3D Hand Shape Analysis for Palm and Fingers Identification

Ludovico Minto, Giulio Marin, Pietro Zanuttigh
Department of Information Engineering, University of Padova, Padova, Italy

***Abstract*— This paper proposes a novel scheme for the extraction and identification of the palm and the fingers from a single depth map. The hand is firstly segmented from the rest of the scene, then it is divided into palm and fingers regions. For this task we employed a novel scheme that exploits the idea that fingers have a tubular shape while the palm is more planar. Following this rationale we applied a contraction guided by the normals in order to reduce the fingers into thinner structures that can be identified by analyzing the changes in the point density. Density-based clustering is then applied and finally a linear programming based approach is employed to identify the various fingers. Experimental results prove the effectiveness of the proposed approach even in complex situations and in presence of inter-occlusions between the various fingers.**

I. Introduction

Hand pose estimation is a very challenging task due to the large number of degrees of freedom associated to the hand movements and to the large number of inter-occlusions between the various fingers. Several computer vision approaches have been presented for this problem [4], but its solution from a single video stream remains very difficult. The recent introduction of consumer depth cameras [1] have made depth acquisition available to the mass market and has paved the way to the introduction of a set of new hand pose estimation approaches exploiting the three-dimensional information contained in depth data.

This paper follows this rationale and introduces a novel hand shape analysis scheme exploiting depth data from a consumer depth camera. The proposed approach works on a single depth map without exploiting the temporal coherence and is based on two main steps. It firstly identifies the 3D points representing the hand shape and divides them into the palm and the fingers region. This is achieved by combining a novel approach that contracts high curvature regions typically associated to fingers into thinner structures with more standard techniques based on shape fitting and clustering. Then the fingers region is further subdivided into segments corresponding to the various fingers. For this task density-based clustering is exploited together with a linear programming optimization that is used to recognize the clusters belonging to the same finger. Challenging issues like the occlusions due to the fingers folded over the palm or inter occlusions between the various fingers are handled reliably by the proposed approach as the experimental results show.

II. Related Works

Hand pose estimation is a long-term research field and a large number of approaches based on video data have been proposed [4]. As previously noted, a bidimensional description is often not sufficient to solve this problem and recent schemes have focused on the exploitation of 3D information acquired with consumer depth cameras.

In particular some recent works focused on the estimation of the hand pose from a single depth map. The work of Keskin et Al. [7] has adapted the Kinect skeletal tracking scheme to hand pose estimation. A coarse-to-fine search based on Latent Regression Forests, has been proposed in [12]. Regression Forests are also used in [14], where the error between the data and the model based on subdivision surfaces is optimized. Another very recent approach is [10] where the hand is modeled with a set of spheres and a cost function is optimized using Particle Swarm Optimization. The approach of [9] searches the optimal solution in the high-dimensional parameter space of hand configurations by using an evolutionary optimization technique. Dense feature extraction exploiting convolutional networks has been used for hand pose estimation in [16]. Another approach exploiting convolutional networks is [8]. Finally [13] uses transductive regression forest exploiting both real and synthetic data for the training.

III. Proposed Approach

The target of the proposed algorithm is to identify the five fingers and the palm region by processing the point cloud of the hand acquired by a depth sensor. Firstly the hand point cloud $\mathcal{H}^s$ that will be used as input for the following stages is extracted from the rest of the scene using the method proposed in [2], [3]. This method exploits a combination of thresholding of the depth values and geometrical constraints on the object size to recognize and segment the hand from the rest of the scene. The proposed approach then follows the pipeline shown in Fig. 1, as described in detail in this section.

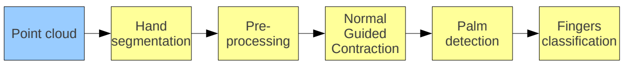

Fig. 1. Pipeline of the proposed approach.

A. *Point cloud pre-processing*

The segmented hand point cloud $\mathcal{H}^s$ is pre-processed to remove some artifacts typical of low cost depth cameras. Firstly, isolated points are removed by using a statistical outlier removal: for each point, the mean distance of the k nearest neighbors is computed, then the mean and standard deviation of the distances are estimated. Points having too high average distance from the mean are classified as outliers. Notice how in this step we are interested in removing only artifacts, therefore the threshold to discriminate outliers is relatively high in order to reduce the chance of removing relevant points. A bilateral filter [15] is then applied in order to reduce the noise while preserving the details and edges of the hand shape. After these steps, a filtered point cloud $\mathcal{H} = \{p_1, ..., p_n\}$ containing the hand samples is available. A reference gesture has been chosen to visualize all the intermediate processing, the corresponding point cloud $\mathcal{H}$ is shown in Fig. 2a.

B. Normal Guided Contraction

A key aspect of the proposed approach is the point cloud processing scheme, that we will denote with Normal Guided Contraction (NGC), introduced in order to better highlight the fingers region. By analyzing the hand shape, it is clear that fingers have a tubular shape while the region of the palm is mostly planar. We exploit this topological difference to better detect and separate the various fingers. From the point cloud we are able to compute for each point p_i a unit vector $\mathbf{n}_i$ that represents quite accurately the normal to the surface in that point. A new point cloud $\mathcal{H}^c = \{p_1^c, ..., p_n^c\}$ is built by moving each point p_i in the direction opposite to the surface normal $\mathbf{n}_i$ at that location, i.e.:

$$p_i^c = p_i - t\mathbf{n}_i \tag{1}$$

The offset t is set to a fixed value, corresponding approximately to the average radius of a finger, in order to maximize the contraction of the fingers regions (for the experimental results we used $t = 9[mm]$). In this way the tubular surfaces are contracted into thinner structures, while planar surfaces are just shifted of a small amount in the direction perpendicular to the plane, keeping the same point density. The idea is that after the contraction step, the high density regions are more likely to be fingers while low density regions are associated to the palm, as can be seen in Fig. 2b. From the figure (look at the pinkie and ring fingers) it is also possible to notice how attached fingers turn out to be spatially more separated then they were before contraction, making easier the task of dividing fingers very close one to the other. Nevertheless, some difficulties may arise when trying to properly recognize the fingers by directly looking at the contracted cloud density, since there may be regions, e.g. the edge of the palm, which could have a high density without being fingers. The resulting point cloud $\mathcal{H}^c$, shown in Fig. 2b, is the input for the next step.

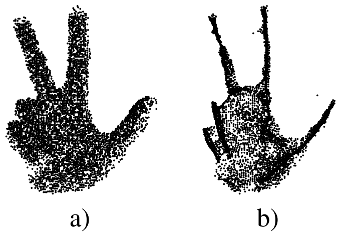

Fig. 2. Normal guided contraction of the hand point cloud: a) Original point cloud filtered ($\mathcal{H}$); b) Contracted version of the point cloud ($\mathcal{H}^c$).

C. Palm Detection

The next step is the segmentation of the hand into the palm and fingers region. This operation is simple in the case of raised fingers but becomes very challenging when the fingers are bent over the palm. Notice how many approaches based on the hand silhouette and also on depth data, e.g., [3], [11], are able to recognize only the raised fingers and typically assign to the palm region all the samples in the lower hand area including the ones corresponding to bent fingers.

After the NGC, we intuitively associate the samples of $\mathcal{H}^c$ within the high density regions to the fingers, the remaining points belonging to the palm. A naive approach to divide the two clusters is to consider a threshold on the number of points inside a spherical neighborhood of a given point in the contracted cloud. Some regions of the palm showing an initial density greater than the one of finger samples may however maintain a final high density even when subject to a slight contraction. Instead, the number of misclassified points is greatly reduced if, given a point, we consider its neighborhood and compare the original spacing between samples in the point cloud $\mathcal{H}$ with the spacing in the contracted point cloud $\mathcal{H}^c$. In order to label the ith point in the cloud as finger $\mathcal{F}$ or palm $\mathcal{P}$, we first consider the set of its k closest points in the contracted cloud $\mathcal{N}_{i,k}^c = \{p_{j_1}^c, ..., p_{j_k}^c\}$ and compute their average distance from p_i^c. We then consider the same neighbors as they appears in the original cloud, that is $\mathcal{N}_{i,k} = \{p_{j_1}, ..., p_{j_k}\}$, and compute their average distance from p_i. The ratio between the average distances before and after the contraction is then compared to the average of the same ratio computed in the overall hand point cloud. Points with a ratio greater than the average are assigned to the finger set $\mathcal{F}$ while the others are assigned to the palm set $\mathcal{P}$, i.e,:

$$\begin{aligned} \bar{d}_i &= \frac{\sum_{s=1}^k \|p_{j_s} - p_i\|}{k} \\ \bar{d}_i^c &= \frac{\sum_{s=1}^k \|p_{j_s}^c - p_i^c\|}{k} \\ r_i &= \bar{d}_i / \bar{d}_i^c \\ \bar{r} &= \frac{\sum_{i=1}^n r_i}{n} \end{aligned} \tag{2}$$

$$\begin{aligned} r_i < \bar{r} \quad &\Rightarrow \quad p_i^c \in \mathcal{P} \\ r_i \geq \bar{r} \quad &\Rightarrow \quad p_i^c \in \mathcal{F} \end{aligned} \tag{3}$$

Fig. 3 helps to better understand this step. Let us first consider a region associated to fingers (shown in Fig. 3a), the average spacing between a point and its neighbors in $\mathcal{N}_{i,k}^c$ is much smaller than the one computed with respect to $\mathcal{N}_{i,k}$. In Fig. 3b, an internal region of the palm is shown, where the spacing does not decrease after the contraction, as the normals in this region are almost parallel. Fig. 3c shows instead a boundary region of the palm, where the spacing decreases but not as significantly as in the fingers region. Here in fact, differently from the fingers, there are more parallel normals or in general the curvature is less pronounced. We decided to use the mean of all the ratios as threshold value, but of course a different thresholding criteria can be used. Fig. 4a shows the output of this first raw assignment; notice how, by working with point clouds and using densities in the 3D space, the proposed approach is invariant to rotations and to the orientation of the hand.

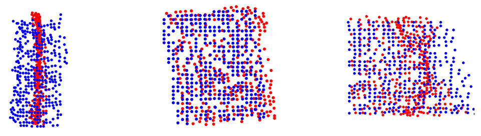

Fig. 3. Difference of the density before and after the contraction for three particular regions (*best viewed in colors*, *blue* points belong to the original point cloud $\mathcal{H}$, *red* points belong to the contracted point cloud $\mathcal{H}^c$): a) Fingers region; b) Palm region; c) Palm edge region.

After this operation, there could still be some isolated spots of erroneously classified points, especially along the palm edges. A refinement process is therefore needed. In particular, small spots labeled as fingers surrounded by larger areas labeled as palm are very likely to be artifacts. For this reason we iteratively check for each point the ratio between palm points and finger points in a neighborhood of the point itself and update its label according to this ratio. To be more robust, we define two thresholds δ_f and δ_p:

$$\begin{aligned} \frac{|\mathcal{N}^c_{i,k} \cap \mathcal{F}|}{|\mathcal{N}^c_{i,k} \cap \mathcal{P}|} > \delta_f \quad &\Rightarrow \quad p^c_i \in \mathcal{F} \\ \frac{|\mathcal{N}^c_{i,k} \cap \mathcal{P}|}{|\mathcal{N}^c_{i,k} \cap \mathcal{F}|} > \delta_p \quad &\Rightarrow \quad p^c_i \in \mathcal{P} \end{aligned} \tag{4}$$

The two thresholds should be both larger than 1 (e.g. $\delta_f = 1.2$ and $\delta_p = 1.5$ in the experimental results), in order to ensure that the assignment is changed only if the sample is surrounded by a large set of samples in the other region. Different values however do not affect too much the results. From Fig. 4c we can notice how the small spots classified as fingers in the region of the palm by the first raw estimation are now correctly classified.

At this point we have a good estimate of the region of the palm and we can compute the best plane fitting the points in $\mathcal{P}$ using SVD and RANSAC [6]. Exploiting this information we get the normal of the plane that represents a good estimate of the hand orientation, a very useful information for any gesture recognition and pose estimation approach. Finally, in order to help the clustering of the fingers in the following step, isolated points belonging to $\mathcal{F}$ are removed from $\mathcal{H}^c$. Notice that after the contraction, fingers samples are typically collapsed into thin regions. Thus, the outlier removal can be applied using a more strict threshold without the risk of removing relevant points, obtaining better results than in the initial removal step. The points marked as isolated are removed also from $\mathcal{H}$.

D. Fingers Identification

We based our finger detection scheme on the notion of density, therefore we decided to use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [5] to

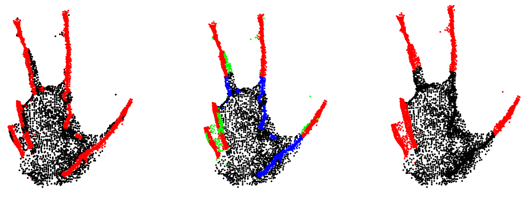

Fig. 4. Fingers and palm classification according to the ratio of the density before and after the contraction (*best viewed in colors*, *red* samples are associated to fingers, *black* to the palm): a) Contracted cloud with the labels after the first assignment; b) Refinement process: in *blue* the finger points that are relabeled as palm, in *green* the palm points that are relabeled as fingers; c) Final assignment after the refinement.

cluster points in $\mathcal{F}$. This algorithm proved to have better performances than standard clustering techniques on this particular type of data. In particular it is able to properly extend the clusters along the fingers point clouds which after the contraction are typically made of thin dense regions with sharp turns in correspondence of finger joints. DBSCAN requires two parameters: the radius ε of the sphere to be used to check if neighbors are reachable, and the minimum number of points that have to be in the ε-neighborhood to consider the point in the cluster. In this step we do not force the output to have five clusters, rather we set the parameters in order to be sure that the algorithm does not cluster more than one finger together (i.e., we slightly over-segment, allowing a finger to be split into more segments but ensuring that multiple fingers are not included in a single segment). This algorithm has several advantages over other classical data clustering algorithms when applied to this specific task, since it does not require to specify the number of clusters to search for, and it is able to discover noise and outliers. Furthermore, it properly handles arbitrary shaped clusters. The input of DBSCAN is the filtered fingers region shown in *red* in Fig. 4b and the result is shown in Fig. 5a.

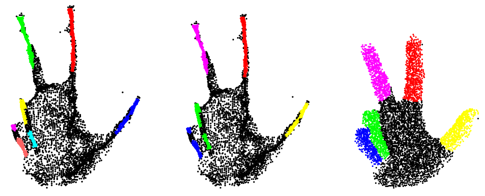

Fig. 5. Fingers classification: a) Clusters computed by the DBSCAN algorithm, each color corresponds to a different cluster; b) Fingers classification (shown on the contracted point) given by the solution of the linear assignment problem; c) Fingers classification shown on the original point cloud.

The task of gathering together clusters belonging to the same finger is not easy. We address this problem by modeling it as a linear binary optimization problem. As a basis for our approach there is the idea that clusters belonging to the same finger are also related by a certain measure of proximity, e.g., they cannot be too far from each other, nor their directions can be too dissimilar, provided that a main direction can be identified for each of the two clusters. In particular, we model the clusters and their similarity as an undirected weighted graph. Our aim is to recognize on that graph five paths, each going through a sequence of clusters belonging to the same finger, by trying to minimize penalties associated to links between unrelated clusters.

Before introducing our binary LP model, we set here some notation. In particular, let $\mathcal{C} = \{c_1, \dots, c_m\}$ be the set of clusters returned by DBSCAN. For each of them, we identify the two furthest areas with respect to the elongation of the cluster itself, which are likely to well approximate the region where other clusters belonging to the same finger might be attached. We build an undirected weighted graph $G = (V, E)$ where nodes V are couples of nodes $v_{i,1}$ and $v_{i,2}$ associated to the ending locations of each cluster c_i (the nodes of the graph in Fig. 6), and edges E, connecting nodes, taking into account a similarity measure among clusters. Over the graph G we define a linear assignment problem that assigns each segment to the corresponding finger l where $l \in L = \{1, \dots, 5\}$.

We call E_{in} the set of inner edges connecting two ending points of the same cluster (the black edges in Fig. 6), whereas E_{out} the set of edges between ending points of different clusters (outer edges). Instead of considering all the links between nodes $v_{i,k}$ and $v_{j,h}$ with $i \neq j$, the spatial distance between the ending points is computed and only the links between cluster extremities whose distance is below a certain threshold are considered, so that the size of E_{out} is considerably reduced. The set of these links is shown in red in Fig. 6. We also consider two additional nodes called source s and sink f, together with the edges connecting each ending point with the source and the sink, E_s and E_f respectively (these two sets are represented by dotted lines in Fig. 6). The sets E and V will be therefore $E = \{E_{in} \cup E_{out} \cup E_s \cup E_f\}$ and $V = \{v_{1,1}, v_{1,2}, \dots v_{m,1}, v_{m,2}\} \cup \{s, f\}$.

We define for each edge in E_{out} a cost value W which takes into account for the similarity between connected clusters. The idea is that two clusters can be considered *similar* if they are spatially close to each other and if they are aligned. $W(e)$, the cost of an edge $e \in E_{out}$ between $v_{i,k}$ and $v_{j,h}$, is therefore:

$$W_e = d(v_{i,k}, v_{j,h}) \cdot (1 - a(c_i, c_j)) \tag{5}$$

where $d(v_{i,k}, v_{j,h})$ is the distance between the two nodes, while $a(c_i, c_j)$ is a measure of the alignment between the two clusters. The alignment is computed by considering all the lines passing through two points, the first in one cluster and the second in the other cluster, and finding the line that has the largest number of points at a small distance from it using a RANSAC approach. The ratio between the number of close points and the total number of points in the two clusters gives the alignment a. Both d and a are scaled in order to obtain weights in the range $(0, 1)$.

Let $X_{l,e}$ and $Y_{l,v}$ be binary variables such that:

$$\begin{aligned} &X_{l,e} = 1 \text{ if label } l \text{ is assigned to edge } e, 0 \text{ otherwise} \\ &Y_{l,v} = 1 \text{ if label } l \text{ is assigned to vertex } v, 0 \text{ otherwise} \end{aligned} \tag{6}$$

the binary LP formulation can be stated as the minimization of:

$$\sum_{l \in L} \sum_{e \in E_{out}} W_e X_{l,e} \tag{7}$$

with the following constraints:

$$\sum_{l \in L} X_{l,e} \leq 1 \quad \forall e \in E \tag{8a}$$

$$\sum_{l \in L} X_{l,e} = 1 \quad \forall e \in E_{in} \tag{8b}$$

$$\sum_{l \in L} Y_{l,v} = 1 \quad \forall v \in V \setminus \{s, f\} \tag{8c}$$

$$\sum_{e \in star(s)} X_{l,e} = 1 \quad \forall l \in L \tag{8d}$$

$$\sum_{e \in star(f)} X_{l,e} = 1 \quad \forall l \in L \tag{8e}$$

$$\sum_{e \in star(v)} X_{l,e} - 2Y_{l,v} = 0 \quad \forall v \in V \setminus \{s, f\},\ l \in L \tag{8f}$$

where the constraint (8a) assigns at most one label to each edge, (8b) assigns exactly one label to each inner edge, (8c) assigns exactly one label to each vertex, except for the source and the sink, (8d) ensures that for each label l, there exists exactly one edge incident to the source which is assigned to label l, the same holds for the sink (8e). Finally, equation (8f) ensures that for each vertex, except for the source and the sink, there are exactly two edges incident to the vertex whose labels are the same as the label associated to the vertex. Moreover, while solving the linear program, specific constraints may be added in order to prevent the existence in the solution of cycles of edges having the same label (rare outcome during tests). Fig. 5b and Fig. 5c show the result of this linear assignment problem with the fingers clusters in Fig. 5a as input.

Finally notice how the output of the proposed approach can be used for a first identification of the various fingers (i.e., tell which segment is the thumb, index and so on) by simply assigning a set of labels to the clusters based on the angle between the vector connecting the hand center to the cluster centroid and a reference direction. This requires that the 5 fingers are visible in the acquired depth map and can fail in some complex configurations with fingers bent one over the other (e.g., gesture G8 in Fig. 10). A more refined recognition scheme will be subject of future work.

IV. Experimental Results

In order to evaluate the performance of the proposed approach, we acquired a dataset of gestures using a Creative SENZ3D depth camera. The dataset contains 16 different gestures repeated 20 times for a total of 320 different acquisitions from 4 different users. Color data has also been acquired but notice that the proposed approach does not use color information. A sample image and depth map for each of the gestures is shown in Fig. 7. The hands have been acquired at a distance of around 30 [cm], with an hand

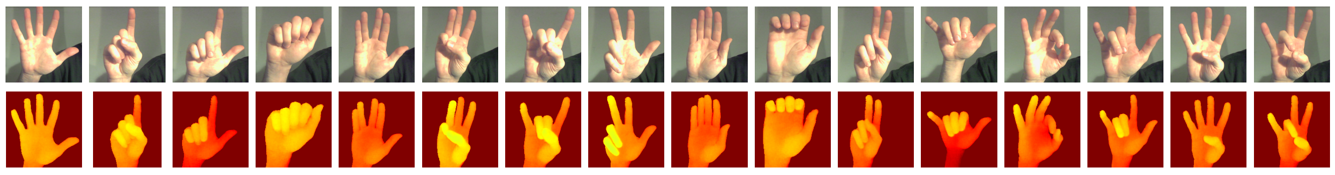

Fig. 7. Sample images and depth maps for each of the gestures in the database.

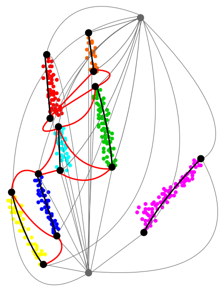

Fig. 6. Sample graph for the LP optimization. The *black* edges are the inner edges connecting the two ending points of each cluster, while the *red* ones are the outer edges. The edges coming out from the source and the sink are depicted in *gray*. (*Best viewed in colors.*)

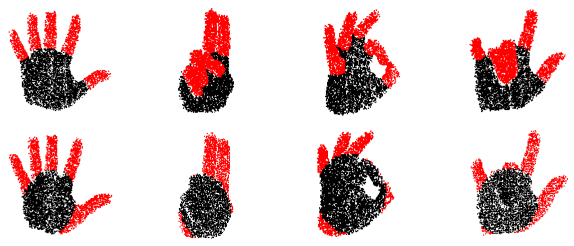

Fig. 8. Palm and finger segmentation: (*first row*) Our approach; (*second row*) Dominio et Al [3]

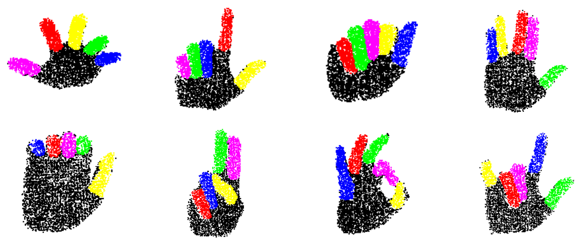

Fig. 9. Output of the proposed approach on some sample gestures: (from the left) G1, G3, G4, G5, G10, G11, G13, G14. (*Best viewed in colors*).

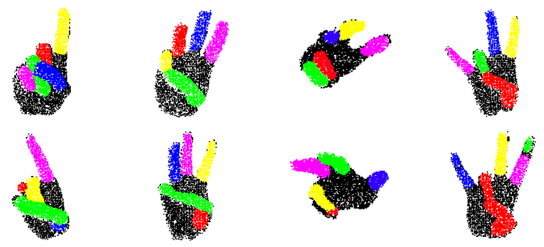

Fig. 10. Examples of the most challenging configurations for the proposed approach: (columns, from the left) gestures G2, G6, G8, G16; (*first row*) correct recognitions; (*second row*) classifications with issues. (*Best viewed in colors*).

typically covering an area of around 150×150 [pixels] on average.

Let us start from the palm detection scheme of Section III-C. The palm region is almost always correctly assigned, resulting in very accurate detections except for some rare cases in which the portion of the palm close to the pinkie is assigned to the fingers region. Some examples and a comparison with [3] are shown in Fig. 8, notice how fingers bent over the palm are correctly recognized, differently from approaches like [3] or [11] that can not handle this situation.

The accuracy of the proposed finger identification approach is reported in Table I, while Figures 9 and 10 show some visual examples of the output of the proposed algorithm (i.e., palm detection and fingers identification). The table shows in the first column the average accuracy of the proposed approach for fingers identification on the considered dataset (i.e., the percentage of identified fingers). The other three columns report the type of errors that led to the wrong identification, i.e., if a finger is not detected, if it is split in two or more parts or if multiple fingers are joined together. Notice that since multiple errors can be made on the same fingers, in some case the sum of the percentages in a row can be greater than 100%.

The fingers segmentation is a more challenging problem. As expected, the proposed algorithm correctly handles the simplest situations, e.g., the open hand (G1 and G15), as it is shown in the first example of Fig. 9. Also a bit more complex situations where the fingers are closer each other but not occluded are perfectly handled, e.g., gestures G3, G12 and G14. In gestures where the fingers are more close together, e.g. G5, G9 or G10 some errors start to appear but the recognition rate is still very high (typically between 95% and 98%). Some examples of the performance in these situations are also represented in Fig. 9. Another challenging situation is when the fingertips are touching each other, e.g., gesture G6, G13 and G16. Also here the recognition rate is very good (the most critical gesture is G13 with a recognition rate of 91%). Notice that on these gestures the most common error is the joining of the two touching fingers, Fig. 10 shows a couple of the few cases in which the algorithm fails to handle this situation. Finally, some fingers (typically the thumb) can occlude others making more challenging the identification of the covered fingers. The approach is quite solid also with this issue. Notice that the number of recognized fingers is more than 90% on all the considered gestures.

Finally, even if the considered gesture is mostly frontal, the proposed approach is able to work also when the hand is inclined in various directions with relatively large inclinations

Gesture	Correctly recognized	Missing Fingers	Joined Fingers	Split Fingers
G1	100%	0%	0 %	0%
G2	90%	0%	10 %	5%
G3	100%	0%	0 %	0%
G4	94%	2%	2 %	3%
G5	95%	4%	0 %	1%
G6	92%	0%	8 %	4%
G7	94%	1%	2 %	3%
G8	100%	0%	0 %	0%
G9	98%	2%	0 %	0%
G10	96%	1%	2 %	2%
G11	95%	0%	4 %	1%
G12	100%	0%	0 %	0%
G13	91%	1%	8 %	4%
G14	100%	0%	0 %	0%
G15	100%	0%	0 %	0%
G16	97%	0%	2 %	1%
AVERAGE	96.4%	0.7%	2.4%	1.5%

TABLE I

ACCURACY OF THE PROPOSED APPROACH IN TERMS OF PERCENTAGE OF CORRECTLY RECOGNIZED FINGERS.

(an example is shown in Fig. 10). In particular, working with the point cloud in the 3D space and not in the 2D domain makes our algorithm invariant to the hand orientation w.r.t. the camera. Only when the hand is almost sideways errors start to appear, mostly due to fingers that cannot be seen.

On average, the algorithm correctly identifies more than 96% of the fingers. The most common error is when the fingers classification step collapses more than one finger together (about 2.4% of the fingers), mostly happening when the fingertips are touching each other or the fingers are bent one over the other. Notice that the proposed approach does not use skeletal information and can only rely on the normal guided contraction followed by segmentation to separate the fingers. In some situations (only 1.5% of the times) a finger can be split in two parts, which happens due to occlusions or to particular poses of the thumb. Also, a finger can be missed by the algorithm because it is too occluded, but this happens very seldom, only in 0.7% of the cases.

Concerning the computational complexity, the proposed approach is not particularly CPU demanding. Very fast normal computation approaches already exist and after normal extraction, the contraction is a simple operation that can be executed in real-time. Palm and fingers segmentation do not require complex operations and the operations of Eq. (2) and Eq. (4) are easily parallelizable. The integer programming part can be done in real-time: on a 2.5GHz i5 CPU, a solution was found on average within 14 ms. The current implementation is not optimized and exploits Matlab code for some steps but an optimized C++ application will be able to do all the processing in real-time.

V. Conclusions

An efficient approach for the recognition of the palm and fingers from a single depth map without exploiting temporal constraints has been proposed. The palm and fingers regions are discriminated by contracting the 3D point cloud along the normal directions and analyzing the changes in the points density due to this process. This allows to recognize also fingers bent over the palm, a quite critical issue for many silhouette and shape-based approaches. Density-based clustering is then used to perform an over-segmentation of the fingers regions and finally the segments are associated to the various fingers by an integer linear programming approach. Experimental results demonstrate the effectiveness of the approach on a challenging dataset containing complex gestures with inter-occlusions and fingers bent over the palm and over other fingers. Notice how after recognizing the various fingers it is much simpler to estimate the hand pose by associating the fingers depth samples to the hand skeleton. Further research will be devoted to the exploitation of the proposed approach for hand pose estimation.

References

[1] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo. *Time-of-Flight Cameras and Microsoft Kinect.* SpringerBriefs in Electrical and Computer Engineering. Springer, 2012.

[2] F. Dominio, M. Donadeo, G. Marin, P. Zanuttigh, and G.M. Cortelazzo. Hand gesture recognition with depth data. In *Proceedings of the 4th ACM/IEEE ARTEMIS workshop*, pages 9–16. ACM, 2013.

[3] F. Dominio, M. Donadeo, and P. Zanuttigh. Combining multiple depth-based descriptors for hand gesture recognition. *Pattern Recognition Letters*, pages 101–111, 2014.

[4] A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(12):52 – 73, 2007.

[5] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD*, pages 226–231. AAAI Press, 1996.

[6] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[7] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

[8] Natalia Neverova, Christian Wolf, Graham W. Taylor, and Florian Nebout. Hand segmentation with structured convolutional learning. In *The 12th Asian Conference on Computer Vision (ACCV)*, 2014.

[9] I. Oikonomidis, M. Lourakis, and A.A. Argyros. Evolutionary quasi-random search for hand articulations tracking. In *Proc. of CVPR*, 2014.

[10] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *Proceedings of CVPR*, 2014.

[11] Z. Ren, J. Yuan, C. Li, and W. Liu. Minimum near-convex decomposition for robust shape representation. In *Proceedings of ICCV*, pages 303–310, 2011.

[12] D. Tang, H.J. Chang, A. Tejani, and T. Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proceedings of CVPR*, 2014.

[13] D. Tang, T.H. Yu, and T.K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proc. of ICCV*, 2013.

[14] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. Fitzgibbon. User-specific hand modeling from monocular depth sequences. *Proc. of CVPR*, 2014.

[15] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of ICCV*, pages 839–846, 1998.

[16] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Graph.*, 33(5):169:1–169:10, 2014.