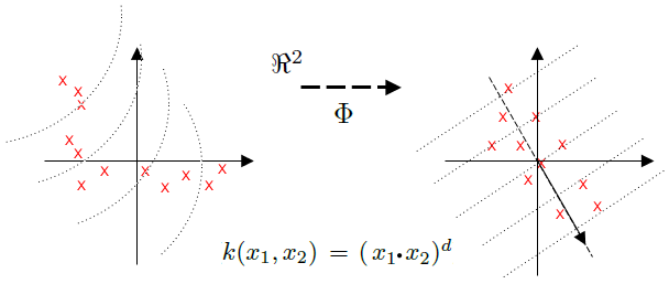


Fig. 6. Performing PCA in a high dimensional space S using a kernel function k.

Having an input HOG data sets $X = \{x_1, \ldots, x_n\}$ extracted from a number of n training images, where x_i is the HOG for the i^{th} training image, we defined the kernel $k(x, y)$ and compute the kernel matrix [11]

$$K = [K_{ij}], K_{ij} = k(x_i, x_j), \tag{2}$$

where $i, j = \overline{1, n}$ and centered kernel matrix:

$$\hat{K} = HKH \tag{3}$$

where $H = I_n - 1_n$, I_n is the $n \times n$ identity matrix and 1_n is a $n \times n$ matrix which has $1/n$ as value for each element.

In this paper we used both, Gaussian Radial Basis Function (RBF) kernel $k(x, y) = e^{-\frac{||x-y||^2}{2\sigma^2}}$ and Polynomial kernel $k(x, y) = (xy)^d$ of degree $d = 2$ and $\sigma = 0.223$. The original data set Φ is mapped into an infinite dimensional space S and any finite data set is linear separable in the feature space with respect to this kernel.

The l highest eigenvalues are computed $\lambda_1 ... \lambda_l$ and their corresponding eigenvectors:

$$A = [a_1, \ldots, a_l] \tag{4}$$

of $\hat{K}$, where $a_i, i = 1, 2, ..., l$ are n dimensional column vectors and scaled such that $|a_i| = 1/\lambda_i$.

The trained KPCA model stores matrix A and all training samples as parameters. The projection z of a test data set (represented by the HOG features extracted from a new test image) onto the first l principal components in the feature space is defined by:

$$y(z) = w(z) - \mu \tag{5}$$

$$w(z) = (H_A)^T (k(x_1, z), \ldots, k(x_n, z))^T = (H_A)^T k(X, z) \tag{6}$$

$$\mu = (H_A)^T K 1_{1,n} \tag{7}$$

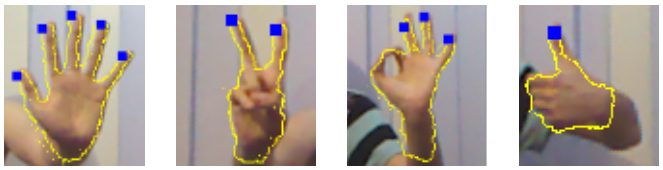
where $H_A = HA$, $w(z)$ and μ are l-dimensional vectors representing the projection of $\phi(z)$ onto the principal components and the mean of $\phi(x)$ over the n training patterns and $1_{1,n}$ is a n-dimensional column vector in which each element has the value $1/n$.

V. Hand Pose Estimation using SVM

In this section we will detail the proposed method of hand pose estimation using SVM classifier and clearly present the training and testing steps. The training phase consists of three stages: (1) gather RGB images and their associated depth information for different m hand poses from Kinect sensor, (2) dimensionality reduction of HOG features of both color and depth images using KPCA, (3) building input patterns and train SVM classifier.

Our approach of building SVM training patterns is presented according to equation 1. We define the set of p training patterns over m hand poses as $I_{train} = \{O_{ij}\}, i = \overline{1, p}, j = \overline{1, m}$. We adopted a supervised learning method for the SVM classifier so that we split this training set into m classes corresponding to the number of considered hand poses (see Fig. 7).

A training pattern that is considered as input in SVM classifier is a vector that results by applying KPCA on the HOG data as we described in section 4. We choose the set of l_{rgb} ($l_{rgb} < n_{rgb}$) and the set of l_d ($l_d < n_d$) eigenvectors which have the l_{rgb} and l_d largest eigenvalues (l_{rgb} and n_{rgb}, l_d and n_d are the number l of the largest eigenvalues and the dimension of the covariance matrix for the color and depth HOG features, respective). Proposed pattern representation includes the first l_{rgb} and l_d principal components of HOG data applied on i^{th} color and depth image, respective. The last element in our training pattern is represented by the positions of identified fingers in section 2.



(a) (b) (c) (d)

Fig. 7. Gesture set used in our study. (a) Open hand pose. (b) Peace pose. (c) Ok pose. (d) Like pose.

SVM is a supervised machine learning algorithm which deals with a set of training patterns, each marked as belonging to one of the many categories. We use SVM training algorithm for building a model that predicts the category of the new hand observation model. It is already proved that SVM has the greater ability to generalize the problem, which is the goal in statistical learning [16].

Given a set of training patterns $(\mathbf{x_1}, y_1) \ldots (\mathbf{x_n}, y_n)$ in $\Re^n \times \Re$, where $y_i \in \Re$, $i = \overline{1,n}$, according to undefined probability distribution $P(\mathbf{x}, y)$, and a error function $V(y, f(\mathbf{x}))$ where $f(\mathbf{x})$ is the predicted output instead of ideal output y for the input $\mathbf{x}$ our problem consists in finding the function f that minimizes the error: $\int_1^n V(y, f(\mathbf{x}))P(\mathbf{x}, y)d_x d_y$.

Because we deal with non-linear separable patterns we map them using a $\Phi(\mathbf{x})$ kernel function into a higher dimension space in order to make them linearly separable. The goal of SVM is to find a hyper plane $\mathbf{w} \cdot \mathbf{x} - b = 0$, which separates the two different samples accurately by maximizing the bilateral blank area $\frac{2}{\|\mathbf{w}\|}$ maximum. Thus, our problem resumes to $min\left\{\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w})\right\}$, where $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) >= 1, i = \overline{1,n}$.

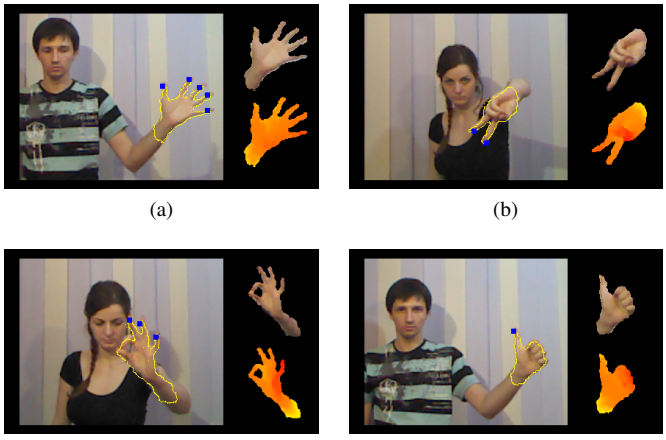
The trained SVM model is tested on q testing patterns ($I_{test} = \{O_{ij}\}, i = \overline{1,q}, j = \overline{1,m}, q >> p$) over the same m hand poses. Testing is made online and for each received image from Kinect sensor an observation model is created in order to estimate its hand position.

VI. EXPERIMENTAL RESULTS

This section will present some experimental hand pose estimation results that are obtained by applying the algorithm described in the previous sections firstly offline for training with a set of training data and then online for estimation in real time of hand position and gesture recognition. We compare results obtained from applying the recognition based only on the filtered HOG features from normal imagery to results obtained with also adding extraction of the most important features of HOG from a normalized depth image. We make evaluations with different kernel functions for KPCA and SVM and compare their accuracy rate.

To test the performance of our proposed method we developed an application that uses RGB-D raw data received from a single Microsoft Kinect Sensor for Windows on an Intel Core i7 2.20 GHz machine. We use the official SDK for retrieving the video stream as an 640×480 (= 307200) size RGB pixels array at a frequency of 30FPS and an identical size depth data array that has values corresponding to distances in millimeters for each pixel.

From the received data we extract only the information in a bounding box that surrounds the segmented hand. This bounding box has variable dimensions depending on hand rotation and distance from the camera and thus the segmented depth and color images have different sizes with a maximum resolution possible equal to 640×480. Depth data is normalized and represented as a grayscale image so that we can extract HOG features from it similar to the extraction method on color imagery. An overall HOG feature is obtained by splitting the image into 6×6 pixel cells, organized in blocks composed of 3×3 cells and using a 9 bin histogram per cell. The features vectors obtained by applying HOG algorithm have different lengths due to different resolutions in input data. We select a maximum of 3 principal components from HOG features using KPCA method that are then used for composing the training pattern for the SVM classifier.



(a) (b) (c) (d)

Fig. 8. Successful recognition results for each considered class: (a) Open hand pose. (b) Peace pose. (c) Ok pose. (d) Like pose.

We have trained our system with a number of 200 observation models (principal components of color and depth image HOGs and fingertip positions) that were separated in five corresponding classes of hand poses. The open hand, peace sign, ok sign and like sign classes were chosen for tests in this paper. For the fifth class we used some non-hand observation models that allows us to also have negative patterns learned. Training models were obtained by capturing real life Kinect data from two test subjects and different hand pose angles to the camera. Online pose estimation including hand segmentation, contour and fingertip tracking, extraction of HOG features, selection of principal components and SVM classification, have shown promising near real-time results (20-25FPS).

Figure (Fig.8a), (Fig.8b), (Fig.8c) and (Fig.8d) show successful recognition results for each considered class obtained by using our method on real-time captured data from the Kinect device.

In Tab. I we show the accuracy rate of tests conducted on 1000 test patterns using only HOG features on color images is compared with the accuracy rate of using both color and depth features. We can see that adding HOG depth features greatly improves the accuracy rate of detection.

TABLE I. ACCURACY RATES WITH AND WITHOUT DEPTH HOG FEATURES

Selected PC number	KPCA-SVM on HOG RGB	KPCA-SVM on HOG RGB-D
1	81%	89%
2	67%	71%
3	53%	69%

A comparative study of the results obtained using different KPCA and SVM kernel functions is presented in Tab. II. It is shown that KPCA-SVM with Gaussian RBF and recursive feature elimination method performs better.

TABLE II. COMPARATIVE STUDY WITH DIFFERENT KERNEL FUNCTIONS

Support Vector Machine kernel	KPCA RBF kernel	KPCA Polynomial kernel
RBF (voting)	87%	76%
Polynomial (voting)	79%	77%
RBF (elimination)	89%	79%
Polynomial (elimination)	81%	84%

VII. CONCLUSIONS

In this paper, we proposed a new approach to hand pose estimation using features derived from combined visual and geometric information obtained from a Kinect sensor. We classify each depth in the image as valid or invalid depth and we create a validity binary mask. Based on this mask we make the hand segmentation from the RGB image received and find contours for the selected regions. A k-curvature algorithm is then used to find curves in the contour and a bisect orientation method is used to extract valid fingertips.

Training is done using a SVM classifier that receives an input observation model composed of KPCA components of the HOG feature vectors for both image and depth information together with the recognized fingertip positions. Creating the HOG features on the Kinect depth data alone is not enough because the gradient orientation would be similar for the whole segmented image. Thus we used a normalization method that makes depth difference near the average value more obvious.

Based on results obtained by measuring pose estimation accuracy using HOG features on RGB data alone and accuracy rates obtained with our method when also using depth data HOG features we have observed that our algorithm obtains better detection rates. Experiments conducted with the proposed method show good detection percentages that can be obtained with a relatively small training dataset.

REFERENCES

[1] S. S. Rautaray, A. Agrawal, *Vision based hand gesture recognition for human computer interaction: a survey*. Artificial Intelligence Review, 2012, pp. 1-54.

[2] I. Oikonomidis, N. Kyriazis and A. A. Argyros, *Markerless and efficient 26-DOF hand pose recovery*. Proceedings of the 10th Asian conference on Computer vision, 2011, pp. 744-757.

[3] N. A. Ibraheem and R. Khan, *Survey on Various Gesture Recognition Technologies and Techniques*. International Journal of Computer Applications, 2012, vol. 50, pp. 38-44.

[4] M. Ren, J. Yang, and H. Sun, *Tracing boundary contours in a binary image*. Image and vision computing, 2002, pp. 125-131.

[5] F. Chang, C. Chen, and C. Lu, *A linear-time component-labeling algorithm using contour tracing technique*. Computer Vision and Image Understanding, 2004, pp. 206-220.

[6] L. Yan and Z. Min, *A new contour tracing automaton in binary image*. 2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2011, pp. 577-581.

[7] T. Trigo and S. Pellegrino, *An analysis of features for hand-gesture classification*. 17th International Conference on Systems, Signals and Image Processing (IWSSIP 2010), 2010, pp. 412-415.

[8] C. Harris and M. Stephens, *A combined corner and edge detection*. Proceedings of The Fourth Alvey Vision Conference, 1988, pp. 147-151.

[9] D. G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*. Int. J. Comput. Vision, November 2004, pp. 91-110.

[10] N. Dalal and B. Triggs, *Histograms of oriented gradients for human detection*. Computer Vision and Pattern Recognition (1), 2005, pp. 886-893.

[11] P. Cheng, W. Li and P. Ogunbona, *Kernel PCA of HOG features for posture detection*. International Conference: Image and Vision Computing New Zealand, 2009, pp. 415-420.

[12] M. Karg, R. Jenke, W. Seiberl, K. Kuhnlenz, A. Schwirtz, M. Buss, *A Comparison of PCA, KPCA and LDA for Feature Extraction to Recognize Affect in Gait Kinematics*. 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII), 2009, pp. 1-6.

[13] Y. Wu, X. Wabg and F. Shang, *Study on 3D Object Recognition Based on KPCA-SVM*. International Conference on Information and Intelligent Computing IPCSIT, vol.18 IACSIT Press, 2011, pp. 55-60.

[14] D. J. Ryan, *Finger and gesture recognition with Microsoft Kinect*. Master's thesis in Computer science (TN-IDE), 2012.

[15] I. Oikonomidis, N. Kyriazis and A. Argyros, *Efficient model-based 3D tracking of hand articulations using Kinect*. Proceedings of the British Machine Vision Conference, 2011, pp. 1-11.

[16] A. Pradhan, *Support Vector Machine - A Survey*. International Journal of Emerging Technology and Advanced Engineering, 2012, pp. 82-85.

[17] T. Kobayashi, A. Hidaka, and T. Kurita, *Selection of Histograms of Oriented Gradients Features for Pedestrian Detection*. In proceeding of: Neural Information Processing, 14th International Conference (ICONIP), 2007, pp. 598-607.

[18] Q. Wang, *Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models*. CoRR, 2012.

[19] Q. Chen, X. Chen and Y. Wu, *Optimization Algorithm with Kernel PCA to Support Vector Machines for Time Series Prediction*. Journal Of Computers, 2010.

[20] P. Padeleris, X. Zabulis and A. A. Argyros, *Head pose estimation on depth data based on Particle Swarm Optimization*. Computer Vision and Pattern Recognition Workshops (CVPRW), 2012, pp. 42-49.