Robust Multi-Algorithm Object Recognition Using Machine Learning Methods

Tobias Fromm, Benjamin Staehle, and Wolfgang Ertel

***Abstract*— Robust object recognition is a crucial requirement for many robotic applications. We propose a method towards increasing reliability and flexibility of object recognition for robotics. This is achieved by the fusion of diverse recognition frameworks and algorithms on score level which use characteristics like shape, texture and color of the objects. Machine Learning allows for the automatic combination of the respective recognition methods' outputs instead of having to adapt their hypothesis metrics to a common basis. We show the applicability of our approach through several real-world experiments in a service robotics environment. Great importance is attached to robustness, especially in varying environments.**

I. Introduction

In the field of robotics, object recognition and perception in general are some of the most essential topics as they are crucial for further research on high-level behavior planning. Especially the area of service robotics is dependent on robust methods to determine the type, position and orientation of objects to interact with. There is a huge diversity of approaches in image processing and object recognition, but most of them are highly focused on certain types of objects and therefore likely to fail on many real-world applications. Examples of such failures are texture-based algorithms on unicolored items or infrared light-based sensors on transparent ones. Recognizing a greater variety of objects differing in shape, color, texture and material therefore demands a combination of specialized sensors and algorithms and the integration of their results.

Combining the results of varying recognition algorithms is known to bear some risks and difficulties, though. There are basically two possibilities to tackle these: Firstly, the algorithms' hypotheses can be unified in some way for direct comparison. If done this way, there also needs to be some assessment of the algorithms' applicability on the concrete problem. Secondly, result unification and applicability evaluation can be performed automatically by using Machine Learning (ML) methods for handling both of these tasks.

The great advantage of the second approach is the adaptability of virtually any algorithm to the recognition process. If done this way, the user does not need to adapt the algorithm's output to any pre-defined scheme, only a ranking of hypotheses needs to be produced. Each of these hypotheses also needs to have some arbitrary score or distance measure attached. Everything else will be done by the ML fusion part which, in the end, will present a final ranking of object hypotheses combined from all algorithms' results.

The authors are with the Institute of Artificial Intelligence, Ravensburg-Weingarten University of Applied Sciences, 88250 Weingarten, Germany `lastname@hs-weingarten.de`

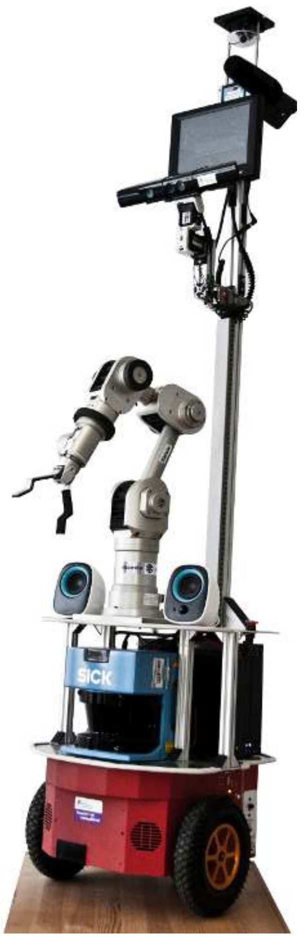
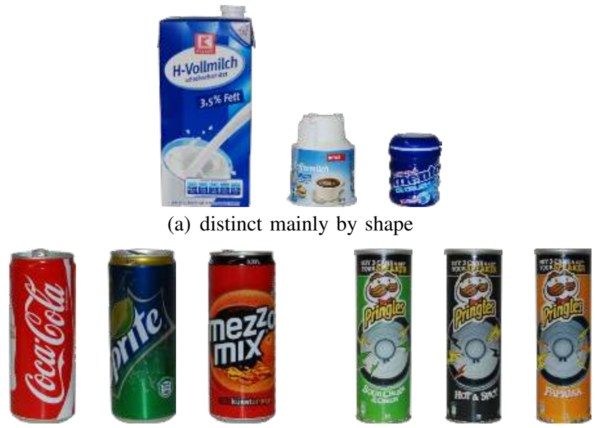


Fig. 1 – *Kate*, the service robot providing the platform for the application of the proposed method.

As special value is placed on the robustness and real-world applicability of our approach, the process in this paper was implemented on our real service robot *Kate* (Figure 1) which is equipped with a Kinect RGB-D sensor. This enables us to execute several algorithms based on texture, color and point clouds in parallel for recognizing a variety of objects in a typical household environment. We emphasize the diversity of the used objects in the mentioned criteria, but at the same time, pay attention to include objects with similar characteristics as well (see the examples in Figure 2). This way, we are able to conduct a more consistent evaluation of our approach in terms of robustness since rather similar objects cause a bigger challenge for the recognition and fusion algorithms than obviously distinctive ones.


(a) distinct mainly by shape

(b) distinct mainly by texture (c) distinct mainly by color

Fig. 2 – Objects exploiting the advantages of different recognition approaches.

Because of the necessity of evaluating our approach in a real-world domain, it is hard to quantitatively compare its results to any related work as their methodology and range of applications, respectively, does not entirely match with ours. Still, the methodical validity of the basics used in the proposed method was shown in the works we refer to in Section II. After we present the utilized existing object recognition approaches in Section III, in Section IV we describe in detail how our recognition score fusion technique works. Following the evaluation of our approach in Section V we will conclude the paper with future prospects in Section VI.

II. Related Work

In our approach, we benefit from the work of Scheirer et al. [1] who introduce the term *Meta-Recognition* (MR) as a performance prediction method for recognition algorithms. They first describe the theory and prove the plausibility of their concept from a statistical point of view using Extreme Value Theory (EVT). In their recent work [2], MR is extended through the usage of ML to allow for more accurate predictions. The authors show that statistical approaches can be outperformed by far because ML classifiers can make use of a-priori knowledge of a bunch of training examples in contrary to statistical classifiers.

In general, MR is an enhancement of conventional sensor fusion methods like the ones which use the Dempster-Shafer theory, a generalization of the Bayes theory [3]. Successful applications for this theory exist [4], but they rely on the validity of probabilities for object existence.

Other works like [5] concentrate on building filters for the fusion of low-level sensor data of same characteristics which is perceived by different sensors. The weight of these filters is usually adjusted object-independently, so any object-dependent advantages of certain recognition algorithms over others cannot be taken account of.

Gould et al. [6] use machine learning to build object-specific classifiers. Their results sound promising and show the applicability of ML in an object recognition domain quite well. However, they combine raw data and computed features from different sensors, but not high-level recognition score outputs. This method is extended by our approach where the user is relieved from the need of dealing with raw data.

In the work of [7], distances of histograms over the texture, shape and color features detected by some algorithms are used which are then fed into a k-Nearest-Neighbor classifier to identify the object. Unfortunately, there is no comprehensible description of the classification process or the classifier settings which were used in this paper. But still, the overall recognition results look convincing and fuel our efforts towards using an ML approach for our work.

Summarized, all the approaches mentioned so far provide convincing results in their respective domains, but, except for [6] and [7], they are not primarily dealing with an object recognition problem. Furthermore, the method we propose herein provides for an easy-to-use combination of existing recognition algorithms in terms of effort to adapt these into the overall classification system. This stands in contrast to the mentioned methods as none of them enables the user to simply add a new recognition algorithm with very little methodical and implementational effort.

When it comes to the collection of an object gallery used for training and recognition, one can find many promising approaches like the Berkeley 3-D Object Dataset (B3DO) [8] or the one of the University of Washington [9], the latter of which even incorporates tree-like hierarchies. Janoch et al. [8] give an evaluation of some of the other available datasets. In contrast to most of the others, the Berkeley dataset is made of many different instances in a large number of categories. They also emphasize that the data was collected "in the wild" instead of a laboratory setting.

Other examples of standardized object dataset collections are RoboEarth [10] and the KIT ObjectModels Web Database [11] which provide interfaces to a database where any user can input their self-created object models including various kinds of data. This is a very promising approach as real-life objects can be stored in this database with open access and contribution for everyone working in this field. For our work, at the moment there is still the need for high-resolution images of the objects with highly textured backgrounds, though. Unfortunately, this so far prevented us from using one of the described datasets. The object gallery used in our approach as well as its assumptions regarding the input data are described in Section V and III, respectively.

III. Object Recognition Algorithms and Pipeline

In the following, we will briefly describe the object recognition pipeline as well as particular frameworks and algorithms used in the context of our approach. For more details please refer to the respective publications and websites.

However, it is very important to emphasize that the following selection of algorithms is utilized to demonstrate the applicability of our approach, but the approach itself is not limited to the described algorithms. Instead, this selection can easily be extended to contain virtually any object recognition algorithm, not necessarily complying to the categories in this section. We can think of various other algorithms based on 2D and 3D sensor data that may be included in the proposed system.

A. *Point Cloud-Based Matching*

As affordable 3-D sensors, namely the Kinect and its relatives, have started flooding the market, 3-D imaging has become easy to use and affordable. The premier software collection to use in this context is the *Point Cloud Library* (PCL) [12] which represents an effort to address the most prominent areas in the field of 3-D point cloud processing. However, the PCL being defined as a library instead of a framework, it incorporates a lot of state-of-the-art algorithms, but without any limitations on how to use them. This makes it lack reusability of high-level solutions. Due to this fact, in the context of point-cloud based object recognition, we make use of our recently developed framework *BOR3D* [13]. This tool presents a novel approach of 3-D object recognition via extending the PCL's versatile capabilities by the union of ease of use and flexibility.

B. *Texture-Based Matching*

The texture of an object holds valuable information and is another property that can be used for object recognition. In this work we utilize the *MOPED* framework [14] which provides a SIFT-based approach for object recognition and pose estimation. It uses a database of features arranged in 3-D space which were extracted and registered (i.e. transformed into the same coordinate system) from a set of 2-D images. MOPED attempts to classify any objects in the images by comparing the interest points with its database and calculating a score to estimate the distance of the input to any similar database model.

C. *Color-Based Matching*

To distinguish between objects varying mainly by color like in Figure 2(c), we use a color distribution histogram over the object's input image for which the Bhattacharyya distance is computed to the histograms of the respective object model. This histogram is built over *Hue-Saturation-Value* (HSV) color space instead of *Red-Green-Blue* (RGB) because of its superior accuracy on image classification tasks by decoupling brightness from chromatic components [15]. Even more, due to our own assessment, we regard hue and saturation only because the value component has proven to be most susceptible to the change of lighting conditions.

D. *Pre-Processing Pipeline*

In order to aggregate these particular recognition algorithms, the object candidates need to be detected first. In our case, BOR3D provides helpful assistance regarding segmentation and clustering of the 3-D point cloud. We use the standard approach of first computing planes in the input image via RANSAC [16], next removing them and then generating clusters from everything else which are taken as

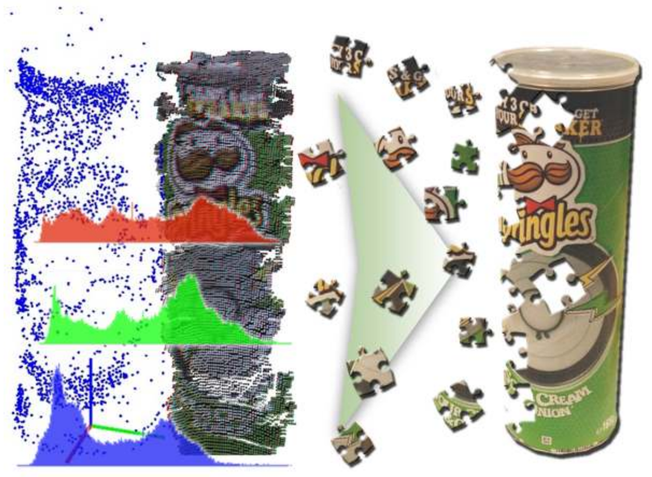


Fig. 3 – Visualization of the score fusion process.

possible objects. These candidates are then being equipped with the RGB image cropped to the respective area so that high-resolution information is retained. Finally, the 3-D point clusters including RGB information are taken as input data for the previously described recognition frameworks.

Summarized, our pre-processing pipeline complies with the standard procedure utilized by many state-of-the-art object recognition approaches:

1) **Gather sensor RGB and depth data.**
2) **Remove planes** from depth data using RANSAC.
3) **Compute clusters** from the remaining 3-D points.
4) **Crop RGB image** according to cluster size and coordinates.

IV. MULTI-ALGORITHM RECOGNITION SCORE FUSION

A. *Terms and Conditions*

First of all, it is important to state that, whenever we use the term *object*, this refers to the model which is stored in the model database. The physical 2-D/3-D data on which the recognition process runs on in the following will be called *candidate*. After running the classifier, one or more *hypotheses* are generated from this candidate which equal the objects the classifier assigns to them together with a *confidence*.

Instead of building a multi-class classifier which takes training samples of all objects, we create a distinct binary classifier for each object because, in case of erroneus training data for a certain object (e.g. incorrectly set class labels), it is easier to keep track of misclassifications if only one object is affected. Distinct classifiers' outputs can be monitored separately with a higher probability to track down the error source.

Another benefit from distinct classifiers is the possibility to build a multi-level object classification system which is able to group, for instance, several flavors of sodas or chips on a lower level, beverages and food on a higher level. Since this may provide an objective for future work, our current approach attempts to lay some architectural foundations.

B. Score Fusion Process

The score fusion is started after the pre-processing pipeline as described in Section III-D has finished. Basically the workflow per candidate can be explained in four major steps:

1) **Run all recognition algorithms separately.** For the results presented in this paper, these are the previously described BOR3D, MOPED, and Color Histograms.
2) **Compute features** from recognition algorithm scores. These features will be explained in detail in the next section.
3) **Perform classification** for each object. The used classifiers and settings are illustrated in Section IV-F.
4) **Select the hypothesis** which returns the highest confidence from its classifier. If no hypothesis exceeds a certain threshold, reject all of them. This threshold will be discussed in Section V-B.

C. Features

For the creation of features which represent the input of the per-object classifiers, we first need to state that these features need to be kept consistent over all classifiers. This ensures the comparability of the individual confidence values. Effectively, keeping similar features over all classifiers maintains consistency as well as the usage of a single, multi-class classifier.

Apart from taking raw recognition scores, it is reasonable to consider certain features which are derived from rankings and distances between them. This way, in compliance with Scheirer et al. [2], the classification results become more invariant against offsets and scaling of the input scores. In this context, we compute different features from the sorted list of results produced by each recognition algorithm.

Hence, we use the top k scores $s_1, ..., s_k$ from the sorted list of algorithm results for the generation of the following set of features:

- the raw recognition scores $s_{1,1}, ..., s_{1,n}, s_{2,1}, ..., s_{m,n}$ of all m objects in the gallery and n algorithms,
- the distances between the top and the $(k-1)$ following scores $\langle \Delta_{1,2}, ..., \Delta_{1,k} \rangle = \langle (s_1 - s_2), ..., (s_1 - s_k) \rangle$ per algorithm,
- the Discrete Cosine Transform (DCT) coefficients of the top k scores per algorithm.

This sums up to a total of

$$mn + (k-1)n + kn = (m + 2k - 1)n \tag{1}$$

features. In order to find the optimal setting of k, experiments showed that the classification quality varies with changing k with a peak at $k = 10$ for our setting. This can be quantified by the F-Measure like described in detail in Section V-A. For $k > 10$, training and classification time rises due to the increased number of features, but there is no improvement on the classification results. In contrary, overfitting may occur in this case which reflects in decreased classification quality.

Except from the raw recognition scores, all other features depend on the ranking which is output by the respective recognition algorithm, but not on one object only. With these features, the classifier also takes the algorithm's class separation quality into account and emphasizes the distribution of scores amongst the respective recognition results. In contrary to this approach, we also evaluated the much simpler approach of using only the raw recognition scores as input features. As expected, this resulted in completely insufficient classifications which is in accordance with [2].

D. Training Phase

Before any productive results can be obtained from a supervised classifier, it needs to be trained with the help of the previously mentioned features. Inputs and outputs used therefore are described in the following:

1) Input: Let m be the number of training samples, n the number of features calculated from the algorithms' output scores, $F_{i,j}$ a distinct feature calculated from algorithm A_j's output, and C_i the binary class label (1 if the respective object is represented by the score combination in this sample, 0 otherwise), then the matrix Y of training input samples is:

$$Y = \begin{bmatrix} F_{1,1} & ... & F_{1,j} & ... & F_{1,n} & C_1 \\ ... & ... & ... & ... & ... & ... \\ F_{i,1} & ... & F_{i,j} & ... & F_{i,n} & C_i \\ ... & ... & ... & ... & ... & ... \\ F_{m,1} & ... & F_{m,j} & ... & F_{m,n} & C_m \end{bmatrix} \tag{2}$$

2) Output: After conducting the training process for a sufficient number of samples from different views and in different poses, the emerging classifier for each object can be used for the classification of objects the system was not able to recognize before. The required number of samples must be determined from the classifier's performance; find more information about this in Section V.

E. Classification Phase

This phase equals productive operation of the proposed fusion approach. Before entering it, the training phase must be completed. Then the following inputs and outputs become valid:

1) Input: Like in the training process, F_j denotes a feature calculated from algorithm A_j's output, and n the total number of features present. The only difference is the missing class label which is the intended classification result. Thus, the classification input vector Y_i is:

$$Y_i = [F_{i,1}, F_{i,2}, ..., F_{i,n}]. \tag{3}$$

2) Output: As the classification result, the classifier will output a binary decision whether the candidate to classify belongs to the learned class (1 if the candidate matches the trained object, 0 otherwise). Additionally, the classifier's confidence in terms of a probability is given by the error rate of the classified sample. This probability can be formalized as

$$P(X_c = X_i | F_c), \tag{4}$$

where X_c denotes the candidate to classify, X_i a specific trained object, $F_c = [F_{c,1}, F_{c,2}, ..., F_{c,n}]$ the feature vector of the candidate combining the features $F_{c,j}$ of all algorithms A_j, and n the total number of features present.

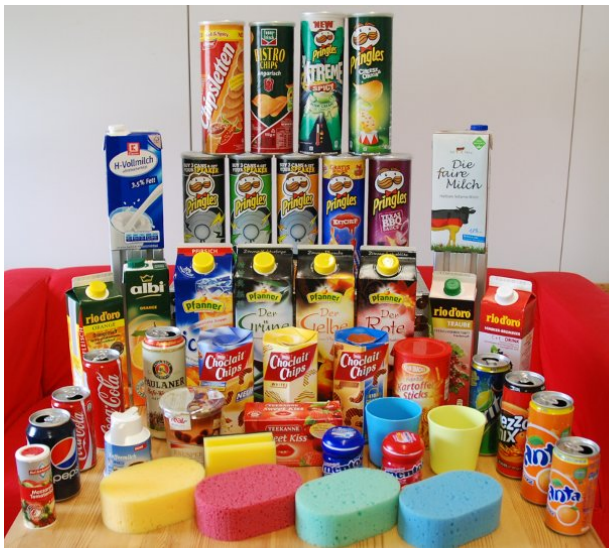


Fig. 4 – Object gallery used in our experiments.

			avg. F-Measure
DT	quality measure	min #records per node	
	gain ratio	5	0.8973
	gain ratio	20	0.9003
	gini index	5	0.8976
	gini index	20	0.8969
NN (2 hid. layers)	#neurons	#iterations	
	40	100	0.9959
	40	200	0.9952
	60	100	0.9966
	60	200	0.9968
SVM (RBF kernel)	gamma	cost	
	5	10	0.9985
	5	25	0.9980
	10	10	0.9932
	10	25	0.9932

TABLE I – Cross validation results of different classifiers on selected parameter sets

F. Classifier Selection

To select a machine learning classifier, one has the choice between a variety of supervised learning algorithms. Due to our experiments conducted in Section V, Support Vector Machines (SVMs) resulted in best performance for our object recognition task. For our experiments, we used the popular LibSVM implementation of Chang and Lin [17].

In general, SVMs impress by their good classification performance and their guarantee to find an optimal solution. This stands in contrast with Neural Networks (NNs), for instance, as these bear the risk of being stuck in a local minimum. NNs also take longer to learn than SVMs. Nevertheless, we also considered using Rprop [18] as an algorithm implementing a Neural Network as well as the C4.5 Decision Tree (DT) [19]. Decision Trees have the advantage of their classification process to be understandable intuitively in contrary to SVMs and NNs, but suffer from worse classification performance.

V. Experiments

We conducted several experiments to evaluate the robustness and performance of our classifiers under different conditions. Like shown in Figure 4, our gallery used for training and recognition consists of 45 objects varying in size, shape, texture and color. We especially emphasize the similarity of some objects in one of these properties while there is diversity in others. Thus, for example, there are several flavors of chips or sodas in a similar packaging, but distinguishable in color or texture.

Our data set was collected from this object gallery under varying conditions regarding scale, view angle and illumination. We recorded about 700 positive samples of each of the 45 individual objects to train our classifiers, which resulted in 30800 negative samples per object. From these, 500 positive and 22000 negative samples per object were used for training the respective classifiers. The remaining 200 positive and 8800 negative samples enabled us to validate the classifier performance on a data set completely independent from the original training set. This will be explained in detail in the next section. As for the number of input features for the classifier, running three algorithms on these 45 objects with $k = 10$ resulted in 192 features according to Equation 1.

For the model creation of our gallery, we set up an automatic turntable which works like in [9], with the Kinect sensor mounted at different heights and collecting data from 360 degrees around the objects.

A. Classifier and Parameters

To consistently verify the classifier and parameter selection over all object classes, we used split-data validation with a fixed partition after running 10-fold cross validation with randomly drawn samples on a variety of parameter sets. This means that, after optimizing the classifier parameters using cross validation, these were validated on the separate testing set. As a metric, the average F-Measure over all objects' classifiers was taken to assess the quality of different classifiers and parameter sets.

The experiment was run on a decision tree with different parameter sets as well as on a neural network and a support vector machine, respectively. Table I shows the resulting average F-Measures using selected parameter sets. Previous experiments on a subset of our training data showed that this selection of all possible parameter sets produces the most promising results. The average cross validation results of all object classifiers using these parameter sets differ only by marginal values of magnitude 10^{-3} as in Table I.

One more important thing to note is the fact that, for practical applicability, training times for the respective classifiers indeed play a role as they differ quite a lot. The difference between a DT which takes a couple of seconds to learn in our setup, an SVM which takes about 30 minutes and a NN which can take up to several hours is a significant factor, especially because we need to train one of them for each object in the gallery.

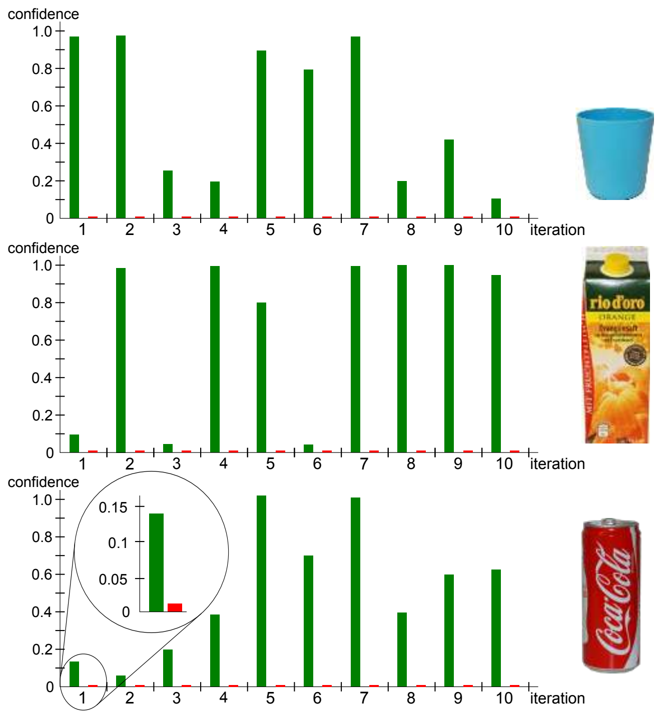


Fig. 5 – Evaluation of the classifier's separation quality by ten iterations on three exemplary objects under varying conditions – the green bars on the left and the red bars on the right of each iteration denote the respective classifier's confidence for the first-ranked and the second-ranked object, respectively. The true objects are shown on the right; they correspond with the green bars. All second-ranked confidence values are less than 0.02.

Hence, against the background of the cross validation results of Table I and having the advantages and drawbacks of the respective classifiers in mind (see also Section IV-F), we decided to use a support vector machine with $gamma = 5$, $cost = 10$ in our system. Afterwards, this configuration was tested on the independent testing data set. This resulted in an average F-Measure of 0.9989, which implies that there were only very occasional misclassifications on the testing data set. However, this number does not directly reflect the system's classification performance on real-world examples, but serves to select the classifier parameter set for the following experiments.

B. Ranking Separation Quality and Rejection Threshold

Next, the ranking for correct classification results was evaluated using our testing data set with respect to the separation quality between the first-ranked (correct) and the second-ranked (incorrect) hypothesis. The metric used here is the confidence which is given by the SVM with each binary classification result as explained in Section IV-E. In the best case, the first-ranked hypothesis will appear with a confidence of 1, the second-best with a confidence of 0.

As shown in Figure 5, this works very well because, even in cases where the first-rank confidence drops to less than 0.1, second-rank confidences still reside at well below 0.02. We observed this effect in all our experiments, so we decided to set the rejection threshold in our framework to 0.02. For this setting, correct first-rank hypotheses are accepted reliably while avoiding to report incorrect hypotheses which occasionally occur if the respective classifier fails to positively recognize an object.

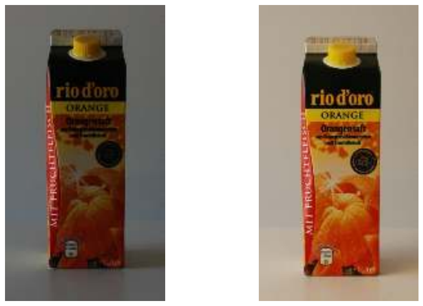


Fig. 6 – One example of illumination change represented in the training and testing data set. Both images were taken with identical aperture, shutter and light sensitivity settings.

C. Robustness Under Difficult Conditions

Another experiment was conducted under strongly varying conditions regarding the training and testing data set. These include changes of the viewpoint and illumination in a way that was not represented during training. Figure 6 shows an example of illumination variation, images taken with identical aperture, shutter and light sensitivity settings. As for the viewpoint changes, using the same automatic turntable as for model creation allowed us to vary viewpoints in a 45 degree pitch and 360 degree yaw range as well as distances from the capturing device between 80 and 120 cm.

We paid attention to distribute these variations randomly over the training and testing data set. The proposed method was run 30 times on a selected set of our objects in order to evaluate the quality of our training data as well as proving the robustness of our approach. A sophisticated test structure was needed to create a setting where each run was reproducible, but sufficiently distinct to ensure the validity of the experiment due to a real-world-like distribution of the testing samples. This is the reason why we limited the test repetitions to the number of 30 which was achievable with reasonable efforts.

Figure 7 shows an exemplary representation of correct and incorrect recognition results for different objects as well as those below the rejection threshold. In general, our approach performs well regarding incorrect hypotheses as there are very few of them. The ratio of incorrect recognitions appearing for the orange juice package (blue part of the middle bar) is caused by one of the other very similar juice packages, the incorrectly recognized Coke cans (blue part of the bottom bar) represent a similar-looking Coke can of smaller volume (see Figure 4). The latter failures were mostly caused by the can's point cloud being cropped through infrared light interference which results from the Kinect hardly being able to deal with reflective metallic surfaces.

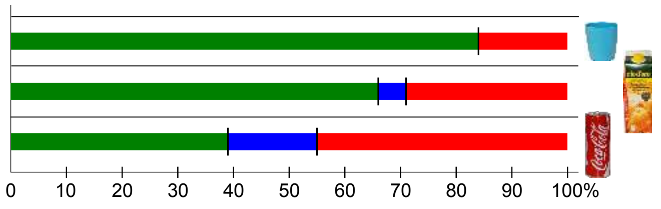


Fig. 7 – Classifier performance on selected objects under conditions strongly differing to those in training – correct hypotheses are shown in green (left part of the bars), incorrect hypotheses in blue (center), results below the rejection threshold in red (right).

Apart from these exemplary objects as shown in Figure 7, the majority of our gallery objects behave approximately like the orange juice package. They show very occasional incorrect hypotheses and results below the rejection threshold in a range of 10-35%, but all other candidates are reliably recognized.

These experiments allow us to draw a major conclusion: If the user encounters incorrect recognition hypotheses, this indicates that they should sample more training data under these particular environmental conditions. Even if independent test data is used to verify classification accuracy like the one in Table I, the classifier is unable to optimally generalize under extremely changing conditions.

However, the results below the rejection threshold in Figure 7 do not have dramatic effects on the overall object recognition accuracy as there is an easy way to deal with them: If encountering an object which the robot is unable to identify, it can, for instance, change its viewpoint by moving a little to either side. This way, the object may be identified even if the first recognition run was unsuccessful. Special value arises from the fact that we detect the object first by clustering the scene into possible objects instead of running one-size-fits-all recognition on the whole scene. This makes it possible to know that there is an object and, in case of an unsuccessful recognition iteration, gather new sensor data and run the recognition pipeline once again.

D. *Training and Classification Time Performance*

Regarding the time needed for training, we ended up with the 45 SVMs taking about 30 minutes per classifier to complete on on one core of a standard 3.1-GHz CPU, that equals 22.5 hours in total. This time does not play a major role, though, as training needs to be done only once prior to productive operation. Much more crucial is the time needed for running the classification which took about 5 ms per classifier. Integrated with data acquisition, segmentation, and clustering, we were able to optimize the whole recognition process to run in about 2 seconds on a setting which includes a table with one object on top of it.

VI. Conclusion and Future Work

Our approach shows the application of established Machine Learning techniques to multi-algorithm object recognition problems. This is of interest especially in a context where different sensors and recognition algorithms need to be integrated into a system without the hassle of adapting hypothesis metrics to a common basis. Instead, outputs of different object recognition approaches are combined automatically. In real-world robotic environments, the method proposed in this paper may lead to many interesting applications due to its robustness.

As a real-world example, the proposed method was implemented on our service robot *Kate*. Please have a look on our YouTube channel to see *Kate* operate in a living room environment using our object recognition technique: `http://www.youtube.com/RoboticsHSWgt`.

One drawback of the current approach is the necessity to extend the number of features with the addition of new objects to the training gallery. This also means that the recognition algorithms have to be re-run and features have to be generated again to fill the extended feature space. Afterwards, the classifiers have to be re-trained. Unfortunately, this effect is not avoidable by any of the supervised learning algorithms we examined in this work. One remedy at least to the dilemma of having to re-train the classifiers might be several promising semi-supervised learning approaches that have emerged in the past years and allow for online learning at runtime, thus sparing the user from waiting for re-training to be finished.

Additional work should also be put into finding features generated by the object recognition scores in order to replace the raw recognition scores from Section IV-C. This way, with a fixed selection of features independent from the object gallery, it is only classifier training which needs to be repeated on gallery extension. Investigating and utilizing both these suggested changes, the process of adding more objects to the gallery becomes much easier to handle.

Possible future endeavors also include multi-level object classification, where objects can be grouped into hierarchical structures containing similar objects. Machine learning methods may provide crucial assistance because they are capable to separate object classes automatically.

Another benefit can be derived from machine learning when it comes to determining the rejection threshold for hypotheses as in Section V-B. From the results of the classifiers desribed in this paper, another super-classifier can be learned which is able to reject a sub-classifier's hypothesis if its confidence is too low.

To get a more accurate representation of real-world situations the recognizable objects can be found in, a more professional training data capturing device should be taken into account like the one used for building the KIT object database [11], a 360 degree yaw / 90 degree pitch high accuracy laser scanner / stereo camera combination.

Additionally, the importance of illumination for objects to distinguish mainly by color rises the demand for a replacement of the currently used conventional color histograms. We are already working on the integration of Color Correlograms [20] as an advanced color-based measure which may come handy by taking spatial relationship between colored areas into account.

Finally, we are currently working on the replacement of MOPED by an approach making use of the ORB descriptor [21] whose training runs on RGB-D data from the Kinect instead of high-resolution images. As soon as these works are completed, we will be able to attach an object database like RoboEarth [10] or the KIT database which enables us to choose from an even greater variety of objects and provide for a lot more applications and real-world examples.

Acknowledgements

The authors would like to thank Janos Knobloch, Karl Glatz, Markus Mauer, Martin Bertsche, Richard Cubek, Stephan Schädle, and Vien Ngo for fruitful discussions as well as Alvaro Collet, the creator of MOPED, for his generous support.

This work was conducted within the Collaborative Center for Applied Research on Service Robotics (ZAFH Servicerobotik) and supported by research grants of the state of Baden-Württemberg and the European Union.

References

[1] W. Scheirer, A. Rocha, R. Micheals, and T. Boult, “Meta-Recognition: The Theory and Practice of Recognition Score Analysis,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1689–1695, August 2011.

[2] W. Scheirer, A. Rocha, J. Parris, and T. Boult, “Learning for Meta-Recognition,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1214–1224, Aug. 2012.

[3] H. Wu, M. Siegel, R. Stiefelhagen, and J. Yang, “Sensor Fusion Using Dempster-Shafer Theory,” in *Proc 19th IEEE Instrumentation and Measurement Technology Conf*, vol. 2, no. May, 2002, pp. 21–23.

[4] M. Aeberhard, S. Paul, N. Kaempchen, and T. Bertram, “Object Existence Probability Fusion using Dempster-Shafer Theory in a High-Level Sensor Data Fusion Architecture,” in *Intelligent Vehicles Symposium*, Baden-Baden, Germany, 2011.

[5] L. Di, T. Fromm, and Y. Chen, “A Data Fusion System for Attitude Estimation of Low-cost Miniature UAVs,” *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1–4, pp. 621–635, 2012.

[6] S. Gould, P. Baumstarck, M. Quigley, A. Y. Ng, and D. Koller, “Integrating Visual and Range Data for Robotic Object Detection,” in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, Marseille, France, 2008.

[7] M. Attamimi, A. Mizutani, T. Nakamura, T. Nagai, K. Funakoshi, and M. Nakano, “Real-Time 3D Visual Sensor for Robust Object Recognition,” in *IEEE/RSJ Internation Conference on Intelligent Roboots and Systems*, 2010, pp. 4560–4565.

[8] A. Janoch, S. Karayev, Y. Jia, J. Barron, M. Fritz, K. Saenko, and T. Darrell, “A Category-Level 3-D Object Dataset: Putting the Kinect to Work,” in *First IEEE Workshop on Consumer Depth Cameras for Computer Vision at the International Conference on Computer Vision (ICCV)*, 2011.

[9] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view RGB-D object dataset,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[10] M. Waibel, M. Beetz, J. Civera, R. D’Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, “Roboearth,” *Robotics Automation Magazine, IEEE*, vol. 18, no. 2, pp. 69–82, 2011.

[11] R. Becher, P. Steinhaus, and R. Dillmann, “Interactive object modelling for a humanoid service robot,” in *International Conference on Humanoid Robots 2003*, Karlsruhe, Germany, 2003.

[12] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[13] M. Bertsche, T. Fromm, and W. Ertel, “BOR3D: A Use-Case-Oriented Software Framework for 3-D Object Recognition,” in *IEEE Conference on Technologies for Practical Robot Applications*, Woburn, MA, USA, 2012.

[14] A. Collet, M. Martinez, and S. S. Srinivasa, “The MOPED framework: Object Recognition and Pose Estimation for Manipulation,” *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, Sep. 2011.

[15] W. Chen, Y. Shi, and G. Xuan, “Identifying Computer Graphics using HSV Color Model and Statistical Moments of Characteristic Functions,” in *IEEE International Conference on Multimedia and Expo*, Beijing, China, 2007.

[16] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[17] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[18] M. Riedmiller, “Rprop - description and implementation details,” Technical Report, 1994.

[19] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[20] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, “Image Indexing Using Color Correlograms,” in *IEEE Computer Science Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: an efficient alternative to SIFT or SURF,” in *IEEE International Conference on Computer Vision*, Barcelona, Spain, 2011.