IRON-BAG: Fast Classification of Humans and Objects in 3D NDT-Maps using Structural Signatures

Thomas Schmiedel and Horst-Michael Gross

***Abstract*— We propose a real-time algorithm for the generic classification of humans and objects in 3D scenes. The algorithm does not depend on color information and works with depth data alone, making it very flexible for a wide area of applications. Further, we will show that it is very resistant to occlusion and will give correct classification results even in cases, where only a fraction of a full human or object can be captured by the depth sensor. Opposed to current approaches based on deep networks, training the IRON-BAG classifier (a bag-of-words model for IRON-features) can be done within minutes, making it easier to add new object classes, to fine-tune parameters and to adapt it to new operational scenarios. The system is easy to use, as it does not impose any constraints on the objects to detect, e.g. there's no limitation regarding shape, height, orientation, or position of humans and objects - knowledge of the sensor-pose or ground plane is not required. Instead of using depth images or point clouds as inputs for our classification pipeline, we solely operate on the Normal-Distribution-Transform-map (NDT-map) data structure. NDT-maps provide a highly memory-efficient representation of depth data, and we show that the information contained within them is sufficient to accurately classify humans and objects from real-world 3D scenes with a speed of around 180 classifications per second on a single CPU core.**

I. Introduction

The classification of humans is an important task for a robot to accomplish: a service robot cannot communicate with its owner, or deliver any services, if it cannot find out where he or she is. On the other hand, when people cooperate with robots or autonomous machines, e.g. in production environments, it is crucial for human safety to have fast algorithms for person detection available. The detection and classification of 3D objects is equally important, as it enables a mobile robot - or vehicles of any kind - to identify and avoid obstacles and to make human-robot-interaction even more comfortable by including semantic knowledge of the environment and the objects the user is interacting with. For instance, it's much easier to ask a service robot to wait next to the couch and have it detect that position automatically, than to store each and every set of coordinates manually beforehand.

Since there are lots of different scenarios for person and object detection, our intention is to propose a flexible classification approach that does not put too many constraints on the sensor setup and the environment. A technique that operates on 3D depth data alone has those advantages: it works independently of illumination, there are many different sensors available for indoor/outdoor use, and it can potentially be faster than a RGB-D setup, simply because no color information must be processed (for a classification example see Fig. 1). The classification of persons and objects in 3D scenes generally requires a prior segmentation step to identify regions of interest. The novelty in this work, however, primarily concerns the classification step after a segmentation has been conducted. Therefore, persons and objects will be treated as isolated regions during the course of this paper. We will show in the evaluation section, however, that IRON-BAG will readily work with arbitrary segmentation methods in order to classify complete 3D scenes in real-time. The paper is organized as follows. At first, we briefly introduce the state-of-the-art in terms of 3D person detection and object detection techniques and explain why IRON-BAG is different (and why we named it that way). Afterwards, in section V-F, the full classification pipeline is elaborated. Section V-G will then demonstrate a simple and fast statistical measure for rejecting 3D objects that do not belong to any object class trained before. The final part of this paper will contain experimental results on a publicly available 3D object dataset using more than $25,000$ depth images as well as a final evaluation of classification and training speed.

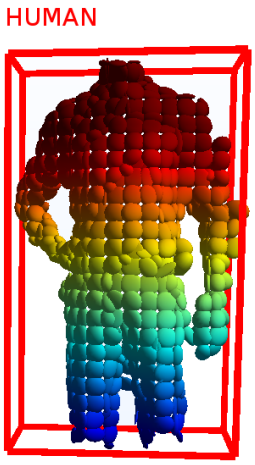

Fig. 1: Correctly classified NDT-map of a standing person.

II. State-of-the-Art

Most use cases for 3D classification procedures can be divided into four major categories: those who focus on humans, those who focus on objects, those who utilize fused RGB-D information, and those who classify with depth data

All Authors are with Neuroinformatics and Cognitive Robotics Lab, Technische Universität Ilmenau, 98694 Ilmenau, Germany.

This work has received funding from the German Federal Ministry of Education and Research (BMBF) to the project SYMPARTNER (grant agreement no. 16SV7218)

alone. Of course, the final algorithm selection depends on the specific application and the available sensor systems. When persons should be detected in indoor environments with somewhat constant illumination, many publications focus on RGB-D data acquired from a PrimeSense device [1][2]. Likewise, when small-scale objects must be detected, fused RGB-D is often the preferred type of input data, as many small-scale objects can only be distinguished by color (e.g. different brands of breakfast cereal packages) [3][4].
However, there are also some cases where additional color information is not available (or not reliable due to strong changes in illumination), and the detection algorithm has to work on depth data alone [5]. Spinello et al. [6] have published an approach to pedestrian detection in range scans with good results. They took range scans from a Velodyne LIDAR device, divided them into segments which were then associated with 17 explicitly defined features, such as width, circularity, and PCA ratio. The final classification was then carried out using AdaBoost.
IRON-BAG is based on the "Fast Interest Point Descriptor for Robust NDT-Map Matching (IRON)"[7] and is inspired by the popular bag-of-words model in computer vision. IRON-BAG expresses each object by a compact structural signature that implicitly contains information about object shape, dimensions, and surface curvature. Those signatures are the only data needed for classification; no information about position, orientation, dimensions, sensor pose, and reference frames must be given explicitly.
Those properties make IRON-BAG very easy to use: training samples do not need any ground truth or sensor pose attached to them, saving plenty of time for sensor setups. Opposed to deep learning based approaches [4], training is a matter of minutes even for large datasets comprising several thousands of point clouds. And finally, the actual classification will only take a few milliseconds to complete, as our approach utilizes the efficient NDT-map data structure [8] for representing all depth information.

III. Motivation

Our main incentive, driven by low-cost oriented service robots, is to deliver real-time performance on consumer hardware without GPU utilization while at the same time achieving very good classification and detection performance in depth data alone. Our understanding is that training must also be fast to enable quick tuning of training parameters for new scenarios and additional object categories.
The algorithm we describe should be widely usable, as it does not depend on RGB camera data and, therefore, is insensitive to illumination. Moreover, our aim is to classify persons and objects in real-time even when strong occlusions are present and only object fragments are visible. This all should be achieved without external knowledge of a ground plane or any other predefined circumstances, such as people normally standing up straight on the ground, or that they have a certain height and so on.

IV. Our Approach

To tackle the demand for a high processing speed, we build upon recent work about the IRON descriptor for NDT-maps [7] which has shown very good real-time performance.
Normal-Distribution-Transform-maps (NDT-maps) provide an efficient representation of 3D depth data by dividing the scene into voxels and storing a multivariate normal distribution for the points within each cell [8]. The raw points are discarded afterwards, enabling fast further processing, since the amount of data is severely reduced. As the covariance part of each voxel's normal distribution $\mathcal{N}(\vec{\mu}, \Sigma)$ is positive semi-definite, NDT-maps can be visualized as collections of ellipsoids (see Fig. 2).

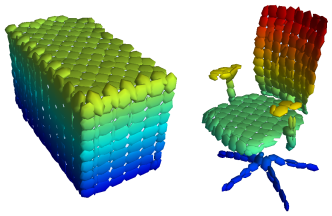

Fig. 2: NDT-map representation of 3D data obtained from a box and a chair (sensor: PrimeSense)

The eigenvector corresponding to the smallest eigenvalue of one NDT-cell $\mathcal{N}_i$ can with certain restrictions (see [7]) be assumed to be equal to the surface normal in the point $\vec{\mu}_i$. This is the basis for the construction of IRON-descriptors for each NDT-cell of an NDT-map. The descriptor $D_i = \{A, S\}$ for NDT-cell $\mathcal{N}_i$ consists of an angular part A and a shape part S which are both 2D histograms of equal size, storing surface curvature and shape of the spherical neighborhood around cell $\mathcal{N}_i$. IRON is locally defined within the 3D map, it is invariant to rotation and invariant to the direction of NDT-cell normal vectors, making it independent of a global reference, such as ground plane and sensor pose.
For the actual classification task, we have adapted the famous bag-of-words model to IRON-descriptors. We show that it is sufficient for good classification results to represent an NDT-object via its similarity to a few NDT-code-words. No external object shape information is needed, as the IRON-descriptors contain that implicitly.

V. Classification Pipeline

In this section, we discuss the steps necessary to transform a 3D point cloud into the final feature vector that is used for classification. The actual classifier as well as a fast outlier measure will be outlined afterwards. We will also briefly discuss a procedure for quickly obtaining plenty of training point clouds from an object. Please note that all the pre-processing steps are fully unsupervised, they do not require label information, but merely exist to find a descriptive and lower-dimensional representation (= a "structural signature") of a collection of NDT-cells. This signature will finally be classified in a supervised fashion.

A. Data Acquisition

On our robots, we use a calibrated Asus Xtion PrimeSense device for capturing depth images - the RGB information is discarded. Then, by means of the previously obtained projection parameters we can reconstruct the 3D point cloud from it. For the recording of training data, a straightforward approach is to put the object on a flat surface and to move around it with a depth camera by hand, or with it mounted on a robot. This will easily record more than $1,000$ depth images a minute. Depending on object complexity and size, $2,000 - 5,000$ images per object category are preferable. Since IRON-BAG does not require knowledge of the sensor pose, this procedure is already sufficient. Afterwards, the ground plane can be automatically identified by using RANSAC algorithm and be subtracted from the scene. This is done to ensure that training samples do contain pure object data only.

B. Computing the IRON-Descriptors

The actual IRON-BAG processing starts with a given training set of point clouds captured from persons and/or different object classes. As a first step, the point clouds are transformed into NDT-maps (see Fig. 2). The size of NDT-cells was chosen to be $5cm \times 5cm \times 5cm$, as this is a good tradeoff between a high algorithmic speed (large NDT-cell size) and a fine detail for classification accuracy (small NDT-cell size). The next step includes the computation of IRON-descriptors for all NDT-cells in all given training NDT-maps. For this to work, the IRON keypoint detector must be disabled by setting the IRON entropy threshold to zero [7]. Otherwise, only descriptors from salient surface regions would be extracted. Since it is required to detect objects with flat areas on them (such as desks, tables, and boxes), however, all NDT-cells must be included for good classification performance. The last step of the data acquisition routine is feature standardization. Therefore, we compute the mean of each descriptor's histogram values and subtract it, afterwards the results are divided by their standard deviation.

C. Whitening

Coates et al. [9] have shown a significant increase in classification accuracy when whitening is enabled and the K-means clustering algorithm is used for code book generation for data from RGB image patches. An explanation for this phenomenon is the fact that K-means is blind to variable correlation. It will therefore tend to under-represent dimensions with low variance within the data and distribute its centroids along those dimensions with high variance (see Fig. 3). As a result, we apply simple PCA-whitening to the training set, the steps are the following:

- All IRON-descriptors from all object classes will form a feature space ($\mathbb{R}^{2 \times a \times d}$, where a is the number of histogram bins to store angular information within the IRON-descriptor and d is the number of bins to represent different distance levels within the IRON-descriptor).
- Ensure the data has mean zero and compute the covariance Σ_s over all data samples (IRON-descriptors).
- Apply eigenvalue decomposition $\Sigma_s = U_s \Lambda_s U_s^\top$.
- Receive whitened feature vectors: $\vec{x}_i' = \Lambda_s^{-1/2} U_s^\top \vec{x}_i$.

The full feature space is now white - the eigenvalues of Σ_i' are one and all data dimensions are represented equally.

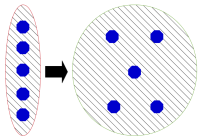

Fig. 3: An outline how whitening can benefit a subsequent K-means clustering step by enabling it to spread centroids (blue) more evenly along all feature dimensions.

D. Code-Book Generation

Now that the descriptors from all NDT-cells of all training NDT-maps have been transformed into a white feature space, we need to find cluster centers (we also call them NDT-code words) within that space. The intuition is that those centers represent IRON-descriptors that are significant and common in many NDT-maps.
For the computation of NDT-code words, the completely whitened training set is required. We use K-means for clustering, but employ an improved seeding strategy somewhat similar to K-means++[10]. We will randomly draw 100 times the required amount of initial seeds and then iteratively remove the seed which is closest to any other seed until K seeds are left. This will favor seeds that are more evenly distributed over the feature space. After that, K-means is repeated until convergence. This is the most computationally expensive part of the presented IRON-BAG technique, however, this step has to be done just once during training. K-means generally needs to know the amount of centroids K beforehand. Depending on the number of object classes and problem difficulty (it is difficult to distinguish between very similar object classes, e.g. a person and a coat stand), we have determined a reasonable range of K between 20 and 100 centroids.
To put the clustering more formally, the set of whitened feature vectors X' is partitioned into K disjoint subsets $X_1', X_2', ..., X_K'$ in such a way that they have minimal within-cluster variance: $\text{minimize} \sum_{k=1}^{K} \sum_{\vec{x}_j' \in X_k'} \left|\left| \vec{x}_j' - \vec{\mu}_k \right|\right|^2$ where $\vec{\mu}_k$ denotes the mean of subset X_k'. After convergence, the code-book B' consists of the computed cluster centers $B' = \{\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_K\}$.

E. Object Signatures

NDT-code-words can be thought of as IRON-descriptors inside the whitened feature space that mark distinct parts of surface curvature and shape within the training set. Given an NDT-map, it is now required to express all IRON-descriptors from that map in terms of their similarity to the previously computed code-words $\{\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_K\}$. There are several approaches to this [9], however, the common idea is to build

a histogram of K code-words and increment those histogram bins for each new incorporated feature vector according to a certain similarity measure (comparable to sparse coding known from many neural network approaches). The standard "one-hot"-assignment, where only the histogram bin of the closest code-word is incremented, has shown to produce sub-optimal results [11]. We therefore employ a smooth assignment procedure using the simple RBF kernel: $f(\vec{x}'_i, \vec{\mu}_k) = \exp(-\gamma_{sig} * ||\vec{x}'_i - \vec{\mu}_k||^2)$. The steps for processing a single NDT-map are as follows:

- Given an NDT-map as well as the code-book B' and the whitening parameters obtained during training $(\Lambda_s^{-1/2}, U_s^\top, \vec{\mu}_{space})$.
- Compute the standardized (see V-A) IRON-descriptors for each NDT-cell.
- Transform the obtained feature vectors into a set of whitened feature vectors X' using the given transform.
- Create an empty histogram with each bin k corresponding to the code-word $\vec{\mu}_k$.
- For each feature vector $\vec{x}' \in X'$ and each code-word $\vec{\mu}_k$ compute the value of $f(\vec{x}', \vec{\mu}_k)$ and store it in the k-th component of a temporary vector $\vec{t}$.
- Normalize $\vec{t}$ using L_1-norm and add it (component-wise) to the histogram.
- After that, for normalization purposes, divide each final histogram bin value by the number of incorporated vectors $\vec{t}$.

We are left with a histogram (= structural signature) $\vec{s}_K$ of the complete NDT-map that consists of just K values. Other than the number of clusters K, the only other variable to choose for signature construction is γ_{sig}. A high γ_{sig} gives more weight to code-words that fit very well, which might lead to a stronger sensitivity to noise. When γ_{sig} becomes too small, all code-words will be weighted equally, leading to unexpressive signatures [11].

F. Classification

After the unsupervised generation of signatures for each object, a supervised classification step must follow. Classification is solely done with signatures, no other information is required. To get an impression how the signatures of two different object categories are distributed within the signature space ($\mathbb{R}^K$), Fig. 4 shows a plot for $K = 4$ code-words.
It is already visible for this small amount of code-words that the object classes form easily separable clusters within the signature space. Also, from 2D t-distributed stochastic neighbor embedding (t-SNE) images [12] of several different signature spaces, we can conclude that histograms from the same object category are oftentimes grouped together, forming dense clouds with some slight overlap to neighboring classes. To keep classification simple, as well as the number of classification parameters low, we decided to implement a weighted nearest neighbor classifier for this task. It operates as follows:

- Given the signature sets $S_1, S_2, ..., S_N$ and class labels $l_1, l_2, ..., l_N$ for N different object classes.
- Training is done by simply building a k-d-tree from all signatures in all signature sets (we are using the fast library for approximate nearest neighbor search - nanoflann - with L_2 metric for this purpose [13]).
- To classify an unknown signature $\vec{s}_{new}$, we query the k-d-tree for its $(K_{neigh} + 1)$ nearest neighbors (K_{neigh} is user-defined and should not be confused with the amount of code-words).
- The L_2 distance between query point ($\vec{s}_{new}$) and each of the K_{neigh} nearest neighbors is divided by the distance to neighbor $(K_{neigh} + 1)$, normalizing them to $0 \leq d_k \leq 1$.
- For each neighbor up to K_{neigh} a weight w_k is computed according to $w_k = \exp(-\gamma_{neigh} * d_k)$ and added to the entire weight of the object class this specific neighbor stands for (as denoted by his label).
- After all K_{neigh} neighbors have been processed, the class with highest weight is selected.

The advantage of this method is simply that close neighbors to the query point are weighted stronger than neighbors further away. The affinity to the query point can be tuned via parameter γ_{neigh}, we normally set it to 2.0.

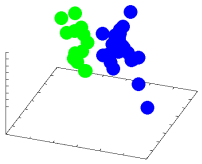

Fig. 4: Signature space ($\mathbb{R}^4$, first 3 dimensions visualized) of 13 full motorbikes (green) and 13 full humans (blue) (see Fig. 5 for raw object point clouds). Both object classes form easily separable clusters.

G. Outlier Detection

Up to now, only scans from those objects can be handled that have actually been trained before. In such a case the nearest neighbor classifier will just select the object class that has highest weight. However, when unknown objects are present, we need a way to reject them. To keep IRON-BAG as general as possible, we did not intend the use of any negative training data, as this would mean to collect depth images from a huge amount of shapes and objects just to indicate that they should *not* be classified. Eventually, this would also sacrifice training speed and inflate general memory consumption. A different approach, however, is to find a statistical model for the already trained object classes and to decide whether a given sample is an outlier or not. For this to work, we must make some general assumptions regarding the distribution of signatures within the signature space:

- We assume the signatures for one object class to be normally distributed.
- If we have K code-words, there remain $K - 1$ degrees of freedom, since all signatures have been normalized using L_1-norm (or to put it differently: if we vary $K-1$ components of a signature, the last one is always fixed).

Making those assumptions has some pleasant side effects. We can compute a multivariate normal distribution $\mathcal{N}(\vec{\mu_c}, \Sigma_c)$ for the signatures of each object class, and we can check for a new signature $\vec{s}$ whether it is significantly away from our class distributions. Here's the procedure:

- Precompute a $\vec{\mu_c}$ and Σ_c^{-1} for each object class.
- Receive a result from the nearest neighbor classification.
- For the signature $\vec{s}$, compute the squared Mahalanobis distance to the proposed object class: $d_s^2 = (\vec{s} - \vec{\mu_c})^T \Sigma_c^{-1} (\vec{s} - \vec{\mu_c})$.
- The CDF (cumulative distribution function) of the chi-squared distribution with $K - 1$ degrees of freedom $\chi^2(K-1)$ gives the probability p of observing a signature closer or equal to the object class than the given signature $\vec{s}$.
- Using the inverse CDF (quantile function), we can therefore compute a squared distance d_{thres}^2 which is further away from the object class than $(p \times 100)\%$ of valid signatures that originate from that object class.
- Now we simply check whether $d_s^2 > d_{thres}^2$, and discard the new scan if the expression is true.

This technique has the advantage of being very fast. Parameters of all $\mathcal{N}(\vec{\mu_c}, \Sigma_c)$ as well as the inverse covariance matrices Σ^{-1} can be computed during training. The threshold d_{thres}^2 will be also calculated beforehand according to parameter p. The outlier rejection therefore boils down to a Mahalanobis distance computation and one variable comparison.

VI. Evaluation

A. Baseline for Parameter Selection

This section is providing the IRON-BAG parameter setup that we used for the subsequent classification tests. This is generally a good baseline, and the algorithm should work well with it.

- lots of training data: $2,000 - 5,000$ point clouds from one object category should be sufficient and are easily obtained (see section V-A)
- NDT-map construction: cell-size: 5 cm, entropy filtering: disabled
- IRON-descriptor computation: the number of histogram bins used for angle representation is 8, the number of bins used for encoding distances within the IRON-descriptor is 3, the radius of the spherical region around the descriptor base is 0.5 m
- code-book construction: 50 code-words/K-means centroids, K-means iterations: until convergence
- signature generation: $\gamma_{sig} = 0.01$
- nearest neighbor classification: $\gamma_{neigh} = 2.0$, number of neighbors to query (K_{neigh}): 17
- outlier rejection: $p = 0.999$

B. Classification Accuracy

To test the classification accuracy, a publicly available 3D object dataset was used [14]. We selected five general object categories: humans, chairs, tables, motorbikes and containers (as the given dataset did not contain humans, we supplemented our test set with recordings from five different persons, see Fig. 5 for an example). Each of those categories comprised of about $5,000$ 3D depth images obtained from five different objects per class, giving a full data set size of about $25,000$ point clouds. All recordings were made using a PrimeSense device [14]. Pre-processing of the depth scans obtained from the dataset:

- All depth images were transformed into point clouds.
- Using RANSAC, we removed the ground plane from all point clouds to ensure they contain just object points. Please note that this is not a step within IRON-BAG (which has no knowledge of a ground plane), but a preparation to ensure equal conditions in the classification test.

We did not combine any scans to get a more complete model. This strongly raises classification difficulty. As the PrimeSense device has a small viewing range and generally cannot look behind an object, point clouds obtained from a single depth image mostly show object fragments only. With this in mind, the actual task becomes to classify object categories from object fragments (or severely occluded objects). Fig. 5 gives an impression of the typical appearance of point clouds from single PrimeSense scans.

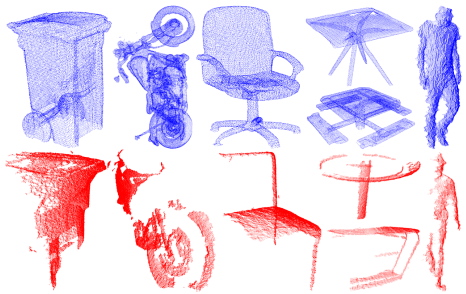

Fig. 5: Example objects from all selected categories: containers, motorbikes, chairs, tables and humans. The upper row (blue) contains the full models as reference. The bottom row (red), however, shows the typical appearance of single depth scans from those object categories. This can sometimes be as little as a single wheel from the motorbike, but classification must cope with that.

Testing procedure:

- Training was done for all categories using a random sample containing 70% of all available scans.
- The other 30% of all samples were used to verify the classification accuracy.
- The whole process was repeated 10 times, training and validation point clouds were drawn randomly each round.

Table I gives the confusion matrix after completing about $75,000$ classifications. Those results basically show that the IRON-BAG classification in over 93% of all cases was able to correctly determine the object category from a single occluded point cloud. There was some minor confusion between fragments of humans and fragments of motorbikes

TABLE I: Confusion matrix of classification results in percent after about $75,000$ classification procedures. The left row contains the object category under test.

	Human	Bike	Chair	Table	Container
Human	**93.83**	5.51	0.01	0.32	0.32
Bike	0.28	**97.14**	1.51	0.71	0.36
Chair	0.01	0.84	**98.03**	0.53	0.59
Table	0.01	0.17	0.28	**99.42**	0.11
Container	0.02	0.21	1.30	0.01	**98.45**

which can be attributed to their similar size and surface curvature which is generally curved in both cases. As a severe difficulty (and/or advantage), the IRON-BAG algorithm does not depend on any additional information, such as object dimensions, sensor pose, or ground plane. E.g., it cannot distinguish a standing human from a motor bike just by looking at their orientation with respect to the ground plane. It must carefully consider surface curvature and local shape in order to make a decision.

C. Rejection of Outliers

In this section, we are evaluating how the IRON-BAG classifier behaves if scans are presented that do not belong to any known class and must be rejected. Table II shows the same classification procedure as before, however, this time for each scan an outlier check is executed. Also, about $12,000$ point cloud fragments from three classes (toy, rock and plant) were presented, which the classifier has never seen before.

TABLE II: Confusion matrix of classification results in percent after about $87,000$ classifications. In addition to previously trained object categories, also unknown object fragments were presented.

	Human	Bike	Chair	Table	Container
Human	**87.01**	7.14	0.01	0.02	0.02
Bike	0.09	**87.57**	1.25	0.36	0.18
Chair	0.01	0.09	**87.77**	0.01	0.18
Table	0.01	0.00	0.23	**87.10**	0.17
Container	0.01	0.05	1.50	0.09	**90.57**
Toy	0.00	14.92	0.64	0.00	0.00
Rock	0.00	2.21	0.34	4.88	1.93
Plant	0.00	1.11	0.04	0.00	0.00

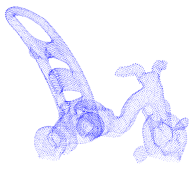

Fig. 6: Full model of the test object "toy".

As can be seen in this second test run, when the outlier rejection step is done after classification, results for trained object categories are slightly impaired (as some correct results were regarded as outliers). However, they are still constantly in the range between $87-91\%$ accuracy. On the other hand, unknown objects could be successfully rejected. The only exception being the toy bicycle (see Fig. 6) which was in 15% of all cases classified as "motorbike", though it can be reasoned that this is not a fully misguided decision, as both objects share many similar features and basically portray the same general device.

D. Speed Assessment

There are two major factors that can be measured here: classification speed and training speed. During evaluation section VI-C, about $87,000$ classifications were done - including the previously unknown test object classes. We measured the average time for the following procedures, which have to be done for every single point cloud classification:

- construction of the NDT-map (see section IV)
- computation of IRON-descriptors for the obtained map
- standardization of those descriptors (see section V-A)
- transformation of the descriptors into the whitened feature space (see section V-C)
- computation of an object signature from the whitened IRON-features (see section V-E)
- classification of that signature (see section V-F)
- fast outlier check using a threshold obtained from the chi-squared distribution (see section V-G)

On average this whole process took only 5.54 milliseconds on a single core of an Intel Core i7-4770 CPU. This translates to about 180 point cloud classification procedures per second on one CPU core and can be considered real-time.
It should be noted that this time does not include the time for loading a point cloud from hard disk, as this would be very different depending on the hardware setup. However, once it is in memory, all the steps depicted above take about 5.54 ms to complete. This is an average value for point clouds from the tested dataset (containing humans, motorbikes, chairs, tables and containers) and will be slightly different for other object classes. The other important measure is training speed. The following steps were taken into account:

- construction of NDT-maps from all training point clouds (see section IV)
- computation of IRON-descriptors for all obtained maps
- standardization of those descriptors (see section V-A)
- computing the whitening transform from all features (see section V-C)
- K-means clustering (until convergence) for code-book generation (see section V-D)
- generation of signatures for all training maps (see section V-E)
- construction of a k-d-tree for the nearest neighbor classifier using all training signatures

The full training process (without the time for loading point clouds from disk) for about $17,500$ training point clouds (a single round in evaluation section VI-B) took 17 minutes on average. K-means clustering was executed until convergence which generally amounted to more than 300 iterations. Limiting this number to $20-50$ will in most cases not impair classification accuracy, but will further improve training speed.

E. Classification of Segmented Scenes

While this paper has covered the complete classification pipeline of isolated objects, there was not much focus on scene segmentation. To show that IRON-BAG can be readily used in combination with an arbitrary prior segmentation step, we have implemented the density-based DBSCAN clustering algorithm [15] as an example. It will divide a full NDT-map into disjoint clusters that are classified afterwards. This process runs in real-time as well (DBSCAN average runtime complexity: $O(n \log n)$). See Fig. 7 for a segmented and correctly classified domestic scene consisting of three different objects.

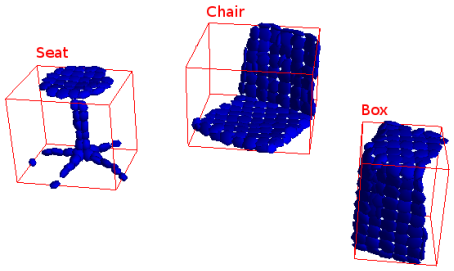

Fig. 7: Example of a segmented and classified 3D scene with some occluded parts.

VII. Conclusion and Outlook

A. Discussion

In this paper we have presented a new algorithm, IRON-BAG, for the robust classification of persons and objects from 3D scenes. The approach is fast (see section VI-D), and therefore well suited for real-time applications.
Due to its generic design, it can be used to classify arbitrary objects and works even reliably when just fragments of the full object can be captured by the depth camera. However, as the algorithm does not make use of color information, objects must have at least some slight distinction in shape and/or surface curvature. In contrast to deep-learning based classification methods, IRON-BAG can be trained on a traditional CPU and training itself will only take a few minutes to complete even for large datasets (see section VI-D). The structural signatures used for classification are constructed from IRON features (see section V-E), which in turn are invariant to rotation, ground plane position, and sensor pose. IRON-BAG inherits those properties, making it equally easy to use. To name an example: a training set might contain just chairs standing upright, but the classifier will also recognize a chair if it is lying on the side, as long as a few relevant features are not occluded.
For proper classification performance, it is nevertheless advised to have lots of training data. This is easily achieved by putting the object on a flat surface and moving around it with the depth sensor until $2,000 - 5,000$ images have been captured (see section V-A) - meanwhile, there is no need to keep track of or estimate the sensor pose, since it is not required for IRON-BAG classification or outlier rejection.

All in all, due to its high speed of about 180 full classifications per second on a single CPU core (about 720 per second on one quad-core CPU) and its **very high accuracy of more than 93% in 75,000 NDT-map classification procedures**, the presented technique enables even low-cost platforms, equipped with little computing power and a simple depth camera, to robustly classify persons and objects in the environment. On the other hand, when strong computing capabilities are available, IRON-BAG will not have a noticeable impact on CPU utilization, leading to lower power consumption, and potentially extending the time a mobile platform can operate without having to recharge its battery.
As the algorithm is able to classify persons and objects at the same time, it has many applications in human-robot-interaction scenarios. Detecting the user is naturally an important step for a mobile robot. However, it is also possible to recommend a seat for the user, to wait at object "couch", or to handle obstacles in a smart way.

B. Outlook

NDT-maps provide interesting properties for the fast segmentation of depth-scenes. As a reference, we have just implemented a simple density-based clustering method (Fig. 7). It shows that IRON-BAG can be easily combined with an arbitrary segmentation procedure which enables it to classify complete scenes in real-time. We will explore further in that direction.

References

[1] L. Spinello and K. O. Arras, "People detection in RGB-D data," in *IROS*, 2011.
[2] M. Munaro and E. Menegatti, "Fast rgb-d people tracking for service robots," *Auton. Robots*, 2014.
[3] K. Pauwels, V. Ivan, E. Ros, and S. Vijayakumar, "Real-time object pose recognition and tracking with an imprecisely calibrated moving RGB-D camera," in *IROS*, 2014.
[4] S. Gupta, R. B. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," *ECCV*, 2014.
[5] D. Mitzel and B. Leibe, "Close-range human detection and tracking for head-mounted cameras," in *BMVC*, 2012.
[6] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart, "A layered approach to people detection in 3D range data." in *AAAI*, 2010.
[7] Th. Schmiedel, E. Einhorn, and H.-M. Gross, "Iron: A fast interest point descriptor for robust ndt-map matching and its application to robot localization," in *IROS*, 2015.
[8] M. Magnusson, T. Duckett, and A. J. Lilienthal, "Scan registration for autonomous mining vehicles using 3D-NDT," *JFR*, 2007.
[9] A. Coates, H. Lee, and A. Ng, "An analysis of single-layer networks in unsupervised feature learning." in *AISTATS*, 2011.
[10] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *SODA*, 2007.
[11] J. C. Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *ECCV*, 2008.
[12] L. van der Maaten and G. E. Hinton, "Visualizing high-dimensional data using t-SNE," *JMLR*, 2008.
[13] J. L. Blanco-Claraco, "Nanoflann library for fast approximate nearest neighbor search," https://github.com/jlblancoc/nanoflann, accessed: 2016-09-29.
[14] S. Choi, Q. Zhou, S. Miller, and V. Koltun, "A large dataset of object scans," *ECCV*, 2016.
[15] M. Ester, H. peter Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *AAAI*, 1996.