Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
January-February 2021: 1–13

DOI: 10.1177/1729881420978363
journals.sagepub.com/home/arx
SAGE

Processing chain for 3D histogram of gradients based real-time object recognition

Cristian Vilar, Silvia Krug and Benny Thörnberg

Abstract

3D object recognition has been a cutting-edge research topic since the popularization of depth cameras. These cameras enhance the perception of the environment and so are particularly suitable for autonomous robot navigation applications. Advanced deep learning approaches for 3D object recognition are based on complex algorithms and demand powerful hardware resources. However, autonomous robots and powered wheelchairs have limited resources, which affects the implementation of these algorithms for real-time performance. We propose to use instead a 3D voxel-based extension of the 2D histogram of oriented gradients (3DVHOG) as a handcrafted object descriptor for 3D object recognition in combination with a pose normalization method for rotational invariance and a supervised object classifier. The experimental goal is to reduce the overall complexity and the system hardware requirements, and thus enable a feasible real-time hardware implementation. This article compares the 3DVHOG object recognition rates with those of other 3D recognition approaches, using the ModelNet10 object data set as a reference. We analyze the recognition accuracy for 3DVHOG using a variety of voxel grid selections, different numbers of neurons (N_h) in the single hidden layer feedforward neural network, and feature dimensionality reduction using principal component analysis. The experimental results show that the 3DVHOG descriptor achieves a recognition accuracy of 84.91% with a total processing time of 21.4 ms. Despite the lower recognition accuracy, this is close to the current state-of-the-art approaches for deep learning while enabling real-time performance.

Keywords

3D object recognition, 3D image processing, histogram of oriented gradients, handcrafted descriptor, embedded systems, machine learning

Date received: 26 March 2020; accepted: 12 November 2020

Topic Area: Vision Systems
Associate Editor: Antonios Gasteratios
Topic Editor: Antonio Fernandez-Caballero

Introduction

Over the last decade, object recognition through visual cameras has been a fundamental computer vision research question. The introduction of consumer depth cameras in recent years has led to an extension of computer vision from 2D to 3D data, thus enabling a real-world visual perception. Methods for object recognition therefore need to be extended to extract 3D object shapes and volumetric features. In addition to 2D data, 3D data can provide

Department of Electronics Design, Mid Sweden University, Sundsvall, Sweden

Corresponding author:
Cristian Vilar, Department of Electronics Design, Mid Sweden University, Holmgatan 10, 85170 Sundsvall, Sweden.
Email: cristian.vilar@miun.se

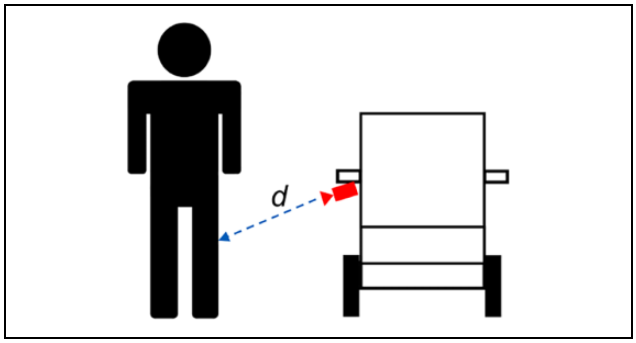

Figure 1. Contactless controlled wheelchair using caregiver's detection and distance measurement to allow wheelchair contactless control.

geometrical information and true distance measurements for the objects, and ideally is insensitive to illumination variations. Therefore, 3D data can be used to improve the overall performance compared to 2D.

This study was motivated by the desire to develop a contactless control of a powered wheelchair using the caregiver's position as a reference to drive the wheelchair in a side-by-side procession. Hence, the powered wheelchair requires to measure relative distances (d) from the surrounding objects while at the same time recognizing the caregiver from any other objects (Figure 1). A depth camera is the most suitable selection for the camera system. However, although depth data processing and 3D object recognition are simple tasks for human perception, they are a huge challenge for computer vision due to limitations of the 3D image data acquisition and the computational power required for real-time 3D data processing.[1] These limitations must be evaluated in advance to choose a proper depth data processing approach.

Deep learning is a cutting-edge approach for object recognition which tries to imitate the human learning behavior by extracting information directly from the raw images. This requires complex algorithms such as convolutional neural networks (CNNs), to extract and classify a hierarchy of increasingly abstract features to detect and recognize the objects in the scene. Despite their good performance, CNNs pose high requirements on computational and memory resources, especially for large data amounts as in 3D point clouds. Unfortunately, powered wheelchairs and autonomous robots have severe constraints in terms of power, space, heat dissipation, and hardware resources,[1] meaning that real-time CNN implementations are unfeasible for our application.

As an alternative to CNNs, the use of handcrafted features is the classic computer vision approach for object recognition. The idea is to extract different features from the raw image to generate an object descriptor, after which a supervised classifier learns patterns from the descriptor to estimate the object's class. The computational requirements for this approach depend mainly on the total number of features (N_{Features}) in the descriptor to be processed by the classifier. We expect a lower N_{Features} to require less computational resources than CNN approaches, and thus to be plausible for implementation in real-time robotics applications. However, reducing the N_{Features} before the classification can lead to a performance decrease, and so a balance between performance and N_{Features} is required.

In this article, we evaluate a 3D handcrafted object descriptor that was developed by Dupre and Argyriou[2] as an extension of the original 2D histogram of oriented gradients (HOG)[3] to support volumetric 3D data (3DVHOG). This descriptor is applied in combination with a supervised support vector machine (SVM) or a single hidden layer feedforward neural network (SLFN) classifier for 3D object recognition. The scientific contribution of this article is to explore firstly the 3DVHOG descriptor for 3D object recognition[2] and secondly the combination of data preprocessing, post-processing, and classifier settings to reduce both the computational cost and the power requirements while balancing the classification performance. Our study therefore provides the base information required to enable implementation in an embedded system for robotics and real-time applications.

We analyze the effect of reducing the extremely high dimensionality of the 3DVHOG features by applying principal component analysis (PCA) as well as the effect of choosing different numbers of hidden neurons (N_h) in an SLFN classifier. We use the Princeton ModelNet10 data set[4] with volumetric images of 10 different object classes as a reference to train, validate, and test the overall data processing steps and also to compare our recognition rates with those of others. Despite targeting an embedded system to detect the caregiver in the end, we perform our analysis on principal component (PC) hardware at this stage. We do this to compare the performance of the found processing chain in general with other object recognition approaches based on the ModelNet10 data set. In the future, we plan to evaluate the system using real data and focus on the caregiver detection. Besides that, the proposed method is not limited to embedded systems and the caregiver detection. It is rather applicable to any other object recognition task as well.

Related works

The extensive existing work on 3D object recognition uses several different approaches that can generally be classified in terms of input data type: (1) RGB-D data approaches, (2) multi-view CNN (MVCNN) approaches, (3) volumetric CNN approaches, and (4) handcrafted 3D object descriptors.

RGB-D data approaches

Depth cameras such as Microsoft Kinect provide an additional 2D parallel output channel to RGB to encode the

depth information (RGB-D). RGB-D approaches, then, extend the 2D image architecture to four channels by adding the depth information for each camera pixel (2.5D). Due to the popularization of depth cameras along with the possibility to using well-known 2D image recognition frameworks, there is a very extensive body of research using RGB-D approaches for 3D object recognition in combination with 2D handcrafted object descriptors[5,6] or 2D CNN approaches.[7,8] However, RGB-D is still a 2D image and so does not fully exploit the complete 3D volumetric information of the objects. We believe that using the entire 3D information will provide better results than the 2D recognition approaches to RGB-D data processing.

MVCNNs approaches

MVCNN approaches transform 3D object recognition into a series of 2D image recognition tasks by rendering each 3D object from different 2D viewpoints and extracting 2D features for each image projection. MVCNNs uses the same well-developed CNN recognition frameworks for 2D images.[9–12] However, in comparison with volumetric approaches, MVCNN approaches have a lower feature dimensionality, are more efficient to compute, and are more robust against noise and artifacts such as holes.[13] Thus, they are more suitable for real-time applications and noisy camera data. A 2D cylindrical panoramic projection (DeepPano)[14] enables rotation invariance and achieves a 88.66% accuracy on the ModelNet10 data set. Su et al.,[13] instead, used an MVCCN approach including 80 rendered object views, achieving a maximum recognition accuracy of 90.1% on the ModelNet40 data set. Johns et al.[15] extended the idea of MVCCN to use generic multi-view camera trajectories and achieved a maximum recognition accuracy of 92.8% on the ModelNet10 data set. Sfikas et al.[16] create an augmented panoramic view by a concatenated spatial and orientation domains to create an augmented panoramic view to feed a CNN, achieving a recognition accuracy of 91.1% on the ModelNet10 data set. Finally, Yavartanoo et al.[17] proposed a cutting-edge multi-view approach (SPNet) that achieved a recognition accuracy of 97.25% on the ModelNet10 data set. This approach uses a stereographic mapping to project the 3D surfaces onto a 2D planar image and has lower processing and memory requirements than other recognition approaches. However, it requires the use of a powerful graphical processing unit (GPU) that involves high power consumption and high heat dissipation. Hence, it is not suitable for robotics or powered wheelchair applications due to their real-time operation and hardware constraints.

Volumetric CNN approaches

Volumetric approaches extract 3D volumetric features through a CNN directly from the 3D data, thus exploiting the complete 3D geometry of the objects without including additional 2D features. The object's data is preprocessed through a voxelization processing step, and then the point-cloud data of each object is converted into a uniform 3D grid of binary voxels.[18] RGB-D data can be preprocessed to convert the depth information to a point-cloud representation and then voxelized,[19] and so this approach is suitable for depth cameras. The first volumetric approach to be proposed was the 3DShapeNets,[4] which represents the 3D mesh as a probability distribution of binary voxels. 3D shape distributions are learned by a five-layer convolutional deep belief network. This approach achieves a recognition accuracy of 83.5% on the ModelNet10 data set. Hegde and Zadeh[20] then proposed the FusionNet CNN volumetric approach that uses up to two CNNs in combination with the AlexNet-based[21] MVCNN approach. The three CNN subnetworks are fused to combine multiple data representations and improve the recognition accuracy to 93.1%. Brock et al.[22] presented the state-of-the-art approach, which uses a 45-layer 3D volumetric CNN and a large data augmentation data set for training, achieving a maximum recognition accuracy of 97.25% on the ModelNet10 data set. Despite their good recognition performance, 3D volumetric CNN approaches are large, complex, and highly computational and memory demanding, meaning that none of the above mentioned volumetric approaches are suitable for real-time operation.[18,17]

Maturana and Scherer[23] proposed the VoxNet approach, which considerably reduces the number of model parameters. This enables real-time operation while at the same time increasing the recognition accuracy to 92% on the ModelNet10 data set. Qi et al.[24] proposed the PointNet, a real-time CNN for object recognition based on a point density occupancy grids data representation, achieving a recognition accuracy of 77.6% on the ModelNet10 data set. Zhi et al.[18] proposed a real-time 3D object recognition approach called LightNet, which combines the task of subvolume, supervision, and orientation prediction to learn discriminative 3D features from multitask learning. LightNet achieves a recognition accuracy of 93.94% on the ModelNet10 data set.

Uniform voxel grids lead to excessive use of memory and processing resources. Reviewed approaches above use, consequently, a relatively small spatial resolution to map the 3D data onto the volumetric uniform grid of voxels, typically 30^3 voxels. Therefore, image resolution is not comparable to that of a 2D camera. Larger grids would lead to intractable processing and memory requirements, as these increase cubically with the resolution.[25] Small spatial resolutions mean less detailed objects and hence lower recognition accuracies. To solve this problem, it has been proposed that the 3D object shape should be mapped onto adaptively subdivided hierarchical grids, with dense cells near the object's surface.[25–27] Although the recognition accuracies achieved with this technique are comparable to those of other approaches, they require less

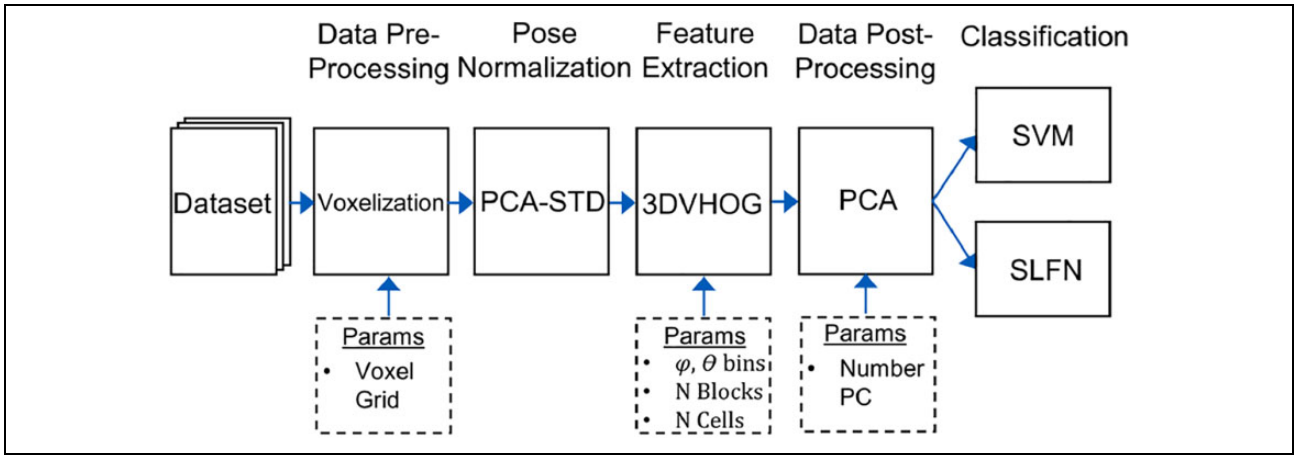

Figure 2. Object recognition processing steps.

computational power and memory resources. However, they require the use of GPU, and so they are not suitable for real-time operation with strong hardware and power dissipation constraints.

Handcrafted descriptors

Handcrafted descriptors are the classical approach to object recognition. The idea is to extract a set of features from the object to generate an object descriptor that can be used as an object signature. By training a supervised classifier, it is possible to learn from the descriptors to identify a pattern regarding the object's class. This 2D classical approach can also be applied to 3D data by using specific 3D object descriptors. The efficiency of the approach relies on the effectiveness of the descriptor in capturing the object class information. A large number of handcrafted approaches exist.[28–30] Generally, handcrafted descriptors can be divided into local and global features.[31] Local feature descriptors capture key points of the object's shape and so are focused on the shape around several key points. They tend to be more computationally expensive, and thus are not suitable for real-time robotics applications with high limitations in terms of hardware and heat dissipation. The most popular local descriptors are spin images,[32] fast point feature histograms,[33] and 3D SURF.[34] Conversely, global feature descriptors capture shape information using the overall appearance of the object. They are increasingly used in object recognition, object manipulation, and geometric characterization. They are efficient in terms of computation time, thus allowing real-time performance.[35] Uses in 3D object shape recognition include ensemble of shape functions,[36] global fast viewpoint feature histograms,[33] and 3DVHOG.[2,37] However, they ignore the local object's details, leading to lower performance. In both of the above mentioned 3DVHOG implementations reviewed,[2,37] the level of local object detail is configured by different 3DVHOG parameters and thus can be modified according to the recognition requirements. However, there is a compromise between the local detail level, the descriptor feature dimensionality, and the elapsed processing time. Analysis of this compromise is the intent of the present article.

Table 1. Object classes and total numbers of training and test objects in the ModelNet10 data set.

Object class	Training objects	Test objects
1-Bathtub	106	50
2-Bed	515	100
3-Chair	881	100
4-Desk	200	85
5-Dresser	200	85
6-Monitor	465	100
7-Nightstand	200	85
8-Sofa	680	100
9-Table	392	100
10-Toilet	344	100
Total $N_{Objects}$	3991	905

Method

Our goal was to evaluate the 3DVHOG handcrafted object descriptor to reduce the computational cost as much as possible compared to deep learning approaches for 3D object recognition tasks. Our proposed processing steps are summarized in Figure 2.

Data preprocessing

We chosen the Princeton ModelNet10 data set as a common reference to validate our classification results. This data set includes a volumetric 3D representation of 10 different object classes ($N_{Classes}$) with separate data sets for training and test (cf. Table 1 and Figure 3). The 3D volumetric objects are first scale normalized to [0, 1] and then quantized in a mesh grid of cubic cells, also called voxels, as a preprocessing step. The number of quantized voxels (N_{Voxels}) for each object is an input parameter that can be

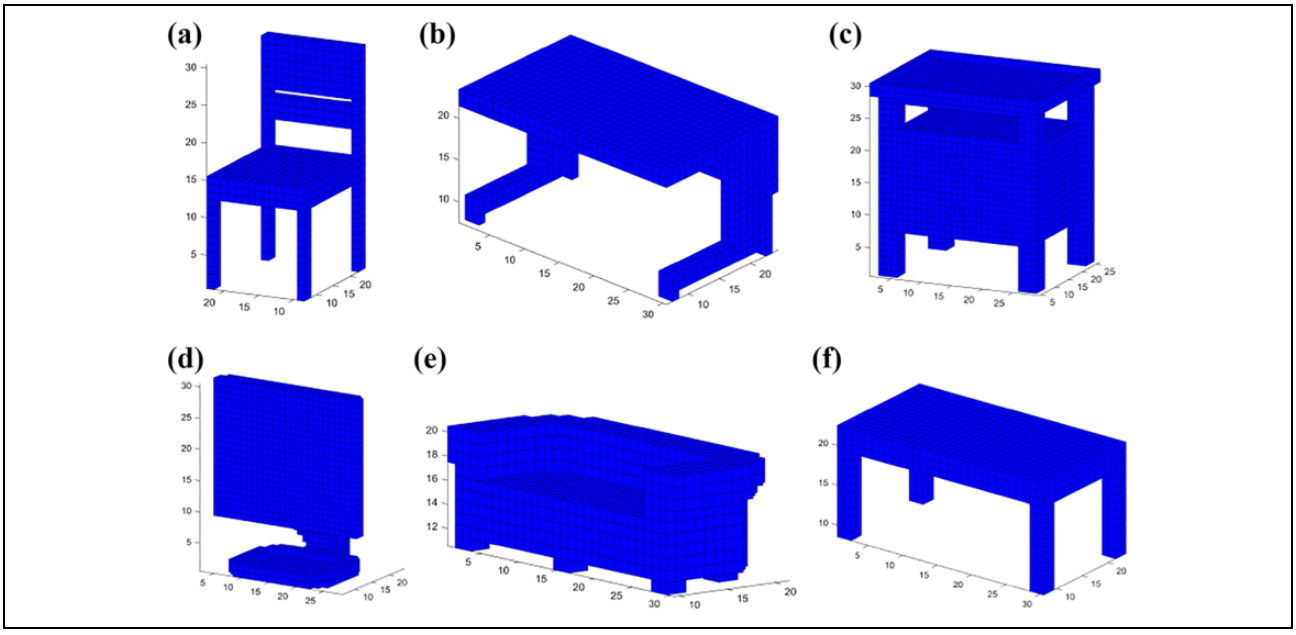

Figure 3. Validation example of data set objects with a grid of 30^3 voxels. Object classes: (a) chair, (b) desk, (c) nightstand, (d) monitor, (e) sofa, and (f) table.

modified to include more detailed object information into the analysis.

Pose normalization

We propose an additional data preprocessing step to achieve rotation invariance in the object classification. Rotation invariance is crucial for detecting objects whose pose is rotated according to the camera position. As a HOG-based descriptor, the 3DVHOG is not rotation invariant. Thus, when an object is rotated the 3DVHOG descriptor changes, making it impossible for the classifier to estimate the correct object class. To solve this, we have proposed a pose normalization method based on the PCA pose normalization in combination with the standard data deviation (PCA-STD).[38] To include the pose normalization preprocessing step into the recognition results, we first rotate each test data set object randomly along the three-axis and later we normalize its pose using the PCA-STD method.

Feature extraction

The 3DVHOG object descriptor[2] was originally developed for environment hazard detection and risk evaluation by detecting the presence of dangerous 3D objects in the 3D scene. Here, we instead use it as a general object descriptor to extract volumetric features from the objects. Like the 2D implementation of the HOG, there are some input parameters to configure the descriptor, but in this 3D implementation they are extended to a 3D representation and hence include the number of angle bins, (θ_{Bins}, φ_{Bins}), the step size ($\text{Step}_{\text{Size}}$), and the cell size ($\text{Cell}_{\text{Size}}$). The total number of blocks (N_{Blocks}) (1) and the total number of features (N_{Features}) (2) for each object depend on the configuration of these parameters and are calculated as follows

$$N_{\text{Blocks}} = * \frac{N_{\text{voxels}}/\text{Cell}_{\text{Size}}{}^3}{\text{Step}_{\text{Size}}} \tag{1}$$

$$N_{\text{Features}} = N_{\text{Blocks}} \cdot N_{\text{Cells}} \cdot \theta_{\text{Bins}} \cdot \varphi_{\text{Bins}} \tag{2}$$

with the number of cells per object (N_{Cells}) calculated as

$$N_{\text{Cells}} = \text{Block}_{\text{Size}}^3 \tag{3}$$

It is important to evaluate the impact of each parameter to choose a proper descriptor setup that minimizes the required N_{Features} while at the same time maximizing the classification accuracy.

Data postprocessing

The different N_{Blocks} are vectorized as single vector descriptors of dimension N_{Features}. All the vector descriptors of each object are then reshaped into a matrix of features as illustrated in Figure 4. Depending on the number of angle bins (θ_{Bins}*φ_{Bins}) and the N_{Cells}, the final vector descriptor of N_{Features}, and by extension the feature matrix, can have an extremely high dimensionality. This high dimensionality limits the real-time operation of the overall processing steps and will also require more computational and memory resources to process all the data.

We therefore apply the PCA as method to reduce the feature dimensionality. The final number of recomputed features depends on the number of principal components

(N_{PC}) used to project the original data (Figure 5). Therefore, the minimum N_{PC} must be evaluated to maximize the classification rate while at the same time reducing the required $N_{Features}$ as much as possible, see equation (4). Once N_{PC} is determined, we project the 3DVHOG descriptors from the test object's data set onto the PCs. The resulting reduced 3DVHOG descriptor has a lower feature dimensionality

$$N_{Features} = N_{PC} \tag{4}$$

The maximum N_{PC} is limited by the total number of objects in the training data set ($N_{Objects} = 3991$; see Table 1). Thus, it is not possible to reduce the initial $N_{Features}$ in more than $N_{Objects} - 1$. If further reduction is required, it will be necessary to increase the $N_{Objects}$ in the training data set by performing a training data set augmentation.

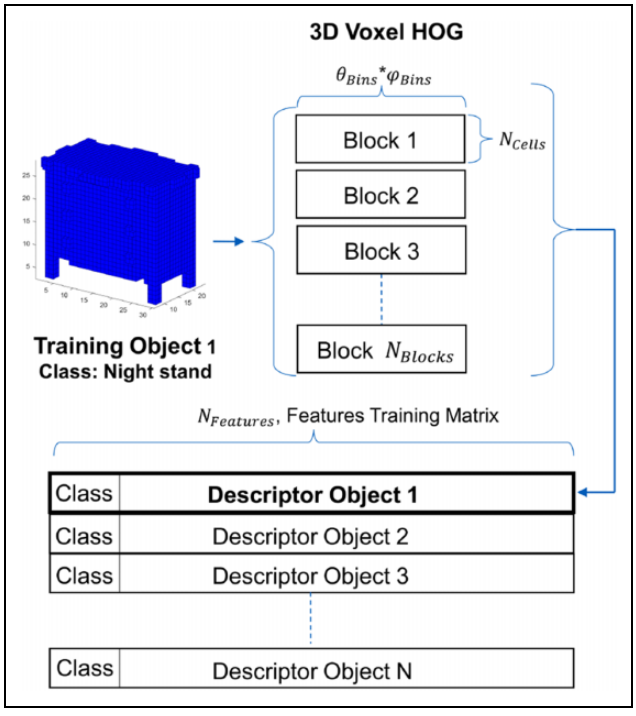

Figure 4. 3DVHOG feature matrix and feature vectorization. 3DVHOG: 3D voxel-based extension of the 2D histogram of oriented gradient.

Object classifiers: SVM and SLFN

Information regarding the object classes must be extracted from the feature vector. Once we have learned this information from the training data set, it is possible to estimate the class of a new input object. The learning process is performed by training a supervised multiclass classifier. We have evaluated two different classifiers: an SVM and an SLFN. We expect different classification results when they work in combination with 3DVHOG and PCA, and an evaluation of this was one of the aims of the present study. Configuration and data parameters of the classifiers are shown in Table 2.

Regarding the N_h used in the SLFN classifier, there is no specific rule to choose a proper N_h that maximizes the classification accuracy.[39] As shown in Table 4, we need to deal with extremely large $N_{Features}$ vectors, and therefore choose the "Shibata and Ikeda" criteria[40] (N_{hSI}) (5) to obtain a lower N_h with respect to $N_{Features}$

$$N_{hSI} = \sqrt{N_{Features} * N_{Classes}} \tag{5}$$

However, we also evaluated N_h as a design criterion to minimize the elapsed processing time and measure the dependence of the classification accuracy on N_h. We selected N_h criteria according to $N_{Features}$. Other criteria,

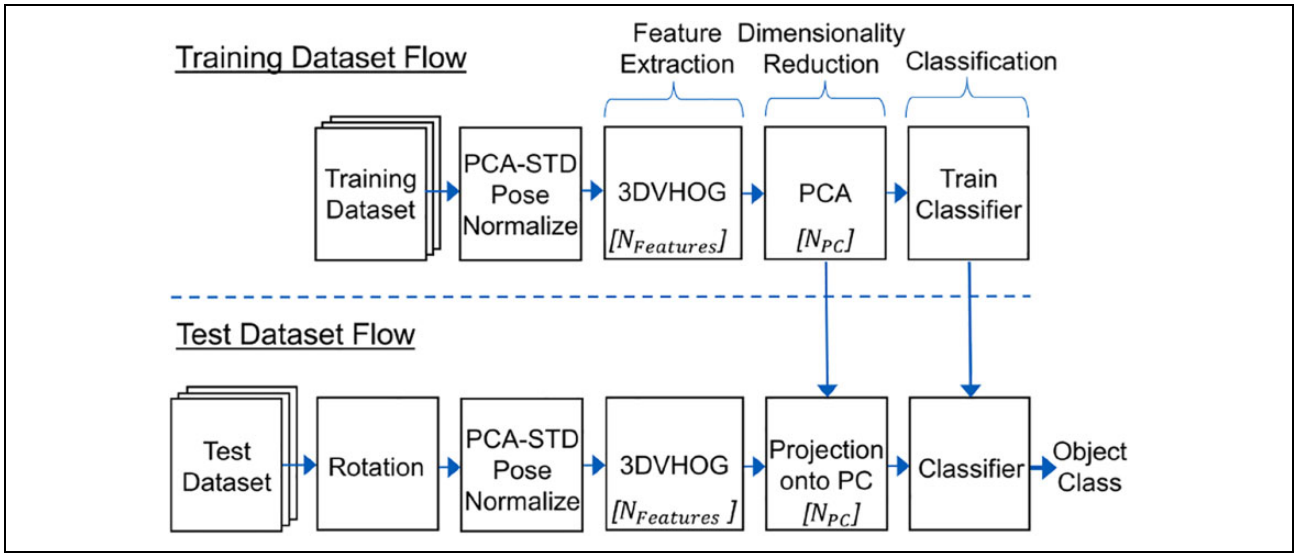

Figure 5. PCA feature reduction. PCA: principal component analysis.

which involve a higher N_h and thus higher memory and computing requirements, were not considered due to the real-time performance and hardware constraints.

Experiments

We defined several experiments to evaluate the effect of the different preprocessing and postprocessing parameters on the classification accuracy (Figure 2). The order of these experiments follows the logic design flow that must be considered for a proper parameters configuration and data analysis, (Figure 6).

Table 2. SVM and SLFN classifier configuration parameters.

Parameter	SLFN classifier
Training	Scaled conjugate gradient
Performance	Cross-entropy
Calculations	MEX
Data division	Random 15%
Number of neurons	Shibata and Ikeda criteria
Parameter	**SVM classifier**
Type	Multiclass
Method	ECOC
Kernel function	Radial basis function
Optimization	ISDA
Data division	Holdout partition 15%

SVM: support vector machine; SLFN: single hidden layer feedforward neural network; ECOC: Error correcting codes; ISDA: iterative single data.

Table 3. 3DVHOG initial configuration parameters.

$Cell_{Size}$	$Block_{Size}$	$Step_{Size}$	φ_{Bins}	θ_{Bins}
6	2	2	18	9

3DVHOG: 3D voxel-based extension of the 2D histogram of oriented gradient.

Table 4. 3DVHOG $N_{Features}$ for grids of 20^3, 30^3 and 40^3 voxels.

Voxels	N_{blocks}	N_{Cells}	$N_{Features}$
40^3	27	8	34,992
30^3	8	8	10,368
20^3	1	8	1296

3DVHOG: 3D voxel-based extension of the 2D histogram of oriented gradient.

Results and analysis

Experiment 1: Voxel grid and PCA

In experiment 1, we analyzed the effect of choosing different voxel grid configuration parameters while reducing the $N_{Features}$ dimensionality using PCA. We used the classification accuracy as a key measurement for both classifiers considering pose normalization for rotational invariance. Classification accuracy was defined as the averaged class accuracy (ACC_{Class})

$$ACC_{Class} = \frac{1}{N}\sum_{i=1}^{N}\frac{(T_{p_C} + T_{n_C})}{(T_{p_C} + T_{n_C} + F_{p_C} + F_{n_C})} \quad (6)$$

where T_{p_C} are the class C true positives, T_{n_C} the class C true negatives, F_{p_C} the class C false positives, F_{n_C} the class C false negatives, and N the number of classes.

Increasing the voxel grid will provide more detailed volumetric information about the objects but can also increase the differences between objects of the same class. By contrast, when the voxel grid is decreased, the objects are less detailed but the differences between objects of the same class are smaller. Smaller intraclass differences make it easier for the classifier to extract a pattern from the feature matrix and, hence, can increase the classification performance. However, this can also cause smaller differences between objects of different classes, thus decreasing the classification performance. It is therefore necessary to find a combination of the lowest voxel grid value, the minimum required N_{PC}, and the right classifier approach to improve the classification accuracy while reducing as far as possible the amount of data that requires processing

In line with the literature and the total $N_{Features}$, we chose a voxel grid with sizes ranging from [20^3 to 40^3] voxels for the experiment. The 3DVHOG initial configuration parameters ($Cell_{Size}$, $Block_{Size}$, $Step_{Size}$, θ_{Bins}, φ_{Bins}) were the same for each voxel grid value, (Table 3) and so the total depends only on the voxel grid in each configuration.

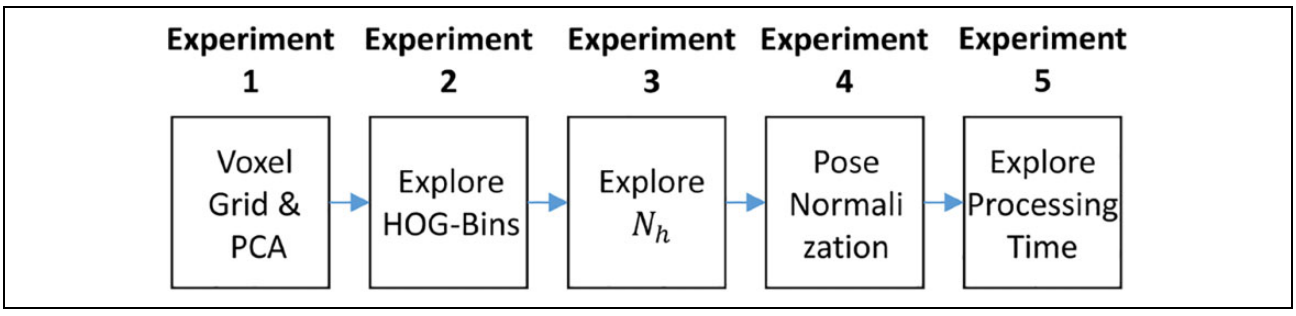

Figure 6. Experimental design flow.

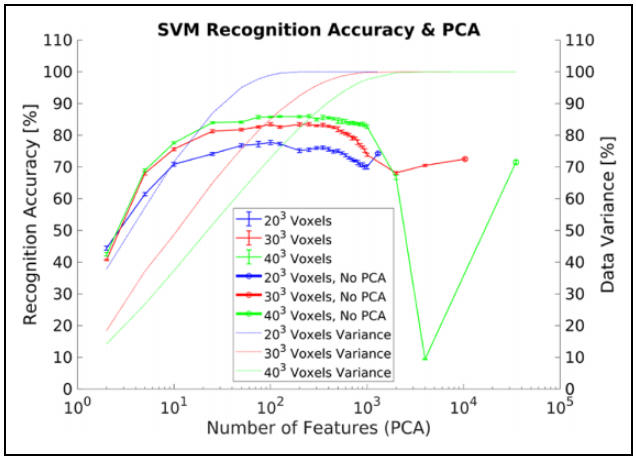

Figure 7. SVM classification accuracy, data variance, and PCA dimensionality reduction for grids of 20^3, 30^3, and 40^3 voxels. SVM: support vector machine; PCA: principal component analysis.

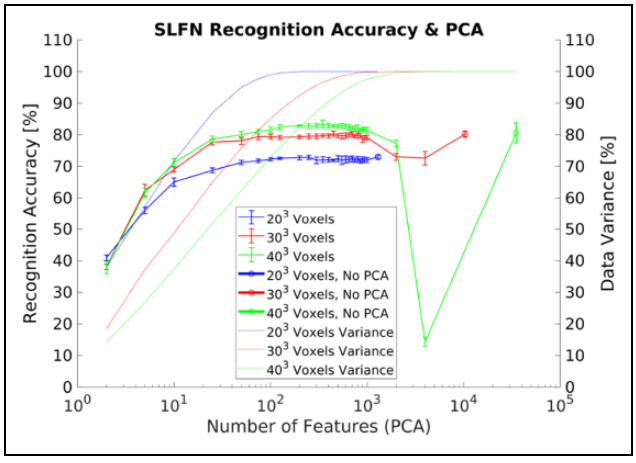

Figure 8. SLFN classification accuracy, data variance, and PCA dimensionality reduction for grids of 20^3, 30^3, and 40^3 voxels. SLFN: single hidden layer feedforward neural network; PCA: principal component analysis.

A higher voxel grid means a higher N_{Features} (Table 4), which in practice can cause difficulty for the SVM and SLFN classifiers due to the high dimensionality of the feature matrix. A higher dimensionality leads to higher processing times and memory requirements, making it unsuitable for real-time operation. However, postprocessing the feature matrix with PCA reduces N_{Features} to N_{PC}, see equation (4).

Classification accuracy, standard deviation, and data variance after all the preprocessing and postprocessing steps illustrated in Figures 2 and 5 for each voxel grid case are shown in Figure 7 for the SVM and Figure 8 for the SLFN. These results were calculated by averaging 10 different measurements. The figures also show the classification accuracy without applying PCA.

Classification accuracy was improved for both classifiers when PCA was applied. The maximum value was achieved by using approximately 100 PC in both cases (SVM: Figure 7, SLFN: Figure 8). The improvement was significantly better for the SVM classifier, which achieved a maximum classification accuracy of 85.5%. Both classifiers performed better with the highest voxel grid analyzed (40^3), but this was marginally better than using a grid of 30^3 voxels. A higher voxel grid increases the size of N_{Features} vector considerably (Table 4) and consequently also the memory requirements and processing time. We would therefore choose a 40^3 voxel grid in combination with 100 PC to improve the recognition accuracy, but we would also consider a 30^3 voxel grid for a real-time application while maintaining an acceptable recognition accuracy rate.

Table 5. 3DVHOG configuration parameters and N_{Features} for 3DVHOG angle bins for grids of 30^3 and 40^3 voxels.

$\text{Cell}_{\text{Size}}$		$\text{Block}_{\text{Size}}$	$\text{Step}_{\text{Size}}$
6		2	2
φ_{Bins}	θ_{Bins}	N_{Features} (30^3 voxels)	N_{Features} (40^3 voxels)
24	12	18,432	62,208
18	9	10,368	34,992
16	8	8192	27,648
14	7	6272	21,168
12	6	4608	15,552
10	5	3200	10,800
8	4	2048	6918
6	3	1152	3888
4	2	512	1728

3DVHOG: 3D voxel-based extension of the 2D histogram of oriented gradient.

Experiment 2: Explore 3DVHOG bins

In experiment 2, we evaluated the impact of using different θ_{Bins} and φ_{Bins} to compute the 3DVHOG descriptor considering pose normalization for rotational invariance. As with the voxel grid, if we increase θ_{Bins} and φ_{Bins}, then the 3DVHOG will capture more detailed information about objects but it will be harder for the classifier to extract a class pattern from the features matrix. In addition, the total N_{Features} and, by extension, the total size of the feature matrix is particularly dependent on the product of θ_{Bins} and φ_{Bins} as is shown in equation (2). Therefore, it was necessary to evaluate the impact of θ_{Bins} and φ_{Bins} in combination with the previous PCA results to reduce the total N_{Features} and therefore the computational requirements as far as possible. However, overfitting φ_{Bins} and θ_{Bins} may complicate the classification process. In addition, overfitting φ_{Bins} and θ_{Bins} quadratically increases the N_{Features} vector length (2) (Table 5). Note that φ_{Bins} is defined between 0° and 360° and θ_{Bins} between 0° and 180°, meaning that φ_{Bins} requires twice as many angle bins as θ_{Bins} to sample an angle with the same resolution. Minimum values correspond to $\varphi_{\text{Bins}} = 4$ and $\theta_{\text{Bins}} = 2$ to measure gradients

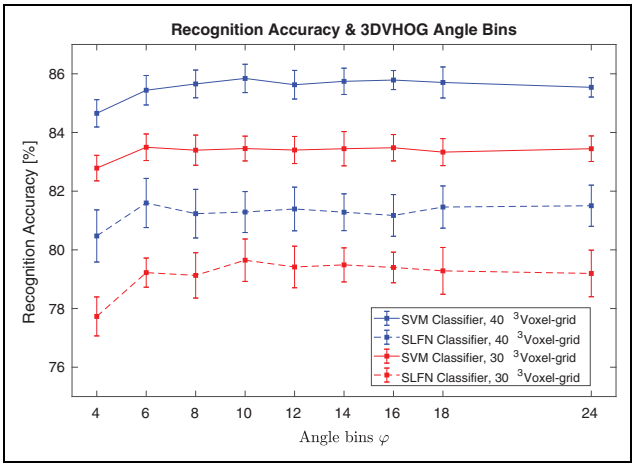

Figure 9. Classification accuracy with respect to the 3DVHOG angles using 100 PC for SVM and SLFN classifiers and grids of 40^3 versus 30^3 voxels. 3DVHOG: 3D voxel-based extension of the 2D histogram of oriented gradient; SVM: support vector machine; SLFN: single hidden layer feedforward neural network.

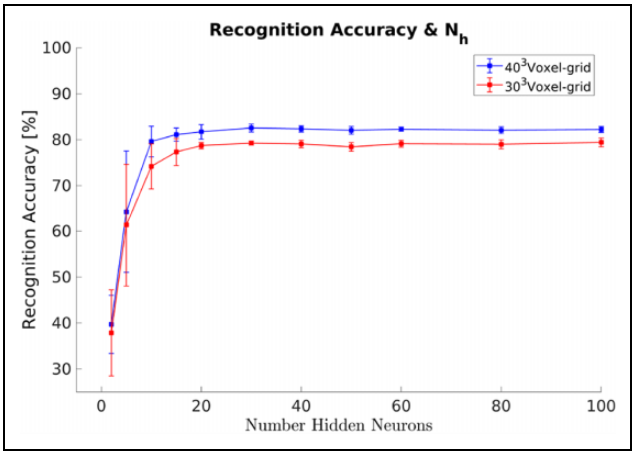

Figure 10. SLFN classification accuracy with respect to N_h for a 30^3 and 40^3 voxels grids with $\varphi_{Bins} = 4$ and $\theta_{Bins} = 2$. SLFN: single hidden layer feedforward neural network.

in all the 3D directions. The goal of this experiment was to find the lowest combination of φ_{Bins}, θ_{Bins}, and N_{PC} that can maximize the classification rate. Experimental results for all the angle bins and voxel grid cases defined in Table 5 are shown in Figure 9 for the SVM and SLFN classifiers. Classification accuracy and standard deviation of the measurements were calculated by averaging 20 measurements.

For SVM and SLFN classifiers, the recognition accuracy remains constant with respect to the number of angle bins. Thus, a low number of angle bins is enough for capturing the object class information. In addition, the SVM classifier performs better than SLFN for all analyzed cases. The maximum recognition accuracy achieved is 85.84% corresponding to $\varphi_{\text{Bins}} = 10$, $\theta_{\text{Bins}} = 5$ and grid of 40^3 voxels. Only for the lowest angle bins case ($\varphi_{\text{Bins}} = 4$ and $\theta_{\text{Bins}} = 2$), the recognition accuracy is marginally lower for both classifiers and voxel grids. However, the difference in terms of recognition accuracy with respect the best case is small while the N_{Features} are reduced significantly (cf. Table 5).

Experiment 3: Explore numbers of hidden neurons

In experiments 1 and 2, we used the $N_{h\text{SI}}$ criteria (5) to choose the proper N_h for the SLFN classifier. We also used PCA for feature dimensionality reduction and aimed to determine the best combination of PC and angle bins to improve the recognition accuracy. In experiment 3, we investigated the possibility of using a lower N_h as a method of reducing the processing time instead of using PCA, and so the data postprocessing indicated in Figure 2 was excluded. The experimental goal was to evaluate the minimum required N_h to improve the SLFN classification accuracy. We hypothesized that lower N_h would allow us to reduce the SLFN classification processing time due to the lower computer processing requirements and avoidance of the need to compute the PCA. Averaging results and data variance of 10 different measurements are shown in Figure 10 for a 40^3 and 30^3 voxel grids with φ_{Bins}=4 and $\theta_{\text{Bins}} = 2$.

As shown in Figure 10, SLFN classification accuracy was invariant to the N_h used. Thus, the results of this experiment agree with those of experiment 2. However, a lower N_h ($N_h < 20$) significantly decreased the classification accuracy. The best results in terms of classification accuracy and standard deviation were achieved by using $N_h = 30$; Increasing N_h beyond this did not change the classification accuracy but the standard deviation of 10 different measurements increased.

Experiment 4: Pose normalization

In experiment 4, we evaluated the pose normalization preprocessing step shown as the second operation in Figure 2. This pose normalization achieves rotational invariance for the 3DVHOG descriptor using the PCA-STD method defined by Vilar et al.[38] Performance is evaluated in terms of averaged $\text{ACC}_{\text{Class}}$ of 20 different measurements as a key measurement. For the evaluation, we rotated all objects in the test data set randomly between 0° and 360° for the three axes. The results were then compared without considering pose normalization and rotated objects.

An initial evaluation without pose normalization showed a recognition accuracy of 88.48% for nonrotated test data set objects but only 43.25% when the objects were rotated, due to the pose dependency of the 3DVHOG descriptor. When pose normalization was included as in Figure 2, the recognition accuracy increased to 84.91% for rotated objects while for nonrotated objecs, the recognition accuracy was 84.61% and thus comparable with that for rotated objects. These results are summarized in Table 6.

Table 6. Experimental maximum accuracy comparison for a grid of 40^3 voxels, 100 PC and $\varphi_{\text{Bins}} = 4$, $\theta_{\text{Bins}} = 2$ with respect to object rotation and pose normalization by PCA-STD.

Test data set rotation	No pose normalization (%)	PCA-STD (%)
No	88.48	**84.61**
Yes	43.25	**84.91**

PCA: principal component analysis; STD: standard data deviation.

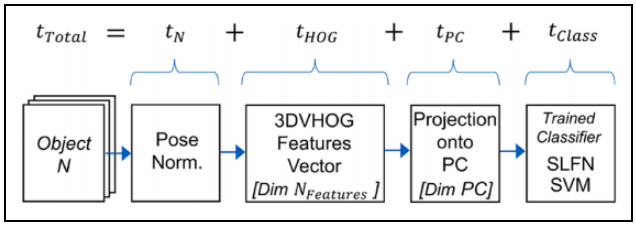

Figure 11. Processing chain and processing times definitions.

Table 7. Laptop computer specification parameters.

Parameter	Specification
Processor	Intel core i7-6500, 2.5 GHz
GPU	No
Memory	8(GB)
Operating system	Windows7 (64-bit)
Software	MATLAB 2018b

GPU: graphical processing unit.

Experiment 5: Explore processing time

In experiment 5, we measured the mean value of the elapsed processing time for the overall object recognition chain. This experiment compared all the processing durations according to the 3DVHOG parameter configuration, pose normalization, and also evaluated the elapsed processing time improvements achieved by using PCA and a lower N_h. Experiments 1 and 2 gave us information on the best 3DVHOG parameter configuration to find a good balance between classification accuracy and required N_{PC} and N_h. According to the previous results, we choose $N_{\text{PC}} = 100$, $N_h = 30$, $\varphi_{\text{Bins}} = 4$, and $\theta_{\text{Bins}} = 2$ to reduce as much as possible the computational cost and therefore enable real-time performance.

With these preprocessing and postprocessing parameter configurations, we trained the SVM and SLFN classifiers to estimate the object class from the test data set objects as shown in Figure 11. We computed, object-by-object, the elapsed processing time for the 3DVHOG descriptor computation (t_{HOG}), data projection onto the PC (t_{PC}), and data classification through the trained SVM and SLFN classifiers (t_{CLASS}). Total elapsed time for all processing chain steps was also computed (t_{Total}). The required time for training the SVM and SLFN classifiers was not considered in this analysis, since training is an offline data processing operation. Processing times were measured in a low-range laptop computer with the specifications shown in Table 7.

Although these measurements are not from an embedded system, they provide a valuable reference to qualitatively compare and analyze the different processing times. As such, our analysis gives an early insight into expected processing to select an appropriate variant for later implementation in the real embedded system. Summarized mean values of the different processing times are shown in Table 8 for a 30^3 voxel grid and in Table 9 for a 40^3 voxel grid.

The measured t_N and t_{PC} were relatively small and almost negligible in comparison with the t_{Class} and t_{HOG}. By using N_{PC} and N_h according to the results in experiments 1 and 3, it was possible to considerably reduce the t_{Class} for the SVM and the SLFN, while the classification accuracy was still comparable.

Table 8. Averaged accuracy and elapsed processing times for a 30^3 voxel grid and (1) an SVM with full set of 512 N_{Features}, (2) an SLFN with N_h criteria ($N_{h\text{SI}}$=71), (3) an SVM with 100 PC, and (4) an SLFN with $N_h = 30$.

$\varphi_{\text{Bins}} = 4$, $\theta_{\text{Bins}} = 2$, voxel grid=30^3, $N_{\text{Features}} = 512$

Case	t_N (ms)	t_{HOG} (ms)	t_{PC} (μs)	t_{Class} (ms)	t_{Total} (ms)	Acc. (%)
SVM_{Full}	1.3	5.7	10	11.6	18.61	81.2
$\text{SLFN}_{h\text{SI}}$	1.3	5.7	—	9.6	16.6	80
SVM_{100}	1.3	5.7	2.4	6.5	**13.5**	82.73
SLFN_{30}	1.3	5.7	—	8.1	**15.1**	79.6

SVM: support vector machine; SLFN: single hidden layer feedforward neural network.

Table 9. Averaged accuracy and elapsed processing times for a 40^3 voxel grid and (1) an SVM with a full set of 1728 N_{Features}, (2) an SLFN with $N_{h\text{SI}}$ criteria (N_h=131), (3) an SVM with 100 PC, and (4) an SLFN with $N_h = 30$.

$\varphi_{\text{Bins}} = 4$, $\theta_{\text{Bins}} = 2$, voxel grid = 40^3, $N_{\text{Features}} = 1728$

Case	t_N (ms)	t_{HOG} (ms)	t_{PC} (μs)	t_{Class} (ms)	t_{Total} (ms)	Acc. (%)
SVM_{Full}	2	13.1	280	37.4	52.7	82.2
$\text{SLFN}_{h\text{SI}}$	2	13.1	—	16.7	31.8	82.2
SVM_{100}	2	13.1	7.3	6.5	**21.6**	84.91
SLFN_{30}	2	13.1	—	8.8	**23.9**	82.7

SVM: support vector machine; SLFN: single hidden layer feedforward neural network.

Object classification analysis

The confusion matrix for the highest recognition accuracy analyzed in Tables 8 and 9 is shown in Table 10. Most of the object classes were relatively well classified, but there was clear misclassification between 4 (desk) and 9 (table)

Table 10. Confusion matrix for SVM classifier with 40^3 voxel grid, $\varphi_{\text{Bins}} = 4$, $\theta_{\text{Bins}} = 2$, Case SVM_{100}, Acc = 84.91%, using the ModelNet10 test data set.

$\varphi_{\text{Bins}} = 4$, $\theta_{\text{Bins}} = 2$, voxel grid=40^3, SVM_{100}											
Estimated class	1	43	0	0	3	1	0	0	0	0	0
	2	2	93	2	0	1	9	0	2	0	1
	3	0	0	93	2	0	3	1	1	0	2
	4	1	0	0	57	0	1	1	0	18	1
	5	0	0	0	2	66	1	18	0	0	0
	6	4	4	0	2	1	84	1	1	0	0
	7	0	1	0	3	16	0	56	2	0	0
	8	0	1	0	7	0	0	0	94	0	0
	9	0	1	0	8	0	1	8	0	82	0
	10	0	0	5	2	1	1	1	0	0	96
		1	2	3	4	5	6	7	8	9	10
						Input class					

SVM: support vector machine.

Table 11. Averaged accuracy and total elapsed time comparison of the PCA-STD + 3DVHOG descriptor with respect to other CNN approaches using the Princeton ModelNet10 data set.

Method	Accuracy (%)	GPU	T_{Total} (ms)
PCA-STD+3DVHOG	84.91	No	21.6
DShapeNets[4]	83.5	Yes	—
Voxnet[23]	92	Yes	3
PANORAMA[16]	91.1	Yes	—
PointNet[24]	77.6	Yes	24.6
RotationNet[41]	98.46	Yes	—
OrthographicNet[12]	88.56	Yes	—
SPNet[17]	97.25	Yes	122.5
LightNet[31]	93.94	Yes	5.9

PCA: principal component analysis; STD: standard data deviation; 3DVHOG: 3D voxel-based extension of the 2D histogram of oriented gradient; CNN: convolutional neural network; GPU: graphical processing unit.

and between classes 5 (dresser) and 7 (nightstand) (Table 1).

Discussion

Comparison with previous results

The 3DVHOG descriptor in combination with PCA-STD pose normalization and PCA dimensionality feature reduction achieved a recognition accuracy of 84.91% and a t_{Total} of 21.6 ms (Table 9). This recognition accuracy is lower than but close to the state-of-the-art approaches shown in Table 11. In addition, our approach enables frame rates of 46 fps and thus real-time performances without the need for a GPU. By contrast, all the reviewed approaches use CNN and GPU and so are not suitable for robotics applications where power consumption, real-time processing, and limited hardware resources are important constraints. Despite the fact that some of the CNN approaches achieve better results in general, our results show that handcrafted descriptors can also be considered, especially when real-time processing and low power are required.

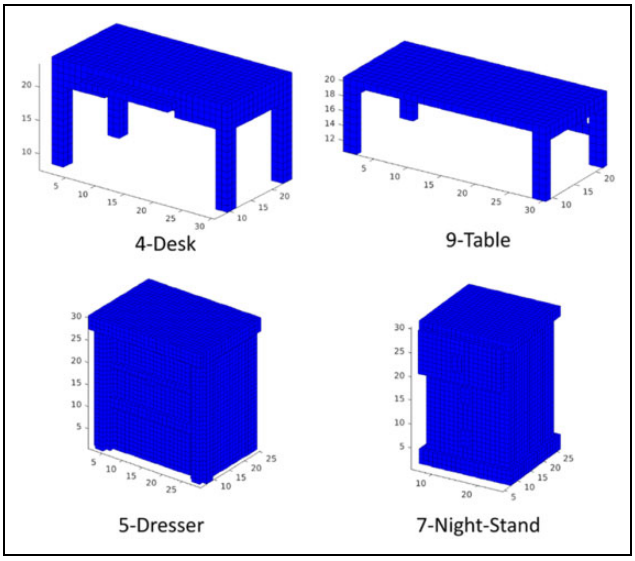

Figure 12. Object similarity between classes 4 (desk), 9 (table) and 5 (dresser), 7 (nightstand).

Classification analysis

As shown in Table 10, there was substantial misclassification between classes 4 (desk) and 9 (table) and 5 (dresser) and 7 (nightstand) (Table 1). These classification errors considerably decreased the final recognition accuracy. The main reason for the misclassified objects is the high similarity between classes (Figure 12). The 3DVHOG descriptor cannot capture enough local detailed information from the classes to allow the classifier to differentiate between them. This problem can be solved by using a higher voxel grid to capture more local detailed information. However, increasing the local detailed information can also reduce the similarities between objects of the same class and consequently reduce the overall recognition accuracy. In addition, a higher voxel grid leads to a considerable increase in N_{Features} (Table 4), and hence also in both t_{Total} a memory requirements. By contrast, increasing the voxel grid from 30^3 to 40^3 voxels (cf. Figures 7 and 8) leads to a relatively small increment in the recognition accuracy. Therefore, we believe that higher voxel grid values will not significantly increase the recognition accuracy.

Experimental design flow

The results from this study can guide us in choosing the best classifier, voxel grid, and configuration of parameters N_{features} and N_h. However, the chosen configuration may be optimal only when using the ModelNet10 data set. We can expect any optimal configuration of these parameters to depend mainly on the degree of similarity between objects from the same class and from different classes. A higher

degree of similarities between objects from the same class in comparison to objects from different classes will require less N_{Features} to estimate the class to which the object belongs. This means that the experimental results depend largely on the data set. As consequence, the reported system exploration illustrated in Figure 6 must be repeated if the same chain of data processing operations is to be reused on ground-truth data for the wheelchair application.

Conclusions

Experimental results show that the 3DVHOG descriptor in combination with PCA-STD pose normalization achieves a classification accuracy of 84.91% and total processing time of 21.6 ms on the ModelNet10 data set. The accuracy is lower, but close to, the accuracy achieved by state-of-the-art CNN approaches, while real-time processing is enabled on low-range CPUs. Thus our approach is suitable for embedded robotic vision constrained by requirements of low power and real-time performance. Less tightly constrained applications can instead benefit from using CNN to achieve the highest possible accuracy.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Cristian Vilar https://orcid.org/0000-0002-1167-8322

References

1. Vilar C, Thörnberg B and Krug S. Evaluation of embedded camera systems for autonomous wheelchair. In: *Proceedings of the 5th international conference on vehicle technology and intelligent transport systems, publications*, SCITEPRESS—Science and Technology, 2019, pp. 76–85.
2. Dupre R and Argyriou V. 3D Voxel HOG and risk estimation. In: *2015 IEEE international conference on digital signal processing (DSP)*, Singapore, 21–24 July 2015, pp. 482–486.
3. Dalal N and Triggs B.Histograms of oriented gradients for human detection. In: *International conference on computer vision & pattern recognition (CVPR '05)*, *IEEE Computer Society*, San Diego, CA, USA, 20–25 June 2005, pp. 886–893.
4. Wu Z, Song S, Khosla A, et al. 3D ShapeNets: a deep representation for volumetric shapes. In: *2015 IEEE conference on computer vision and pattern recognition (CVPR)*, Boston, MA, USA, 7–12 June 2015, pp. 1912–1920.
5. Liu Z, Zhao C, Wu X, et al. An effective 3D shape descriptor for object recognition with RGB-D sensors. *Sensors* 2017; 17: 451.
6. Bo L, Ren X and Fox D. Depth kernel descriptors for object recognition. In: *IEEE international conference on intelligent robots and systems*, 25 September 2011, pp. 821–886.
7. Schwarz M, Schulz H and Behnke S. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In: *2015 IEEE international conference on robotics and automation (ICRA)*, Seattle, WA, USA, 26–30 May 2015, pp. 1329–1335.
8. Socher R, Huval B, Bhat B, et al. Convolutional-recursive deep learning for 3D object classification. In: *Advances in neural information processing systems*. ACM Digital Library, 2012, pp. 656–664. https://dl.acm.org/doi/10.5555/2999134.2999208.
9. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, USA, 27–30 June 2016, pp. 770–778.
10. Li L, Hoe JKE, Yan S, et al. ML-fusion based multi-model human detection and tracking for robust human-robot interfaces. In: *2009 Workshop on applications of computer vision, WACV 2009*, Snowbird, UT, USA, 7–8 December 2009, pp. 1–8.
11. Simonyan K and Zisserman A. Very deep convolutional networks for large-scale image recognition. *CoRR* 2014.
12. Kasaei H. OrthographicNet: a deep learning approach for 3d object recognition in open-ended domains. *Arxiv, abs/1902.03057*, 2019.
13. Su H, Maji S, Kalogerakis E, et al. Multi-view convolutional neural networks for 3D shape recognition. In: *2015 IEEE international conference on computer vision (ICCV)*, Santiago, Chile, 7–13 December 2015, pp. 945–953.
14. Shi B, Bai S, Zhou Z, et al. DeepPano: deep panoramic representation for 3-d shape recognition. *IEEE Signal Process Lett* 2015; 22: 2339–2343.
15. Johns E, Leutenegger S and Davison AJ. Pairwise decomposition of image sequences for active multi-view recognition. In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, 2016, pp. 3813–3822.
16. Sfikas K, Theoharis T and Pratikakis I. Exploiting the PANORAMA representation for convolutional neural network classification and retrieval. In: *Eurographics workshop on 3D object retrieval, The Eurographics Association*, 23 April 2017, pp. 1–7.
17. Yavartanoo M, Kim EY and Lee KM. SPNet: deep 3d object classification and retrieval using stereographic projection. In: *Lecture notes in computer science*, Cham, 2 December 2018, pp. 691–706. Cham: Springer.
18. Zhi S, Liu Y, Li X, et al. Toward real-time 3D object recognition: a lightweight volumetric CNN framework using multitask learning. *Comput Graphic* 2018; 71: 199–207.
19. Caglayan A and Can AB. Volumetric object recognition using 3-D CNNs on depth data. *IEEE Access* 2018; 6: 20058–20066.

20. Hegde V and Zadeh R. FusionNet: 3D object classification using multiple data representations. *arxiv.org/abs/1607.05695* 2016.
21. Krizhevsky A, Sutskever I and Hinton GE. ImageNet classification with deep convolutional neural networks. In: *Advances in neural information processing systems 25*, New York: Curran Associates, Inc.
22. Brock A, Lim T, Ritchie JM, et al. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv:1608.04236* 2016.
23. Maturana D and Scherer S. VoxNet: a 3D convolutional neural network for real-time object recognition. In: *2015 IEEE/rsj international conference on intelligent robots and systems (IROS)*, Hamburg, Germany, 28 September–2 October 2015, pp. 922–928.
24. Qi CR, Su H, Mo K, et al. PointNet: deep learning on point sets for 3D classification and segmentation. In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017, pp. 652–660.
25. Riegler G, Ulusoy AO and Geiger A. OctNet: learning deep 3D representations at high resolutions. In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017, pp. 3577–3586.
26. Su H, Jampani V, Sun D, et al. SPLATNet: sparse lattice networks for point cloud processing. In: *2018 IEEE/CVF conference on computer vision and pattern recognition*, 2018, pp. 2530–2539.
27. Klokov R and Lempitsky V. Escape from cells: deep KD-networks for the recognition of 3d point cloud models BT. In: *IEEE international conference on computer vision (ICCV)*, 2017, pp. 863–872.
28. Guo Y, Bennamoun M, Sohel F, et al. 3D object recognition in cluttered scenes with local surface features: a survey. *IEEE Trans Pattern Anal Mach Intell* 2014; 36: 2270–2287.
29. Zhang L, João M, Ferreira A, et al. Survey on 3D shape descriptors. *Fundação para a Cincia e a Tecnologia* 2007; 3.
30. Bekkari A and Mammass D. 3D objects descriptors methods: overview and trends. In: *2017 international conference on advanced technologies for signal and image processing (ATSIP)*, Fez, Morocco, 22–24 May 2017, pp. 1–9.
31. Kuang Z, Yu J, Zhu S, et al. Effective 3D shape retrieval by integrating traditional descriptors and point convolution. *IEEE Trans Multimed* 2019; 21: 3164–3177.
32. Kuang Z, Yu J, Zhu S, et al. Using Spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans Pattern Anal Mach Intell* 1999; 21: 433–449.
33. Rusu RB, Holzbach A, Beetz M, et al. Detecting and segmenting objects for mobile manipulation. In: *2009 IEEE 12th international conference on computer vision workshops, ICCV workshops*, Kyoto, Japan, 27 September–4 October 2009, pp. 47–54.
34. Knknoan J, Prasad M, Willems G, et al. Hough transform and 3D SURF for robust three dimensional classification. In: *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, 2010, Berlin, Heidelberg: Springer.
35. Kasaei SH, Tomé AM, Lopes LS, et al. GOOD: a global orthographic object descriptor for 3D object recognition and manipulation. *Pattern Recognit Lett* 2016; 83: 312–320.
36. Wohlkinger W and Vincze M. Ensemble of shape functions for 3D object classification. In: *2011 IEEE international conference on robotics and biomimetics*, Karon Beach, Phuket, Thailand, 7–11 December 2011, pp. 2987–2992.
37. Scherer M, Walter M and Schreck T. Histograms of oriented gradients for 3d object retrieval. In: *18th international conference in central europe on computer graphics, visualization and computer vision, WSCG 2010—In co-operation with EUROGRAPHICS, Full Papers Proceedings*, Plzen, Czech Republic, 1–4 February 2010, pp. 41–48.
38. Vilar C, Krug S and Thornberg B. Rotational invariant object recognition for robotic vision. In: *3rd international conference on automation, control and robots*, Prague, Czech Republic, 11 October 2019, pp. 1–6.
39. Sheela KG and Deepa SN. Selection of number of hidden neurons in neural networks in renewable energy systems. *J Sci Ind Res* 2014; 73: 686–688.
40. Shibata K and Ikeda Y. Effect of number of hidden neurons on learning in large-scale layered neural networks. In: *ICCAS-SICE 2009—ICROS-SICE international joint conference 2009, proceedings*, Fukuoka, Japan, 18–21 August 2009, pp. 5008–5013.
41. Kanezaki A, Matsushita Y and Nishida Y. RotationNet: joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: *The IEEE conference on computer vision and pattern recognition (CVPR)*, Salt Lake City, UT, USA, 18–23 June 2018, pp. 5010–5019.