

# Self-Learning 3D Object Classification

Jens Garstka and Gabriele Peters

*Human-Computer Interaction, Faculty of Mathematics and Computer Science,  
FernUniversitt in Hagen - University of Hagen, D-58084 Hagen, Germany*

**Keywords:** Active Vision, Active Learning, Object Classification, 3D Feature Descriptors, Reinforcement Learning.

**Abstract:** We present a self-learning approach to object classification from 3D point clouds. Existing 3D feature descriptors have been utilized successfully for 3D point cloud classification. But there is not a single best descriptor for any situation. We extend a well-tried 3D object classification pipeline based on local 3D feature descriptors by a reinforcement learning approach that learns strategies to select point cloud descriptors depending on qualities of the point cloud to be classified. The reinforcement learning framework learns autonomously a strategy to select feature descriptors from a provided set of descriptors and to apply them successively for an optimal classification result. Extensive experiments on more than 200.000 3D point clouds yielded higher classification rates with partly more reliable results than a single descriptor setting. Furthermore, our approach proved to be able to preserve classification strategies that have been learned so far while integrating additional descriptors in an ongoing classification process.

## 1 INTRODUCTION

An important step towards an effective scene understanding is a reliable object classification. A central requirement for classification algorithms is their invariance to varying conditions such as location, scale, pose, partial occlusion, or lighting conditions. The basic approaches to extract useful information from images for either object recognition or classification are similar and there are a lot of surveys available (Sun et al., 2006; Li and Allinson, 2008; Andreopoulos and Tsotsos, 2013; Loncomilla et al., 2016).

While initially the application scenarios had a strong focus on facial recognition, the spectrum became significantly more diverse with the introduction of reliable local 2D feature descriptors. Finally, due to the results of deep convolutional neural networks (Ciresan et al., 2012), the research in this area received new impetus that continues to the present day. However, there are some cases in which the previously mentioned image-based approaches to object classification do not work properly on principle. This applies to situations where no structured image information is available, e.g., when the lighting is insufficient or the objects are monochrome and due to their shape without sufficient shading. Figure 1 shows two examples that reflect such situations, where additional

3D information in form of 3D point clouds could help to improve object classification.



Figure 1: Examples of objects where a classification solely on the basis of color information could be difficult. (left image: Francis Tiangson, Pinterest).

In this work we present an approach to object classification based on local feature descriptors for 3D point clouds. Particularly, the focus is not on a single local 3D feature descriptor, since the 3D object classification results of any single descriptor vary a lot depending on the density and structure of a 3D point cloud. Instead, the approach presented is essentially based on a machine learning method. We use a reinforcement learning framework to adaptively select and apply different 3D feature descriptors depending on the 3D point cloud to be classified. This leads to a clear improvement of the classification results compared to the use of single local 3D feature descriptors only. The paper is divided into the following sections:

Section 2 provides an overview of the methods relevant in the context of this work. Section 3 introduces our proposed approach and describes its components in detail. Section 4 describes the experiments and intermediate results. The final 3D object classification results are summarized in Section 5. Finally, Section 6 provides an outlook to future extensions and adaptations of the presented approach. The approach proposed in this work including all parameters of the model and extended experiments are described in detail in (Garstka, 2016).

## 2 RELATED WORK

This section starts with a brief overview of related keypoint detectors (subsection 2.1) and local 3D feature descriptors for 3D point clouds (subsection 2.2). The majority of currently available algorithms for keypoint detection and local 3D feature descriptors are summarized in a survey of 3D object recognition methods by (Guo et al., 2014). Thus, only algorithms and approaches that are relevant for our work are quoted below. This is followed by a subsection on classification approaches for 3D point clouds (subsection 2.3) and a short introduction of *Q*-Learning as a reinforcement learning technique we utilize for our approach (subsection 2.4).

### 2.1 Keypoint Detectors

Computational costs of local 3D feature descriptors are mostly high. It does not make sense to compute feature vectors for all points of a point cloud. We use a keypoint detector to reduce the number of feature vectors. Based on the evaluations of (Salti et al., 2011) and (Filipe and Alexandre, 2014) we use the keypoint detector introduced in context of the shape signature feature descriptor (ISS) by (Zhong, 2009).

### 2.2 Local 3D Feature Descriptors

In this subsection, we give an overview of those local 3D feature descriptors used in the context of this work. In short, we use five local 3D feature descriptors for 3D point clouds: The spin image (SI) introduced by (Johnson and Hebert, 1998) is a 2D histogram that is rotated around the normal vector of a point. The point feature histogram (PFH) and the fast point feature histogram (FPFH) collect information from a local environment based on the so-called Darbeaux frame. Both were introduced by (Rusu et al., 2008). The signature of histograms of orientations (SHOT) by (Tombari et al., 2010b) is a set

of histograms of angles determined for multiple segments of a spherical environment. The values of these histograms are concatenated to a signature. And finally the unique shape context (USC) by (Tombari et al., 2010a), which is a normal aligned spherical histogram. These local 3D feature descriptors have been selected because of their broad spectrum of different properties. Two approaches, SI and USC use histograms, PFH and FPFH create signatures using surface properties and SHOT is a hybrid solution of histograms and surface properties. The dimensions of the feature descriptions should cover the largest possible range, from FPFH with 33 dimensions to USC with 1960 dimensions. In the same way the speed of the descriptors should cover a large range, from SI, which is the fastest to PFH which is more than 1000 times slower. Finally, three of the descriptors require a local reference frame (SHOT, SI, and USC) and the others do not (FPFH and PFH).

### 2.3 3D Classification Approaches

A common way to classify an object based on a given set of local feature descriptions consists of two steps. The first step is inspired by text categorization approaches, e.g., (Joachims, 1998). This so-called *bag-of-words* representation has become an eligible method for categorizing visual content. An early approach is the visual categorization with bags of keypoints (Csurka et al., 2004). The basic approach consists of mapping high-dimensional vectors, whose values are usually continuous, to a finite set of quantized representatives. These representatives form the so-called *visual vocabulary*. A histogram in the same size as the vocabulary is used to count the mapped feature descriptions and is called *frequency histogram*. Therefore, the method is often referred to as a *bag-of-features* in this context. In a second step these frequency histograms are used as input vectors for classifiers. Support vector machines (SVM) are often used as a classifier. Primarily, SVMs are binary classifiers. Therefore, each object class requires its own SVM. The frequency histogram is then applied separately to the SVM of each object class. There are numerous approaches that follow this basic principle, e.g., (Madry et al., 2012; Yang et al., 2014).

### 2.4 Reinforcement Learning – Q-Learning

A reinforcement learning (RL) system consists of an *agent* that interacts with an *environment*. Based on the current *state*  $s_t$  of the environment at time  $t$  the agent decides with respect to the learned experience what

action  $a_t$  will be performed next. The experience arises from consequences the agent undergoes within the environment, i.e., a positive or negative *reward*  $r_{t+1}$ , which reflects whether the action  $a_t$  was appropriate to bring the agent closer to its *goal* (Sutton and Barto, 1998).  $Q$ -learning (Watkins and Dayan, 1992) is an algorithm to solve the reinforcement problem for a finite number of discrete states of a fully observable environment. It determines values which describe the quality of an action  $a \in \mathcal{A}(s)$  for a state  $s \in \mathcal{S}$ . These so-called  $Q$ -values can be used for a mapping  $\pi$  called *policy* between the current state and the next action.  $Q$ -learning updates its  $Q$ -values using action  $a_t$  in state  $s_t$  at time  $t$  while observing  $Q$ -values of the next state  $s_{t+1}$  and the immediate reward  $r_{t+1}$ :

$$Q(s_t, a_t) = Q(s_t, a_t) +$$

$$\alpha \left[ r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right], \quad (1)$$

where  $\alpha$  is a parameter that controls the learning rate and  $\gamma$  is the *discount rate* with  $0 \leq \gamma \leq 1$ . The latter determines how strongly immediate rewards are weighted compared to future rewards. In this way the *return*, i.e., the total discount of future rewards, for state-action pairs is estimated. Initially, all  $Q$ -values are initialized with a constant value, typically zero. In this phase, the  $Q$ -values cannot be used for decisions. Therefore, the actions are typically selected randomly with a probability  $\epsilon = 1.0$ . Or in other words: the RL agent follows a *random policy*. This phase is called *exploration phase*. As soon as the  $Q$  values get more stable, the portion of randomly selected actions  $\epsilon$  is successively reduced. This is the transition to the *exploitation phase* where the RL agent follows a so-called  $\epsilon$ -*greedy policy*. As long as  $\epsilon \neq 0$ ,  $Q$ -learning can react to changes in the behavior of the environment and makes adjustments to the  $Q$ -values.  $Q$ -learning is proven to converge to an optimal value, which means that the best possible way to solve a given task can be dictated by taking actions greedily with respect to the learned  $Q$ -values (*greedy-policy*) (Watkins and Dayan, 1992).

### 3 METHODS

In our approach several local 3D feature descriptors are autonomously selected and applied to a 3D point cloud to be classified. This is not achieved by a one-time optimization of the classification pipeline but by a continuous learning process, which has been implemented in the form of a RL framework (Sutton and Barto, 1998). The basic structure of the classification pipeline is presented in Subsection 3.1 and is ex-

tended with the RL framework as described in Subsection 3.3. Subsection 3.4 shows how the adaptiveness of  $Q$ -learning is adopted to add local 3D point cloud descriptors dynamically during ongoing classification processes.

#### 3.1 Classification Pipeline

The structure of the basic classification pipeline is schematically shown in Figure 2. Within this pipeline some parameters have to be defined in advance. We take these parameters from an evaluation described in (Garstka and Peters, 2016). The dataset used in the

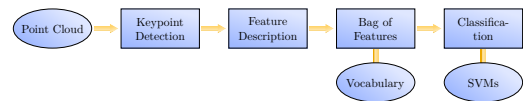


Figure 2: The structure of the basic classification pipeline, which is extended later by a RL approach.

context of this work is the RGB-D Object Dataset of the University of Washington (Lai et al., 2011). The dataset contains 51 object classes with 300 different objects where each object was captured in different poses, resulting in 207841 distinct point clouds, or approximately 4000 point clouds per object class on average. We use only the 3D point cloud data of the dataset for our experiments. Apart from the complete set of 3D point clouds, we also use a reduced set of 3D point clouds, which only consists of 10 of the 51 object classes (see Figure 3).



Figure 3: One view of one object for each of the 10 selected object classes. These are left to right, top to bottom: cap, coffee mug, food bag, greens, hand towel, keyboard, kleenex, notebook, pitcher, and shampoo.

The classification pipeline starts with a keypoint detection algorithm. As already noted in Section 2.1, we use the ISS keypoint algorithm introduced by Zhong. The average number of keypoints determined by ISS on the given set of point clouds is approx. 131. In the second step the basic classification pipeline continues with the computation of a local 3D feature description at each keypoint. The descriptors used have been presented in Section 2.2. Next, the calculated feature descriptions are sorted into the bag of features frequency histogram. Beforehand, the number of visual words of the vocabulary, i.e., the num-

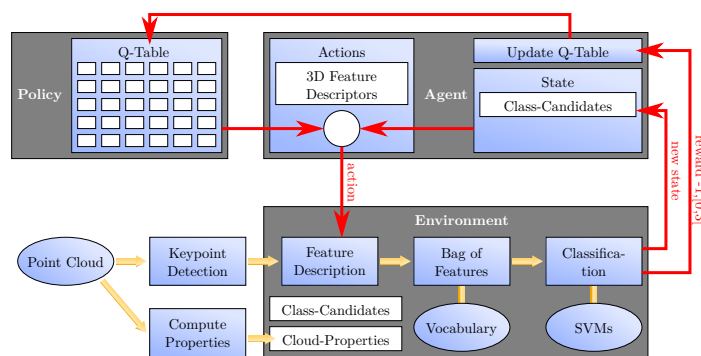


Figure 4: Extension of the classification pipeline with a RL framework.

ber of bins of the frequency histogram, has to be defined. Within the scope of preceding evaluations we obtained best results with vocabulary sizes of 50 for SI, 100 for PFH, FPFH and SHOT, and 200 for USC, which are also used for the experiments described in Section 4. The vocabulary is determined using *k*-means++ (Arthur and Vassilvitskii, 2007) with an Euclidean distance.

The last step of the pipeline is the classification. Each object class is bound to a set of corresponding SVMs, one for each local 3D feature descriptor. The kernel function used in SVMs is a Gaussian radial basis function. The best parameter values are taken from an evaluation described in (Garstka and Peters, 2016) quoted above: the kernel parameter  $\gamma = 0.008$  and the SVM penalty parameter  $C = 125$ . All SVMs have been trained using every second point cloud of the corresponding object class as positive example, which are  $\approx 2000$  3D point clouds. The double amount of  $\approx 4000$  randomly selected 3D point clouds from all other object classes have been used as negative examples.

### 3.2 Reference Values

In order to obtain an average classification rate for each local 3D feature descriptor, the basic classification pipeline has been applied for each local 3D feature descriptor separately to all 3D point clouds, which were not used for the training of the SVMs. In the following this setting is referred to as *single descriptor setting*. The results can be divided into 3 cases:

1. Exactly for one object class the prediction value of its SVM is positive, and it is the correct object class. This case is hereinafter referred to as an *exact match*.
2. For several object classes the prediction values of their SVMs are positive, but the object class the SVM of which has the highest prediction value

corresponds to the correct object class. The latter will be hereinafter referred to as the *best match*.

3. In all other situations the classification fails.

The assignment rates from the first and second case are summarized as *classification rate*. Table 1 shows the classification rates of all individual local 3D feature descriptors while performing the described basic classification pipeline. These values serve as reference values for the subsequent extension of the pipeline with the RL framework. The parameters required for each local 3D feature descriptor were taken from the respective original publication for each descriptor and from the evaluation described in (Garstka and Peters, 2016). In the single descriptor setting the

Table 1: Classification rates in the single descriptor setting. The results shown have been obtained for each local 3D feature descriptor applied separately, on the one hand applying the basic pipeline on all 51 object classes, on the other hand only on the reduced set of 10 object classes.

descriptor	classification rate for	
	51 classes	10 classes
SI	7.4%	23.8%
PFH	6.0%	62.9%
FPFH	9.4%	65.0%
SHOT	3.6%	22.8%
USC	8.5%	59.7%

portion of the exact class assignments (case 1 of the previously described three cases) among the classification rates is 0% in all cases.

### 3.3 Reinforcement Learning Framework

The extension of the basic classification pipeline with a RL approach is illustrated in Figure 4 and described in Subsubsection 3.3.1 to 3.3.3. To extend the basic classification pipeline with a RL approach the general proceeding is as follows: Beginning with a 3D

point cloud, the first step of the basic pipeline, i.e., the recognition of the keypoints, is performed. Next, the feature descriptions are determined at each keypoint. Instead of using a single descriptor, the RL agent selects one of the available descriptors to determine the feature descriptions. Then the remaining two steps of the pipeline are executed for the selected descriptor. Once the prediction value of the SVM of the respective descriptor corresponding to each object class has been determined, all object classes with a negative prediction value are excluded. The remaining classes are hereinafter referred to as class candidates. The further procedure depends primarily on the set of class candidates. If it contains more than one object class, the classification pipeline resumes at the second step. In this case, the RL agent selects another unused descriptor with which new feature descriptions are determined at the already detected keypoints. Then the remaining steps of the pipeline are executed again. The new prediction values corresponding to the remaining classes are used to further reduce the class candidates. After a few iterations this process ideally ends up with the one matching object class remaining. However, due to the similarity of many object classes, this will rarely be the case. To prevent the selection of all feature descriptors during each classification, a restriction of computation time is introduced. As soon as the time limit is exceeded the object class with the highest prediction value within the remaining class candidates is returned as best matching object class. The implementation of these concepts, the components, and parameters are described in detail in the following.

### 3.3.1 Basic Components of the Framework

The *environment* of the RL framework is defined by the set of class candidates, which at the beginning contains all object classes and additional properties of the input point cloud (see Figure 4). The point cloud properties and their associated discrete values are:

1. The number of keypoints relative to all objects:
  - 1st quartile:  $\rightarrow$  slight structure  $\rightarrow$  1
  - 2nd, 3rd quartile:  $\rightarrow$  medium structure  $\rightarrow$  2
  - 4th quartile:  $\rightarrow$  considerable structure  $\rightarrow$  3
2. The ratio between the two successive eigenvalues of the covariance matrix of the point cloud,  $r_1 = e_2/e_1$  and  $r_2 = e_3/e_2$ , where  $e_1 \leq e_2 \leq e_3$ :
  - $r_1 \leq 3.0 \wedge r_2 \leq 3.0 \rightarrow$  uniform, not flat  $\rightarrow$  1
  - $r_1 \leq 3.0 \wedge r_2 > 3.0 \rightarrow$  elongated, not flat  $\rightarrow$  2
  - $r_1 > 3.0 \wedge r_2 \leq 3.0 \rightarrow$  uniform and flat  $\rightarrow$  3
  - $r_1 > 3.0 \wedge r_2 > 3.0 \rightarrow$  elongated and flat  $\rightarrow$  4

The RL *agent* knows the current *state* of the environment, i.e., the class candidates and the point cloud

properties. Based on the state the agent can perform an *action*, i.e., the selection of a local 3D feature descriptor the agent has not yet applied. The agent needs a *policy* to decide which descriptor should be selected next in the current state. When using *Q*-learning, policies are usually based on the so-called *Q*-table, which contains a value for each pair of state and action that reflects how suitable an action is in a given state. Under the assumption that the *Q*-table contains only optimal *Q*-values, the agent selects the best local 3D feature descriptor for the current state based on the values of the *Q*-table. The selected descriptor leads to a change of the environment, i.e., a reduced set of class candidates. As the *Q*-table is initially empty, it is build up successively during the classification.

### 3.3.2 Terminal States and Rewards

The learning mechanism of RL is based on rewards. In general, a reward can be given for each action  $a$  in a state  $s$ . However, in the context of our classification framework a decision on the success of a classification is possible only after the classification process is terminated. Therefore, all possible terminal states have to be defined first (see Table 2). An *'exact*

Table 2: Summary of rewards for terminal states in our RL framework.  $C$  is the set of class candidates,  $n_C$  the number of all object classes,  $t$  the computation time required to get a result, and  $t_{\max}$  the time limit.

	name	reward
terminal state	1) exact match	$3.0 - t/t_{\max}$
	2a) no actions/match	$2.0 -  C /n_C$
	2b) no actions/miss	$1.0 -  C /n_C$
	3a) timeout/match	$2.0 -  C /n_C$
	3b) timeout/miss	$1.0 -  C /n_C$
	4) fail state	-1.0

*match'* means that the set of remaining class candidates contains only the correct object class while no timeout occurred. In case of the two terminal states distinguished for *'no actions'* no local 3D feature descriptor is left to select, while in case of the two terminal states distinguished for *'timeout'* the computation time limit is exceeded. In both cases *'match'* means that the best matching class is the correct object class while *'miss'* means that the best matching class is an incorrect object class. The fourth case *'fail state'* comprises all cases that are not covered by the previously stated cases, such as the case that the set of remaining class candidates does not contain the correct object class while there are still descriptors left to be selected and no timeout occurred. For 2a) and 3a) the best matching object class is determined from the set of class candidates by the highest sum of the

prediction values over all iterations. In our approach only terminal states allow a statement of success or failure. Thus, all other rewards  $r$  are initially set to a value of 0. The rewards for the six cases described above are shown in Table 2.

### 3.3.3 Time Constraint

As already pointed out, a time constraint is set to prevent the successive application of all available local 3D feature descriptors during a classification. The motivation is that the average computing times for each descriptor measured separately should fit into the preset time interval, while at the same time it should not be possible to apply the whole set of available descriptors within this interval. Table 3 contains the average computation time of each local 3D feature descriptor per keypoint in milliseconds and the average computation time for a whole object assuming an average number of 131 keypoints per object. Based on these results, a value of 10 seconds was set as a limit for all subsequent experiments.

Table 3: Average computation times (Intel Xeon E5630 CPU) for each used descriptor. For the third column an average number of 131 keypoints per object is assumed.

	comp. time:	t(ms)/keypoint	t(s)/object
descriptor	SI	0.0449	< 0.01
	PFH	64.5	8.45
	FPFH	6.69	0.88
	SHOT	0.282	0.04
	USC	9.95	1.30

### 3.3.4 Exploration and Exploitation

At the beginning the  $Q$ -table is initialized with zero values. During the exploration phase the values of the  $Q$ -table converge. In the course of the learning process first the portion of random actions  $\epsilon$  is kept at 1.0, but the learning rate  $\alpha$  is slowly reduced to a value of 0.001. Then it is, together with the portion  $\epsilon$  of randomly selected actions continuously reduced to a value of 0.0 (exploitation phase). In practical terms the exploration phase consists of a total of 100 million classifications reducing the learning rate  $\alpha$  beginning with the 60 millionth classification. After 90 million classifications the random portion  $\epsilon$  is also reduced.

## 3.4 Adaptiveness

As already pointed out in Subsection 2.4 a special property of  $Q$ -learning is its support of on-line learning. Under the assumption  $\alpha > 0$  and  $\epsilon > 0$ ,  $Q$ -learning can react to changes in the environment by

adjusting the  $Q$ -table continuously. We exploit this property to dynamically add new local 3D feature descriptors during ongoing classification processes. This enhances the flexibility of a 3D object classification system considerably, since the strategies to classify objects that have been learned so far, can be preserved and exploited by dynamically adapting them, integrating new descriptors smoothly in the classification process. For this purpose we include complete classifications of known objects with probability  $\epsilon$  while using a random-policy.

## 4 EXPERIMENTS AND INTERMEDIATE RESULTS

The main focus of this section lies on the experiments and their intermediate results. Subsection 4.1 contains a figure to illustrate how the values of the  $Q$ -table converge during the learning process. The final 3D object classification rates that can be achieved with our proposed approach are given in Subsection 4.2. They are summarized well-arranged later in Section 5. Finally, the adaptiveness of our approach is examined in experiments described in Subsection 4.3. The following applies to all illustrations: the dotted blue line illustrates the learning rate  $\alpha$ , while the dashed blue line corresponds to the value of  $\epsilon$ .

### 4.1 Exploration Phase

Figure 5 shows the development of the average of all  $Q$ -values, depending on the number of performed episodes. It takes a few million classifications until the value gradually reaches its maximum. However, when the learning rate  $\alpha$  is slowly decreased from the 60 millionth classification on, it is obvious that the quality of many state-action pairs has been overestimated so that the average  $Q$ -value now declines until it slowly converges to a stable value.

### 4.2 Distribution of Terminal States

Figure 6 shows the distribution of the terminal states that is obtained on the full set of 51 object classes. For a better illustration, only the interval from 80 million to 100 million classifications is displayed. The colored curves represent the percentages of the terminal states as introduced in Section 3.3.2 with the exception of fail states. In addition, the red curve summarizes all positive results, i.e., those results that returned a correct object class in one or another way. It is the sum of the results for the terminal states 'exact match', 'no actions/match', and 'timeout/match'.

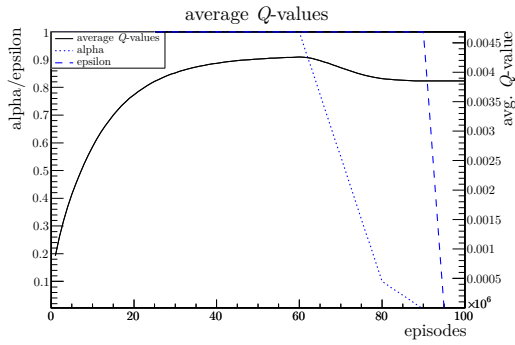


Figure 5: This graph shows the development of the average  $Q$ -value during the exploration phase of the RL framework depending on the number of episodes carried out.

The value of the red curve after the final episode thus represents the final classification results of our RL approach in the setting of 51 object classes. The fi-

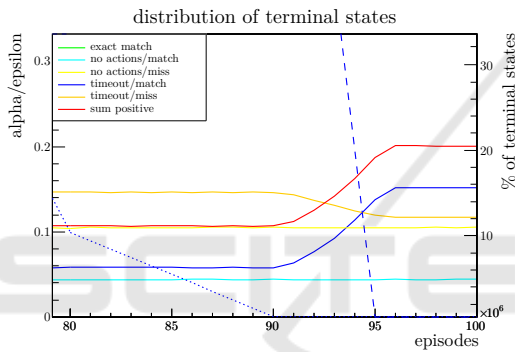


Figure 6: This figure shows the distribution of the terminal states obtained on the full set of 51 object classes depending on the number of performed episodes. The red curve represents correct 3D object classifications. By reducing  $\alpha$  and  $\epsilon$  to a value of zero, the final results of the last episode are based on a greedy policy and represent the best, learned strategy to select the descriptors for 3D object classification.

nal distribution of the terminal states after the last episode, represented by the endpoints of the curves shown in Figure 6, is summarized in Table 4, supplemented by the percentage of fail states. This table also contains the results for the reduced set of 10 object classes.

### 4.3 Adaptive Learning

To explore the adaptiveness of the RL framework, we perform similar experiments as described in the subsections before. The difference, however, is that the learning process is started with a reduced set of only four of five local 3D feature descriptors. Accordingly, the values of  $\alpha$  and  $\epsilon$  are initially 1.0. After 25 million episodes the learning rate  $\alpha$  is reduced to a value of 0.1. In this way, a potential over-fitting is compensated. After 45 million episodes  $\epsilon$  is also reduced

Table 4: Distribution of terminal states after the last episode with 'sum of positive results' summarizing the percentages for the states 'exact match', 'no actions/match', and 'timeout/match' and thus representing the final 3D object classification rate of our approach.

terminal state	percentage	percentage
	51 classes	10 classes
exact match	0.0%	16.0%
no actions/match	5.4%	0.4%
no actions/miss	11.0%	0.2%
timeout/match	16.1%	58.6%
timeout/miss	11.6%	8.5%
fail state	55.9%	16.3%
sum of positive results	21.5%	75.0%

to a value of 0.1. From this moment on the reinforcement learning agent is in an exploitation phase where classification rates are reasonably high, even if 10% of the actions are performed as random descriptor selections. Figure 7 illustrates the adaptiveness of our approach for the case that the FPFH descriptor (Rusu et al., 2009) is omitted at the beginning, using the reduced set of 10 object classes. Without FPFH the classification rate is  $\approx 57\%$  (episode 45-50 million). After 50 million episodes, FPFH is added

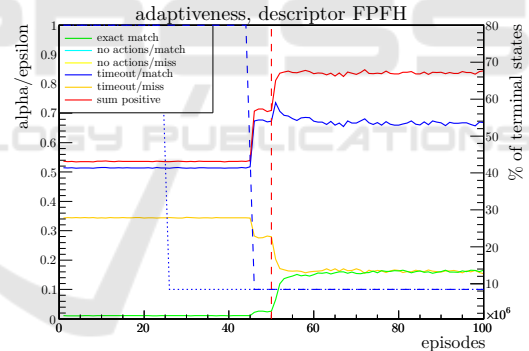


Figure 7: This graph illustrates the adaptiveness of our approach adding FPFH to the set of descriptors (at 50 million episodes). The curves represent the distribution of the terminal states depending on the number of performed episodes.

to the list of available descriptors. This is indicated by the red dashed line in Figure 7. Immediately afterwards the number of correct classifications increases significantly within a few episodes and stabilizes to the value of about  $\approx 67\%$ . The difference of  $\approx 67\%$  to the result of 75% that was reported in Section 4.2 is due to the increased portion of random descriptor selections and the adapted learning rate. If both,  $\alpha$  and  $\epsilon$ , were set to zero the results would be identical.

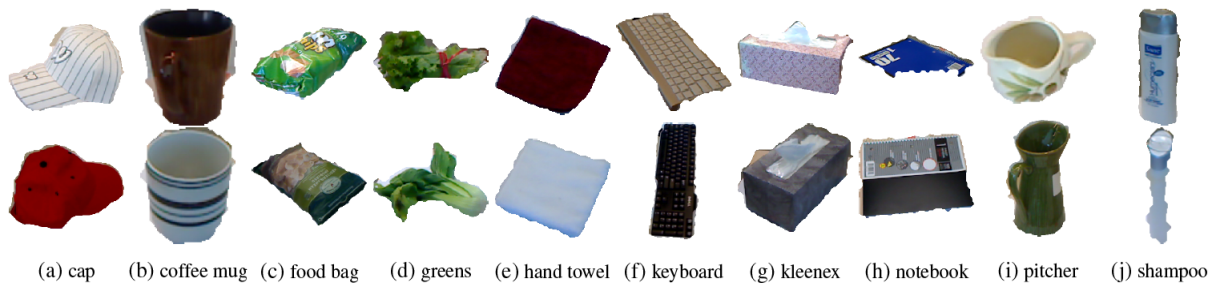


Figure 8: These images show examples from the reduced set of 10 object classes. The upper row shows images from an object in a specific pose where classification yielded the best result, while the lower row shows an object from the same object class where classification yielded the worst result. The overall geometry for most of the classes is similar for best and worst matches with a few exceptions, the most obvious are the classes 'greens' and 'pitcher'.

## 5 3D OBJECT CLASSIFICATION RESULTS

The final 3D object classification rates that can be achieved with our proposed approach of learning strategies to select point cloud descriptors are summarized concisely in Table 5. The classification rate

Table 5: Gain of the proposed approach in terms of classification rates.

classification rates for:	51 classes	10 classes
proposed approach	21.5%	75.0%
single descriptor setting	9.4%	65.0%

of 21.5% that has been achieved with our approach on the full set of 51 object classes, for example, is the sum of the percentages for the terminal states 'exact match', 'no actions/match', and 'timeout/match' (see Table 4). This value has to be compared with the highest classification rate that can be achieved in the single descriptor setting. This classification rate has been provided by the FPFH descriptor with 9.4% (see Table 1). Thus, the classification rate could be more than doubled with our approach in the case of 51 object classes. In the case of the reduced set of 10 object classes the classification rate of 75.0% achieved with our approach (see Table 4) is an improvement of 10 percentage points compared to the best classification rate of 65.0% which could be achieved with FPFH (see Table 1). Furthermore, the classification rate of 75.0% contains a share of 16.0% exact class assignments (see Table 4, 'exact match'), whereas the single descriptor setting across all descriptors did not yield any exact assignment at all (see Subsection 3.2). This means that the results obtained within the proposed RL framework are partly also more reliable than those obtained in the single descriptor setting. Figure 8 shows examples for success and failure cases. The object instances and poses shown in the upper row correspond to point clouds where the classifica-

tion yields the best results, while the point cloud of the object instances and poses shown in the bottom row lead to the worst classification results. With the exception of the object classes 'greens' and 'pitcher' the proposed method seems to impose no bias in the sense that point clouds with distinctly different geometry (in comparison to other instances of their class) are systematically classified worse. This argumentation can be verified by comparing the images of all object instances from the reduced set of 10 object classes given in the supplemental material.

## 6 CONCLUSION AND OUTLOOK

We presented a self-learning approach to object classification from 3D point clouds. We extended an approved 3D object classification pipeline based on local 3D feature descriptors by a reinforcement learning approach that learns strategies to select point cloud descriptors depending on qualities of the point cloud to be classified. The reinforcement learning framework is provided with a number of 3D feature descriptors and learns autonomously via trial and error a strategy to select and apply them successively for an optimal classification result. Thus, the classification process does not follow a rigid scheme anymore, but dynamically adapts its classification strategy to a changing environment. Our experiments demonstrated that this approach is able to provide higher classification rates in comparison to results obtained in rigid scheme classification settings. In addition, some of the results turned out to be more reliable. With few exception the proposed method seems to impose no bias in the sense that point clouds with distinctly different geometry (in comparison to other instances of their class) are systematically classified worse. A special advantage of the reinforcement learning framework consists in its flexibility and



adaptiveness. The latter allows for the subsequent integration of additional 3D feature descriptors while the system is already running in an application scenario. Our approach proved to be able to preserve classification strategies that have been learned so far and at the same time to smoothly integrate new descriptors in already learned strategies. The adaptiveness of the proposed self-learning approach enhances the flexibility of a 3D object classification system considerably, as new feature descriptors will be developed in the future and the learning process for a special application scenario does not have to be started from scratch again.

## REFERENCES

- Andreopoulos, A. and Tsotsos, J. K. (2013). 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8):827–891.
- Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035.
- Ciresan, D. C., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, USA, June 16-21, 2012*, pages 3642–3649.
- Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague.
- Filipe, S. and Alexandre, L. A. (2014). A comparative evaluation of 3d keypoint detectors in a RGB-D object dataset. In *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications, Volume 1*, pages 476–483.
- Garstka, J. (2016). *Learning strategies to select point cloud descriptors for large-scale 3-D object classification*. PhD thesis, FernUniversität in Hagen.
- Garstka, J. and Peters, G. (2016). Evaluation of local 3-d point cloud descriptors in terms of suitability for object classification. In *ICINCO 2016 - 13th Int. Conf. on Informatics in Control, Automation and Robotics, Volume 2*.
- Guo, Y., Bennamoun, M., Sohel, F. A., Lu, M., and Wan, J. (2014). 3d object recognition in cluttered scenes with local surface features: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(11):2270–2287.
- Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98, 10th European Conference on Machine Learning*, pages 137–142.
- Johnson, A. E. and Hebert, M. (1998). Surface matching for object recognition in complex three-dimensional scenes. *Image Vision Comput.*, 16(9-10):635–651.
- Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE International Conference on Robotics and Automation*, pages 1817–1824.
- Li, J. and Allinson, N. M. (2008). A comprehensive review of current local features for computer vision. *Neurocomputing*, 71(10-12):1771–1787.
- Loncomilla, P., Ruiz-del-Solar, J., and Martínez, L. (2016). Object recognition using local invariant features for robotic applications. *Pattern Recognition*, 60:499–514.
- Madry, M., Ek, C. H., Detry, R., Hang, K., and Kragic, D. (2012). Improving generalization for 3d object categorization with global structure histograms. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1379–1386.
- Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms for 3d registration. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*, pages 3212–3217.
- Rusu, R. B., Blodow, N., Marton, Z. C., and Beetz, M. (2008). Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008, Acropolis Convention Center, Nice, France*, pages 3384–3391.
- Salti, S., Tombari, F., and Stefano, L. D. (2011). A performance evaluation of 3d keypoint detectors. In *3D Imaging, Modeling, Processing, Visualization and Transmission, 2011 International Conference on*, pages 236–243. IEEE.
- Sun, Z., Bebis, G., and Miller, R. (2006). On-road vehicle detection: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(5):694–711.
- Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9(5):1054–1054.
- Tombari, F., Salti, S., and Di Stefano, L. (2010a). Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 57–62. ACM.
- Tombari, F., Salti, S., and di Stefano, L. (2010b). Unique signatures of histograms for local surface description. In *11th European Conference on Computer Vision (ECCV), 2010, Proceedings, Part III*, pages 356–369.
- Watkins, C. J. C. H. and Dayan, P. (1992). Technical note q-learning. *Machine Learning*, 8:279–292.
- Yang, Y., Yan, G., Zhu, H., Fu, M., and Wang, M. (2014). Object segmentation and recognition in 3d point cloud with language model. In *Int. Conf. Multisensor Fusion & Information Integration for Intelligent Systems*, pages 1–6.
- Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *12th International Conference on Computer Vision (ICCV Workshops)*, pages 689–696. IEEE.