# Real-time sign language recognition using a consumer depth camera

Alina Kuznetsova, Laura Leal-Taixé, Bodo Rosenhahn
Institute fuer Informationsverarbeitung, Leibniz University Hannover
Appelstr. 9A, Hannover, 30167, Germany
{kuznetso,leal,rosenhahn}@tnt.uni-hannover.de

## Abstract

*Gesture recognition remains a very challenging task in the field of computer vision and human computer interaction (HCI). A decade ago the task seemed to be almost unsolvable with the data provided by a single RGB camera. Due to recent advances in sensing technologies, such as time-of-flight and structured light cameras, there are new data sources available, which make hand gesture recognition more feasible. In this work, we propose a highly precise method to recognize static gestures from a depth data, provided from one of the above mentioned devices.*

*The depth images are used to derive rotation-, translation- and scale- invariant features. A multi-layered random forest (MLRF) is then trained to classify the feature vectors, which yields to the recognition of the hand signs. The training time and memory required by MLRF are much smaller, compared to a simple random forest with equivalent precision. This allows to repeat the training procedure of MLRF without significant effort.*

*To show the advantages of our technique, we evaluate our algorithm on synthetic data, on publicly available dataset, containing 24 signs from American Sign Language(ASL) and on a new dataset, collected using recently appeared Intel Creative Gesture Camera.*

## 1. Introduction

Sign language and gesture recognition is an important problem in computer vision and machine learning and a fair amount of research is done in this area. Due to advances in sensing technologies, there is a rapid progress in the robustness and quality of the solutions. Lately, a lot of work in static and dynamic gesture and pose recognition appeared. Most of the approaches for sign language recognition can be roughly divided into two groups:

- pose estimation is performed and the parameters of the pose are used to determine a gesture;
- gesture recognition is performed directly on raw image data or on image features.

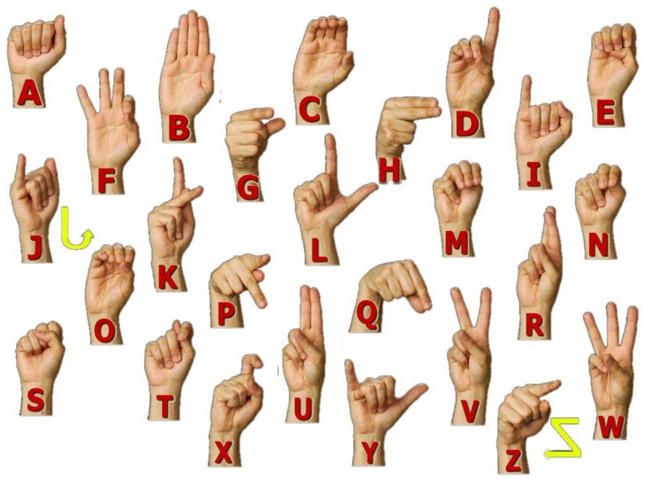

Figure 1: Signs of the ASL sign language [1]; some signs are very difficult to differentiate, for example, signs for the letters ’m’ and ’n’, or for the letters ’s’ and ’t’.

Since the initial hand pose estimation task appears to be even more complicated than the task of gesture recognition itself, more works can be found in the second group. The current state-of-the-art methods are outlined in Sect. 2.

The problem of sign language and gesture recognition can be divided into recognition of the static gestures and of the dynamic gestures, e.g. the American Sign Language (ASL) contains both static and dynamic gestures.

The main challenges of the static sign recognition problem are:

1. the number of signs (there are, for example, 24 static signs in ASL (Fig. 1), or 26 static signs in Russian sign language);
2. similarity between some signs (see Fig. 1 for an example);
3. variation in appearance of a sign due to difference in viewpoint (intra-subject variation);

4. variation in performance of a sign by different subjects (inter-subject variation);

Apart from that, there is a need to distinguish a static sign from a dynamic sign or an accidental hand posture.

In our work, we propose an approach for the static sign recognition and offer the solution for the above mentioned challenges. We mainly concentrate on the first three challenges, i.e. we recognize the full set of signs, performed by a single subject.

## 2. Related work

Gesture recognition with a single camera is a very challenging problem, remaining unsolved for a long time.

Early works used different kind of image convolution to form feature vectors based on a single RGB image of a hand. In [13], the authors use wavelelet families, computed on edge images, as features to train a neural network for 24 sign classification. Haarlet-like features, computed on gray-scale images and on silhouettes were used in [10] for classification of 10 hand shapes. Principle Component Analysis (PCA) was applied directly on images to derive a subspace of hand poses, which is then used to classify the hand poses (see [9]). In [19], a modification of HOG descriptors is employed to recognize static signs of the British Sign Language. SIFT-feature based description was used to recognize signs of ASL in [17]. All these methods depend heavily on the lighting conditions, subject's appearance and background. Additionally, pose normalization is required before feature computation, since the features mentioned above are not rotation-, position- and scale- invariant.

Due to appearance of range sensors there have been several breakthroughs in the area of recognition using a single camera.

The most well-known advances are in human pose recognition using the Microsoft Kinect sensor [22]. The authors used a special kind of features, computed directly on depth images, to classify human body parts. The same features were lately applied for the problem of hand pose estimation [15] and shape classification [16].

Hand detection and segmentation is a much easier task in case of availability of the depth data, therefore a number of methods appeared relying solely on segmentation derived from depth data, such as contour-based method, described in [21], and rule-based method, presented in [7].

Additionally, since depth data is robust to illumination and subject appearance changes, a number of methods appeared on calculating image features directly on depth images [20], [23]. Obviously, these methods still do not account for different viewpoints.

One more conventional approach for hand gesture estimation relies on a 3D model, maintained in memory, and fitted to an image [18]. The approach of comparing the rendered model in some configuration with the original image was successfully applied both for sign recognition and for pose estimation [12]. These methods naturally account for different hand position and rotation, but the variation in rotation enlarges parameter space and therefore makes the matching procedure less stable and more time-consuming.

Many of the works focus on recognition of a small subset of signs. As already mentioned, for most of them difference in viewpoint (or alternatively, different rotation, position and scaling of a hand), environment and subject appearance, represents significant difficulty.

In our work, we address the problems mentioned above by using rotation-, position- and scaling- invariant features. Additionally, in the fashion of [16], we apply a multi-layered random forest (MLRF) for classification. This allows us to significantly reduce the training time for MLRF and the memory consumption, which implies the possibility of retraining the forest automatically in a very short times using a home personal computer.

We evaluate our algorithm on synthetic data to show the robustness of the features against pose variations, on a public dataset from [20] for comparison to the stat-of-the-art and on the data we collected using Intel Creative Gesture Camera [2] to show the applicability of the method to a different kind of sensor.

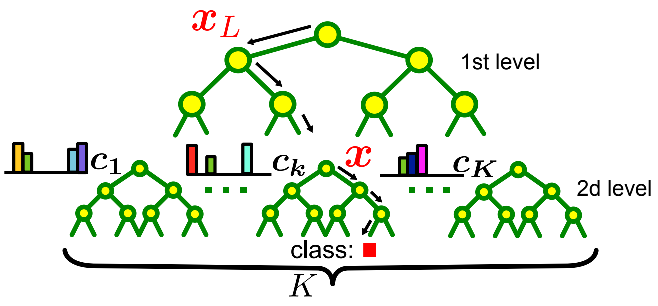

Figure 2: Multi-layered forest structure; on the level one, the forest determines the cluster of a vector; on the level two, the final class label is assigned by the forest, corresponding to the cluster label assigned on the first level.

### 2.1. Contributions

As mentioned in Sect. 2, we propose to improve sign language classification using two main ideas.

Firstly, since a hand sign can have very different appearance when seen from a different view (see Fig. 3), we propose to use rotation-, translation- and scaling- invariant image features to reduce intra-class variation. Such features exist for 3D point clouds, which can be derived from the depth data, delivered by a depth sensor. These features can be separated into two categories — local and global features. The overview of the state-of-the-art features for object recognition is presented, for example, in [4]. We chose

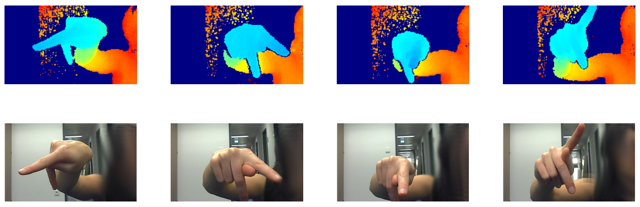
Figure 3: Difference in appearance of the sign 'P' depending on the view (the new dataset).

to use ensemble of shape function (ESF) descriptor [24], since it is translation-, rotation- and scale- invariant, and can be computed in real time.

Secondly, we propose to use multi-layered random forest (MLRF) for classification (Fig. 2). The intuition behind this approach is to find groups of similar feature descriptors and for each group train a separate classifier that can better distinguish the classes withing this group. Ideally, each group contains less classes then the original set.

There are several advantages of using MLRF:

- **Potentially higher accuracy** Due to the usage of two levels of forest, averaging happens between the trees on the first level, and therefore the error already made in some trees is not propagated to the second level. The second-level random forests are trained on clusters containing much less variation then the initial training set, and therefore a smaller forest is required to make classification robust.

- **Smaller training time and memory** Since random forest consists of binary trees, for each node the training parameters should be stored, the memory size to store the forest increases exponentially with the depth of the forest. Since MLRF contains much less nodes, the memory consumption is reduced drastically. For example, to store a forest of depth $D = 20$ with $T = 10$ trees, where each node has 5 parameters, the minimum amount of memory required is 560 Mb, and a forest of depth $D = 25$ requires over 18 Gb memory. A MLRF having 10 forests on the second level, and the depth of the forests on the both level equal to 10, requires around 6 Mb of memory, and to store a MLRF with depths $D_1 = 13$ and $D_2 = 12$ only 26.5 Mb is needed.

The proposed method is device independent, since it does not exploit any characteristics of a particular depth sensor, and therefore can be applied both in case of Kinect device, as well as in case of a time-of-flight camera (we used Intel Creative Gesture Camera in our experiments).

## 3. Feature extraction

**Depth image preprocessing** The hand is detected and segmented by thresholding depth values, and a point cloud for further processing is obtained.

To normalize the depth image and transform it to the real 3D point cloud, we firstly transform depth image to a point cloud using inverse perspective transformation:

$$x_{3D} = \frac{x_{2D} - c_x}{f_x} I(x_{2D}, y_{2D}) \quad (1)$$

$$y_{3D} = \frac{y_{2D} - c_y}{f_y} I(x_{2D}, y_{2D}) \quad (2)$$

$$z_{3D} = I(x_{2D}, y_{2D}) \quad (3)$$

where $(c_x, c_y, f_x, f_y)$ are intrinsic parameters of a depth sensor. In case of Kinect sensor, we use the default values, provided by OpenNI library [3].

**ESF descriptor** ESF descriptor consists of a set of histograms, concatenated together. The first histogram describes the distribution of distances between two random points in the point cloud (function D2). The second histogram describes the distribution of angles enclosed by two lines created by randomly sampling three points from a point cloud (function A3) The third histogram describes the distribution of areas enclosed by three randomly sampled points (function D3).

Additionally for each line, connecting two points, it is determined, if it is on the point cloud surface, off the surface, or intersects the surface. In each case, a separate histogram is built for each function. For the function D2, a histogram containing the length ratio in case of line intersection is additionally computed. Each histogram has length 64, and therefore the whole descriptor, consisting of the 10 histograms, has length 640.

This descriptor has the following features:

- given that the whole object is visible, it is rotation- and translation- invariant; additionally, the descriptor is also scale invariant, since the range of possible values of D2, D3 and A3 is normalized before computing the histograms;
- invariant under mirror transformation;
- it does not require unstable and time consuming normal computation;
- it is reported to be fairly discriminative [4, 24].

The hand has fairly different appearance depending on the gesture, so we treat the sign classification problem as the object classification problem. We compute one ESF descriptor for the segmented hand region. Examples of ESF descriptors for different classes are given in Fig. 4.

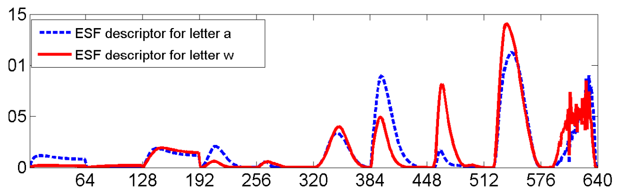

Figure 4: ESF descriptors for the letters 'a' and 'w'.

## 4. The multi-layered random forest (MLRF)

**Random forest** Random forest (RF) is a classification and regression technique [5, 8], that has become popular lately due to its efficiency and simplicity.

Random forest consists of $T$ decision trees $t_i$. Each decision tree is normally a binary tree. Data is propagated through each tree starting from the root of a tree. At each node, the data is split using a binary test in the form $f(\boldsymbol{x}) < \tau$, where $f$ and $\tau$ are determined from the training data. If the inequation holds, the data is sent to the left branch, otherwise — to the right branch. All trees have the same depth $D$.

At each leaf node of a tree a predictor is stored. In our case, it is a probability distribution of class (or cluster) labels, that reached this node on the training stage. Whenever a data vector $\boldsymbol{x}$ reaches the node $k$ of the $m$-th tree, the prediction of the tree $m$ for this vector is formed by the predictor stored in the leaf $k$.

The prediction of the whole RF $t$ for the $\boldsymbol{x}$ is usually formed by averaging the predictions of the separate trees:

$$t(\boldsymbol{x}) = \frac{1}{T}\sum_i t_i(\boldsymbol{x}), \tag{4}$$

where $T$ is the number of trees in the RF.

During training, as the training data is propagated down the tree, at each node several suggestions for $f$ and $\tau$ are sampled randomly, and the best parameters are selected according to some criteria (we generate 5 split suggestions per node during training). In this work, we used information gain as such criteria.

As $f$ function, we chose to use two dimensional separation plane, where two coordinates are randomly sampled from the feature vector: $f(\boldsymbol{x}) = x_{i_1}w_1 + x_{i_2}w_2$. Therefore, each node of a decision tree is characterized by 5 parameters in total.

Prior to applying random forest for training, we normalize the feature vectors, so that their distribution has the mean value 0 and the variance 1.

**Clustering the data** Recently several works appeared on creating multi-layered random forest. For example, in [16] the authors use spectral clustering to pre-cluster data and then use multi-layered random forest to perform hand parts regression.

Experiments showed that RF performs better in distinguishing between smaller number of classes, so we pre-clustered the data to improve classification accuracy. Our goal in this case is to produce clusters that contain much less classes then the whole training set. We opted to use RF-based clustering technique, since in our experiments spectral clustering failed to produce such clusters.

Clustering itself essentially means finding structures in the data, which is equivalent to distinguishing between the given data and random data having the same range of values. Therefore, we create an artificial training dataset, containing all the data we want to cluster as one class, and the randomly generated data in another class. This data is generated by randomly sampling values from each marginal distribution of the vector coordinates of the training data.

One of the random forest features is that in some sense similar samples tend to fall within the same node during training. This can be characterized by so-called proximity matrix $P$ of size $N \times N$, where $N$ is the size of training set.

$$P_{ij} = P_{ji} = \frac{|\{m|\boldsymbol{x}_i, \boldsymbol{x}_j \in L_k^m\}|}{T}. \tag{5}$$

Here $L_k^m$ is the set of all samples $\boldsymbol{x}$, that end up in the node $k$ for the decision tree $m$.

The proximity matrix $P$ can be regarded as a similarity measurement and therefore can be used as an input for a clustering algorithm. We transform the proximity matrix to a distance measure $m_P(i,j) = \sqrt{1 - P_{ij}}$.

It can be seen, that this is a non-Euclidean metric, and therefore a suitable clustering algorithm is required.

We chose to use hierarchical clustering [11] to cluster the data based on the produced similarity metrics. For hierarchical clustering, we use complete linkage to aggregate data into a cluster tree.

We use less bins in ESF descriptor for clustering, because it both allows to speed-up the training and reduces the noise in the computed descriptors. According to our experiments, more coarse histograms already contain enough information to find similar signs, in the same time containing much less noise. We define $L$ as the histogram aggregation level parameter, where $L = 0$ defines the full 64-bins histograms, $L = 1$ reduces the size of histogram to 32 bins, level $L = 2$ produces 16 bins, etc.

Therefore, the overall clustering algorithm has the following steps:

1. compute the features at a given aggregation level $L$;

2. create two artificial clusters — one containing all training data, one containing randomly sampled data;

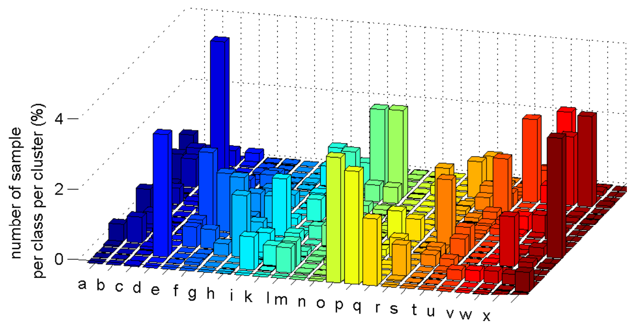

Figure 5: Data clustering results for the [20] dataset using hierarchical clustering with $K = 20$ clusters; the horizontal axis show the classes (letters) and the clusters, the vectical axes shows the number of samples in each cluster, corresponding to the current class.

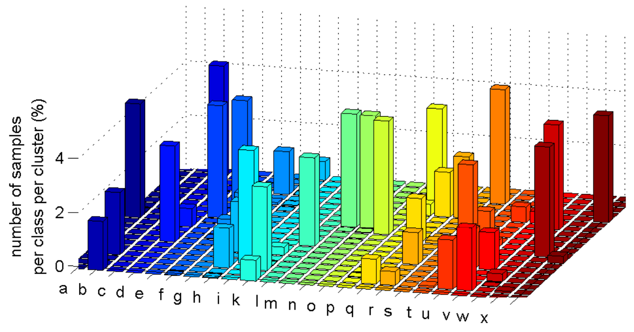

Figure 6: Data clustering results for the synthetic data using hierarchical clustering with $K = 20$ clusters; the horizontal axis show the classes (letters) and the clusters, the vectical axes shows the number of samples in each cluster, corresponding to the current class.

3. train a forest to differentiate between the two classes and compute $P$ for this forest;

4. apply hierarchical clustering algorithm to find clusters in the data.

Examples of clustering results are shown in Fig. 5 and Fig. 6. It can be seen, that the clustering actually produces meaningful results, i.e. similar signs are grouped together in clusters and for each cluster the number of classes inside it is reduced. For example, for real data the signs 'o','p','q' are grouped together in one cluster, and the signs 'm' and 'n' are grouped into another cluster; as shown in Fig. 7, the depth images of these signs are quite similar and so are the descriptors.

As expected, artificial data clustering separates the classes almost perfectly, producing clusters, containing only one class, while real data contains much more noise and therefore the clusters contain small portions of different classes (Fig. 5).

**Training** The multi-layered random forest (MLRF) is a random forest, consisting of two layers.

To train the MLRF, we first perform clustering of the data. Initially we set the number of clusters to $K = 20$. As we showed earlier, the clusters build from the real data are noisy and contain small portions of data from many classes. Our goal is to create clusters, containing small number of classes, therefore we prune clusters by:

- merging small clusters to the bigger clusters; we set the minimum cluster size to 50% of the size of a class in the training data; if a cluster is smaller than this size, for each element of this cluster we find a cluster, containing the most elements of the same class, and merge this element to the new cluster; due to this threshold the number of clusters is in fact determined automatically for each dataset. We could observe, that in case of ASL signs classification problem, $12 - 15$ big clusters are formed.

- cleaning big clusters by removing classes, that have too small number of representative in a cluster.

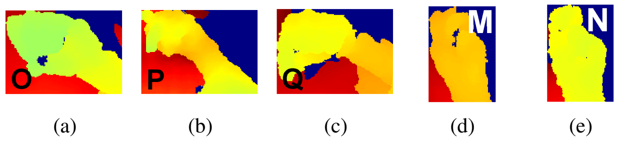

Figure 7: The depth images of the signs, forming two different clusters in the clustering, depictured in Fig. 5; images 7a,7b,7c come from one cluster, 7d,7e — from another cluster.

After the data is clustered, we train the first level random forest on the aggregated feature vectors. This forest assigns a cluster label to each incoming vector. Afterwards, for each of the derived clusters, we train a separate random forest on the full feature vectors to distinguish between similar signs.

We define the **cumulative depth** of the MLRF as the sum of depths of the RF on the first and the second layers. We also set the number of trees to be the same for each forest in the first and the second layers.

An important step is to re-normalize the feature vectors for each forest on the second level. Such re-normalization is necessary, because similar vectors are clustered together, and therefore the variance withing each cluster is much lower, then the variance of the whole sample. The variance is derived by the classes that are contained in this cluster, and skipping re-normalization can decrease classification performance.

**Testing** During the testing phase, each sample is passed through the first-level forest to determine, from which class the sample comes from. The first-level forest determines the cluster label of a sample. Afterwards, the sample is passed to the corresponding forest on the second level to determine its class label.

## 5. Evaluation

We evaluate our algorithm in different settings on synthetic data, publicly available database, containing ASL signs [20] and the depth data collected from the Intel Creative Depth Camera. We compare the results of our method to the state-of-the-art methods [20] and [16] in terms of classification error (the number of incorrectly classified samples divided by the size of the test set), training time and memory consumption.

We use an unparallelized MATLAB random forest implementation [14]. The computational times are provided for one core CPU employed in computations.

### 5.1. Synthetic data

In a natural environment, a subject's hand position performing different signs relative to a camera can change a lot. Therefore, using the features robust to such changes increases the performance.

To demonstrate robustness of the ESF features, we generated synthetic data using the hand model from [6] to create depth images of 24 static letters of ASL. For the training set, we generated 500 images per letter, in all cases the hand was parallel to the camera. For the test set, we applied rotations around three main axis to evaluate the classification error versus the angle of rotation of the hand model.

We separate rotation around the axis X and Y, forming the camera plain, and around the camera Z axis. As shown in Fig. 8, the latter has no influence on the entire performance, since the point cloud shape itself stays exactly the same. Stronger rotation around X and Y axis causes stronger performance decrease, since it changes point cloud appearance (see Fig. 3). This result shows, that to be able to robustly recognize gestures from a different viewpoint in general the corresponding training data is still needed. However, the rotation of a hand within the limits of 10 degrees does not degrade classification performance significantly.

To justify the usage of the multi-layered RF, we evaluated the training time of a conventional RF vs. the MLRF with the same cumulative depth (see Sect. 4). As can be seen, the multi-layered RF training is around 5 times faster then a simple RF (Fig. 9).

Note, that the classification error for both forests is the same (Fig. 10) starting from the depth $D = 20$. The classification error for the MLRF is a little higher for the smaller depth. Our experiments showed that having depth less then

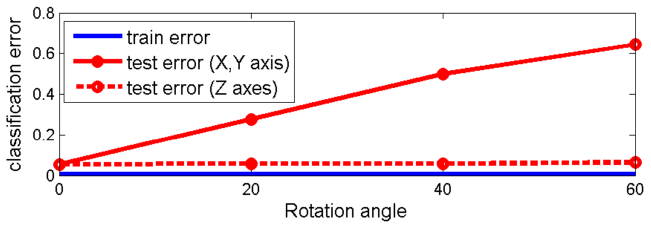

Figure 8: Train and test error depending on rotation angle limits of the test sample; the forest used has the number of trees $T = 10$ with depth $D = 20$.

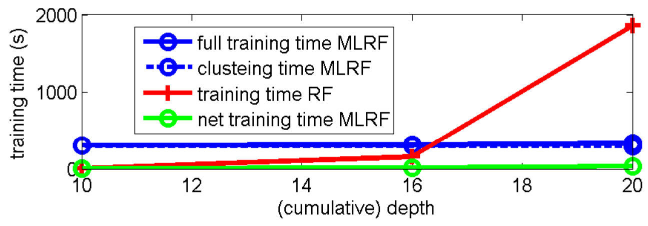

Figure 9: Training time (s) needed for a simple RF and for the multi-layered RF with $T = 10$ depending on the forest depth (for the multi-layered forest, depth of the first layer is $D/2$, as well as the depth of the second layer).

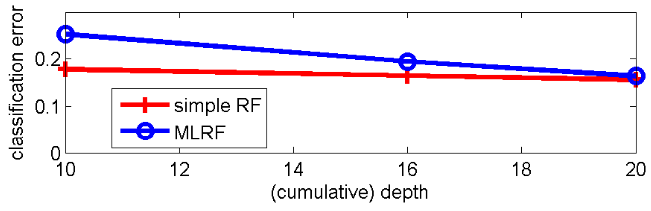

Figure 10: Classification error depending on the depth of the forests.

10 makes the first-level forest too weak to classify the data into the correct cluster.

We also evaluated the MLRF performance depending on the separation between two layers. The optimal separation between two layers of forest in depth is approximately $D/2$ (Fig. 11).

### 5.2. Real data

**The public dataset [20]** The dataset contains 24 static signs from American sign language (ASL), performed by 5 subjects. There is high variability in how the people perform signs and in the relative position to the camera, although the variability in pose is relatively low for one subject.

To evaluate how good our method generalizes, we trained the forests on 4 subjects and then evaluated the performance on the subject, left out of the training. In Tab.

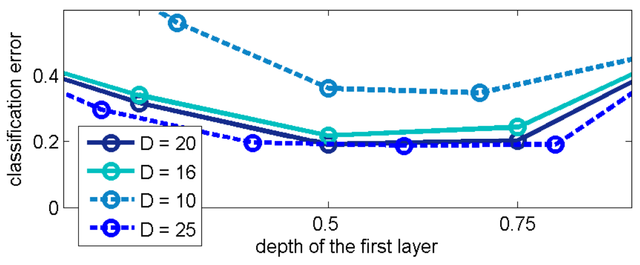

Figure 11: Classification error depending on the separation in depth between the first and the second layers of the forest ($T = 10$).

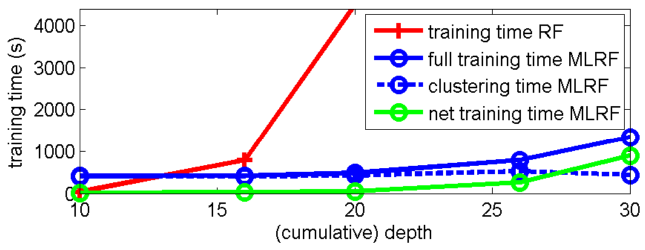

Figure 12: Training time depending on the depth of the forests.

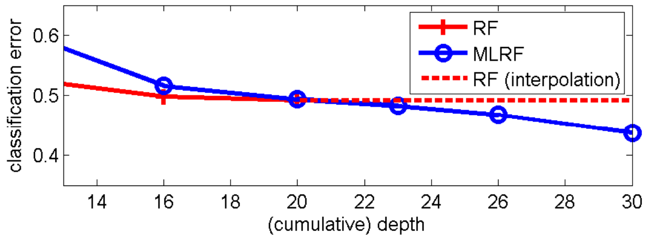

Figure 13: Classification error depending on the depth of the forests.

1 this is denoted as l-o-o experiment. The reported on the web-site baseline error for the database is 51% for the depth data. As can be seen, the training error in these experiments is decreased by almost 10%. Additionally, in case of using half of the data for testing and half for training (h-h experiment), we were able to decrease the error to 15%.

To justify the usage of the MLRF, we compare its performance to the performance of a standard random forest. We vary the depth of the random forest and compare the classification error of the simple RF with the MLRF and measure the training time (Fig. 12 and Fig. 13). Since the database overall size is around 65000 of images, the MLRF achives even more win in training time — it is around 10 times faster then the simple RF for the depth $D = 20$.

In [16], the performance of the shape classification for the [20] dataset is presented. In the l-o-o experiment, the

| method | h-h | l-o-o | time(s) | mem |
|---|---|---|---|---|
| GF + RF [20] | 31% | 51% | — | — |
| RF ($D = 20$) | 15% | 49.1% | 4485 | 560 |
| MLRF ($D = 20$) | 23.4% | 50.1% | 522 | 6.05 |
| MLRF ($D = 30$) | **13%** | **43%** | 1132.3 | 192.5 |

Table 1: Performance comparison on the [20] database (in all forests, $T = 10$) in terms of the error for h-h and l-o-o scenarios; memory consumption is given in Mb.

| | dataset [20] | Intel Camera data |
|---|---|---|
| error $D = 20$ | 6% | 22% |
| time (s)$D = 20$ | 212 | 68.77 |
| error $D = 30$ | 2.45% | 15.3% |
| time (s)$D = 30$ | 821 | 476 |

Table 2: Performance evaluation (MLRF) for one subject on the data from [20] and on the data, collected with Intel Gesture Camera.

achieved accuracy is 85% vs. 57% of our method and in the h-h experiment, the reported performance is 97.8%. However, the reported training time is around 4000 s per tree on a quad core PC. For our method, the training time for an unoptimized MATLAB version of the MLRF is less then one thousand of seconds on one core. In one-subject experiment, our method shows the accuracy of 97.4% (see Tab. 2). Therefore, in one-subject experiment our method has the same performance as the method in [16], while the training times allows to re-calibrate the system for a new subject very fast.

**Evaluation on the new dataset** To illustrate applicability of the recognition method to other sensors, we collected a dataset using Intel Creative Gesture Camera [2]. The device represents a low-cost time-of-flight camera with the range from 10cm to 1m, maximal frame rate of 50fps and the resolution of $240 \times 320$ pixels (for depth data). High speed and near range make the camera potentially useful in the field of gesture recognition.

The dataset contains 3 subjects performing 24 signs from the ASL sign language. For each letter, we collected around 250 data frames. The purpose of this dataset is to account for natural variation in a hand pose relative to the camera, and therefore the participants were asked to rotate the hand relative to the camera by 30 degrees in all directions to create more variability (see Fig 3 for an example).

Since the data has lower resolution ($240 \times 320$ vs. $480 \times 640$ pixels for the Kinect), we expect some recognition performance decrease. Indeed, the results show (see Tab. 2) some performance decrease, which we explain by low camera resolution and high variance in hand pose.

## 6. Conclusion

In this work, we proposed the usage of multi-layered random forest for multi-class classification for a problem with a large number of classes, where different classes have different extent of similarity. The main advantages of the usage of the MLRF are the **low training times** and **low memory consumption** given the same or slightly better accuracy.

We also proposed the usage of translation-,rotation- and scale- invariant features for gesture recognition, that improved the results of the recognition significantly, even in case of hand pose changes. Our experiments showed, that these features have in general better discriminating properties compared to the features presented in [20], however, they still lack generalization properties.

We evaluate the performance of our method depending on the key parameters and compare it with the state-of-the-art methods. Our method demonstrates low training time, low memory usage and high accuracy in one subject tests, which make it applicable in a scenario, when fast system re-calibration is acceptable.

## References

[1] Fingerspelled alphabet. http://lifeprint.com/. 1

[2] Intel creative gesture camera. http://click.intel.com/intelsdk/Default.aspx. 2, 7

[3] Openni. http://www.openni.org/. 3

[4] L. A. Alexandre. 3D descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, oct 2012. 2, 3

[5] Y. Amit and D. G. Y. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997. 4

[6] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision (ECCV)*, pages 640–653, Firenze, October 2012. 6

[7] L. Billiet, J. A. Oramas Mogrovejo, M. Hoffmann, W. Meert, and L. Antanas. Rule-based hand posture recognition using qualitative finger configurations acquired with the kinect. In *Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods*, pages 1–4, Feb. 2013. 2

[8] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001. 4

[9] N. Dardas and E. Petriu. Hand gesture detection and recognition using principal component analysis. In *Computational Intelligence for Measurement Systems and Applications (CIMSA), 2011 IEEE International Conference on*, pages 1–6, 2011. 2

[10] M. V. den Bergh, F. Bosche, E. Koller-Meier, , and L. V. Gool. Haarlet-based hand gesture recognition for 3d interaction. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV 2009)*, December 2009. 2

[11] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003. 4

[12] N. K. Iason Oikonomidis and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.101. 2

[13] J. Isaacs and S. Foo. Hand pose estimation for american sign language recognition. In *System Theory, 2004. Proceedings of the Thirty-Sixth Southeastern Symposium on*, pages 132–136, 2004. 2

[14] A. Karpathy. Machine learning matlab toolbox. https://github.com/karpathy/Random-Forest-Matlab. 6

[15] C. Keskin, F. Kira, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *ICCV Workshops*, pages 1228–1234. IEEE, 2011. 2

[16] C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proceedings of the 12th European conference on Computer Vision - Volume Part VI*, ECCV'12, pages 852–863, Berlin, Heidelberg, 2012. Springer-Verlag. 2, 4, 6, 7

[17] T. Kim, K. Livescu, and G. Shakhnarovich. American sign language fingerspelling recognition with phonological feature-based tandem models. In *SLT*, pages 119–124. IEEE, 2012. 2

[18] A. Kuznetsova and B. Rosenhahn. Hand pose estimation from a single rgb-d image. In *Proceedings of the 9th International Symposium on Visual Computing*, ISVC'13. Springer-Verlag, 2013. 2

[19] S. Liwicki and M. Everingham. Automatic recognition of fingerspelled words in british sign language. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 0:50–57, 2009. 2

[20] N. Pugeault and R. Bowden. Spelling it out: Real-time asl fingerspelling recognition. In *ICCV Workshops*, pages 1114–1119, 2011. 2, 5, 6, 7, 8

[21] Z. Ren, J. Yuan, and Z. Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In K. S. Candan, S. Panchanathan, B. Prabhakaran, H. Sundaram, W. chi Feng, and N. Sebe, editors, *ACM Multimedia*, pages 1093–1096. ACM, 2011. 2

[22] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society. 2

[23] M. Van den Bergh and L. Van Gool. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In *Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision (WACV)*, WACV '11, pages 66–72, Washington, DC, USA, 2011. IEEE Computer Society. 2

[24] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *ROBIO*, pages 2987–2992. IEEE, 2011. 3