IMPLICIT SHAPE MODELS FOR OBJECT DETECTION IN 3D POINT CLOUDS

Alexander Velizhev[1,2], **Roman Shapovalov**[1], **Konrad Schindler**[3]

[1] Graphics & Media Lab, Lomonosov Moscow State University, Russia
[2] Photogrammetry Lab, Moscow State University of Geodesy and Cartography, Russia
{avelizhev,shapovalov}@graphics.cs.msu.ru
[3] Photogrammetry and Remote Sensing Group, ETH Zürich, Switzerland
schindler@geod.baug.ethz.ch

Commission III/3

KEY WORDS: Recognition, LiDAR, Laser Scanning, Point Cloud

ABSTRACT:

We present a method for automatic object localization and recognition in 3D point clouds representing outdoor urban scenes. The method is based on the implicit shape models (ISM) framework, which recognizes objects by voting for their center locations. It requires only few training examples per class, which is an important property for practical use. We also introduce and evaluate an improved version of the spin image descriptor, more robust to point density variation and uncertainty in normal direction estimation. Our experiments reveal a significant impact of these modifications on the recognition performance. We compare our results against the state-of-the-art method and get significant improvement in both precision and recall on the *Ohio* dataset, consisting of combined aerial and terrestrial LiDAR scans of 150,000 m^2 of urban area in total.

1 INTRODUCTION

Mobile mapping systems have become a popular means of rapidly acquiring large-scale 3D point clouds. These systems typically consist of scanning LiDAR and/or multiple video cameras, in combination with high-accuracy GPS/IMU sensors for navigation, mounted on a moving vehicle such as a van, train or helicopter. Applications include highway and railroad mapping and monitoring, cultural heritage documentation, 3D urban modeling, and many others. The primary output of a mobile mapping system consists of a huge unstructured cloud of 3D points, which then needs to be analyzed further to extract the desired task-specific information. The main step in most applications is local semantic analysis to detect semantically meaningful objects.

Object detection is difficult to perform automatically because of the large variability of real-world objects. Manual detection is time-consuming due to the huge amounts of data, and also error-prone because human visual system is not accustomed to interpreting unorganized point sets. Thus, automatic object extraction is both practically useful and technically challenging.

Detected objects might be either objects of interest for the task at hand, or unwanted objects to be masked out, such as cars in an urban mapping project. In the present work we develop a general framework for detecting objects of different classes based on a few given training examples. We concentrate on objects which are "small" and relatively compact, such as cars, traffic signs and lamp posts, as opposed to large structures, such as buildings and bridges.

The problem of semantic object detection in mobile mapping data is an active research topic. One of the most successful ways to approach it is the following pipeline (Golovinskiy et al., 2009): (i) filter out points of the ground; (ii) extract connected clusters of points (*connected components*), which are considered object hypotheses; (iii) compute a set of features for each connected component (e.g. height, volume, spin image etc.); (iv) apply a supervised classification algorithm to the feature vector to determine the object class. The main weakness of this pipeline is that

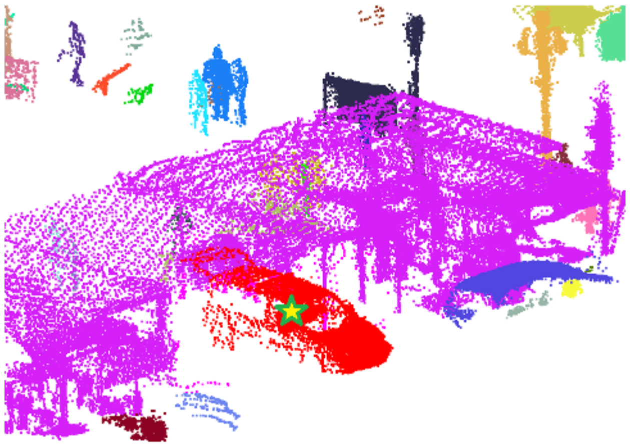


Figure 1: Example of a car within a complex urban scene, correctly detected by our method (colored red)

each connected component is described by a simple feature vector, which encodes only global properties. Such features are not discriminative enough for real-world object classes, which exhibit considerable intra-class variability. Moreover, occlusions and noise may cause significant problems. In particular, if a connected component contains more than just a single object, the spurious points are treated as noise or outliers. They corrupt global features unless they are correctly detected and removed, either explicitly or implicitly.

A group of *part-based* methods widely used in computer vision have also been adapted to recognizing 3D meshes (Toldo et al., 2009; Knopp et al., 2010). These methods are typically tested on scans of individual objects obtained under laboratory conditions. Extraction of meshes is possible when the point cloud is dense and clean enough, but cannot be accomplished reliably for mobile mapping data, hence those methods are not directly applicable to our problem. Nevertheless, their key idea is very powerful: they represent the surface with a part-based model, i.e. a set of local interest point descriptors together with their spatial configuration. The part-based representation is robust against partial occlusions and can handle moderate deformations of object geometry.

We make use of a part-based model for recognizing 3D objects in mobile mapping LiDAR data obtained in urban environments. Our approach can be described as an extension of the method of Golovinskiy et al. (2009): we replace supervised classification (via SVM) with a 3D implicit shape model (ISM) (Knopp et al., 2010), i.e. a part-based representation based on robust voting for the object center. In this paper we describe the adaptations which allow us to apply ISM to mobile mapping data. Our method shows significantly improved object detection performance in comparison to the state-of-the-art (Golovinskiy et al., 2009). Part-based models can better cope with inaccurate extraction of connected components. The method requires only few training examples per class, which is an important property for practical use. A typical detection result is shown in Figure 1.

2 RELEVANT WORK

This section is organized as follows: first we review methods for object detection in 3D point clouds, then we focus on 3D keypoint detectors and shape descriptors, since they are important in context of our work. We do not consider here semantic point labeling methods because of the following reasons: (i) these methods are usually limited to a small number of classes (ii) in the case of mobile mapping systems we have very dense and detailed data, so even discounting the effects of noise and occlusions it is usually not possible to assign every single point to one out of a small number of classes. We thus consider object detection, rather than point-wise labeling, to be more suitable for urban mapping tasks.

2.1 Object Detection

Driven by the development of LiDAR hardware, early papers were devoted to the analysis of point clouds from airborne sensors. In that case the point density on the mapped objects is low, and only a few categories of objects can be reliably detected (such as trees, buildings). Thus, one of the most popular applications is detection and modeling of urban buildings. These methods (Oude Elberink and Vosselman, 2009; Kluckner and Bischof, 2010; Lafarge et al., 2010) aim to detect individual roofs and model them by fitting polyhedral models satisfying natural geometric constrains such as edge parallelism.

The analysis of point clouds obtained with terrestrial mapping systems is a relatively new research topic. Rutzinger et al. (2009) have applied region growing to automatically extract vertical walls from terrestrial and airborne laser scanning data. It has then been extended to individual tree detection (Rutzinger et al., 2010) by first detecting planes and then selecting non-planar connected components with high roughness and low point density ratio. Another technique has been developed for the detection of poles (Brenner, 2009). Poles are defined as upright structures of a certain maximum diameter, surrounded by empty space. An alternative procedure for pole detection has been introduced by Lam et al. (2010). It is based on robust vertical line fitting. Patterson et al. (2008) introduced an object detection technique that combines bottom-up and top-down descriptors. In a first stage object hypotheses are proposed by searching for similar local descriptors in a pre-computed dictionary of descriptors, for which the correct labels are known. Then each hypothesis is verified by matching with extended Gaussian images of objects from the training dataset. The common weakness of all those methods is that they are tailored to specific classes with favorable geometric properties, and therefore cannot easily be extended to other classes and to the multiclass case.

A framework for multiclass object detection has been introduced by Golovinskiy et al. (2009). This algorithm consists of the four steps: background subtraction, generation of object hypotheses, feature computation, and classification. Each step reveals higher-level information about the objects for the cost of losing some of them. The authors report that 92% of the ground truth objects are retained after the first step, and almost all of them (90%) are located correctly on the second step. Still, on the final step only 60% of the objects are located and classified correctly, thus the weakness of the method is in the two last steps.

The problem of detection and recognition of free-form objects in 3D point clouds has received attention in computer vision (Johnson and Hebert, 1999; Mian et al., 2005). In contrast to category-level recognition described above, the focus is on detecting specific objects from a set of models with little noise. Slightly transformed CAD models are typically used for testing. The main challenge addressed by those methods is partial occlusion. Typical solutions are variants of keypoint matching, followed by geometric verification of the object shape via robust fitting.

Recently, the bag-of-features (BoF) method has been adapted to also recognize 3D shapes (Toldo et al., 2009). The cloud is represented as unordered set of its local descriptors. Its advantage is the ability to match articulated objects. Knopp et al. (2010) in the implicit shape models (ISM) enforce spatial consistency as well, which gives the model more discriminative power while keeps its robustness against articulation and noise. To the best of our knowledge, such methods have not yet been applied to complex real-world point clouds such as scanned urban environments, which exhibit high intra-class variability as well as significant amount of measurement noise. We describe how to apply them in practice in Section 3.2.

2.2 3D Keypoint detectors and Shape Descriptors

Part-based methods use keypoint detection and description at their cores. Keypoint detectors aim to detect those local regions in 3D point clouds which could be used for informative description of objects by 3D shape descriptors robust to certain kinds of transformations. In this subsection we briefly review such techniques.

Various keypoint detection techniques for 3D point clouds have been introduced recently (Fadaifard and Wolberg, 2011; Steder et al., 2010). We also refer to evaluations of keypoint detectors (Mian et al., 2010; Salti et al., 2011; Yu et al., 2011). The majority of 3D detectors and descriptors are designed either for triangle meshes or for range images, so it is not clear how to adapt them to raw point clouds, unless they could be triangulated easily, which is rarely the case for urban scans that typically contain scattered points and small objects. Many different local and global shape descriptors have been developed. The most popular ones are spin images (SI) (Johnson and Hebert, 1999) and extended Gaussian images (Horn, 1984); new descriptors have also been introduced recently: angular spin images (Endres et al., 2009), FPFH (Rusu et al., 2009), and 3D SURF (Knopp et al., 2010).

3D point clouds obtained by mobile mapping systems are scaled in world coordinate units. This fact is used to compute metric properties, such as estimated volume or average height above the ground (Golovinskiy et al., 2009; Shapovalov et al., 2010). Further useful features include spectral features that measure whether the neighborhood of a point is locally planar, linear, or scatter (Medioni et al., 2000), as well as the direction of the normal vector relative to the horizontal plane (Triebel et al., 2006).

3 VOTING-BASED DETECTION FRAMEWORK

In this section we describe the details of our processing pipeline. We point out that the system is fully automatic. A user only needs to set some parameter values for detection. To train the detector only few training examples of the desired object classes are needed. The detector aims to localize the centroids of all the objects of the target classes present in the test point cloud, and to determine their class labels. The task is split into two consecutive steps, namely *object hypotheses generation* and *recognition*. While the former is domain-specific (in this paper, urban scenes), the later is largely agnostic w.r.t. the domain.

3.1 Hypotheses generation

The goal of this step is to get a list of putative objects, represented by their associated set of 3D points. Since these hypotheses form the input for the subsequent recognition, the list of hypotheses should be complete, i.e. the emphasis is on high recall. Note that a hypothesis can contain more than one object.

Similar to Golovinskiy et al. (2009), we use domain knowledge to filter out parts of the point cloud which do not contain any of our objects. First, large horizontal planes are deleted from the scene, since they are likely to represent roofs or ground. The scene is split into a regular grid of small cells, and robust plane fitting is carried out independently for each cell with RANSAC. Neighboring cells overlap to avoid discretization artifacts. If RANSAC finds a near-horizontal plane, which fits a sufficiently large fraction of points (in our implementation $>30\%$), then those points are assumed to be either part of the ground or part of a flat roof, and are excluded from further consideration.

Next, connected components are extracted such that every point in a component has at least one neighbor within a given radius (set to 0.25 m in our implementation). The components are then filtered with three conditions: (i) components with too few points are discarded, since small segments tend to be noise; (ii) components with too large extent are discarded, which gets rid of most building walls; (iii) components located too high above the ground are discarded to weed out structures on roofs, in which we are not interested. To reliably define the ground level we use street axes from OpenStreetMap.[1] We point out that the filtering steps are tailored to urban areas and may have to be adapted to other domains. An example is shown in Figure 2. The remaining components after filtering are considered hypotheses which might contain one or more objects of interest. In practice most components contain a single object or a small number of objects. In contrast to previous work we do not attempt to split each component into single objects, but defer the decision to a later stage, where more information is available.

3.2 Recognition

This stage uses object hypotheses as input and aims to recognize and localize objects within each component. To this end we adopt the implicit shape model (ISM) framework. This method, which can be seen as a combination of visual dictionaries (Sivic and Zisserman, 2003) and the generalized Hough transform, was first introduced by Leibe and Schiele (2003), and later extended to 3D mesh models by Knopp et al. (2010), who used voxel-based local descriptors. To the best of our knowledge it has not yet been applied to real-world 3D outdoor data. ISM first has to be trained in an offline learning stage, to be later applied for classification.

The training stage requires a number of training examples for each class, in our case 3D point clouds of class exemplars, which

[1]http://www.openstreetmap.org

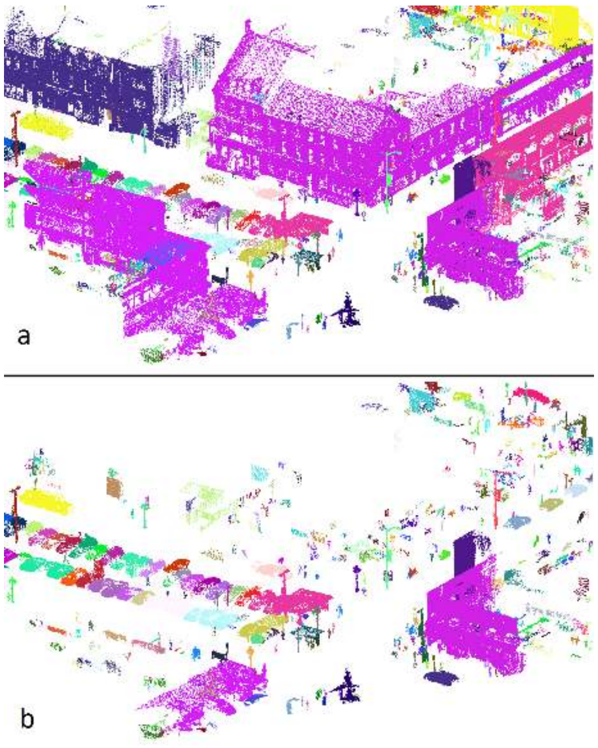


Figure 2: 3D point cloud before filtering (a) and after filtering (b). Different colors correspond to different connected components

do not need to be perfectly segmented. First, keypoints are extracted, and their neighborhoods are represented by local descriptors. Then the descriptors are clustered to obtain a dictionary of *geometric words* stored together with their possible displacements from the object center.

3.2.1 Local description We found that extraction of few well-defined keypoints using only local geometric properties of the point cloud is quite unstable due to noise and occlusions. We thus prefer to sample relatively large number of keypoints at random locations. As descriptor we employ a variant of the popular spin image (SI) descriptor (Johnson and Hebert, 1999), which is known to be rather robust to noise and can be computed from raw 3D points. Intuitively, spin image encodes the distribution of surface normals in a local neighborhood. We refer to the original publication for details. The normals are estimated from the points by moving least squares (McLain, 1976). As reference direction we choose the vertical of the world coordinate system, which is always known in surveying applications.

An issue with the conventional SI descriptor is that it does not take into account variations in point density, hence density differences will reduce the similarity between descriptors. In mobile mapping data the density can vary enormously depending on the distance from the sensor, the surface orientation, and the material properties. We therefore propose to normalize the descriptor to a fixed density, by dividing through the number of points in the local neighborhood. Furthermore, the SI descriptor suffers from the $180°$ ambiguity of the normal vectors. This is usually solved by flipping the normals accordingly (assuming that the angle between the normals of nearby points should be $< 90°$). However that approach has a disadvantage: one can invert all normal directions and the solution also will be correct (but produce different SI descriptors). Another approach is to direct all normals outwards form (or inwards to) the object center, but that does not work in case of components with multiple objects (since the cen-

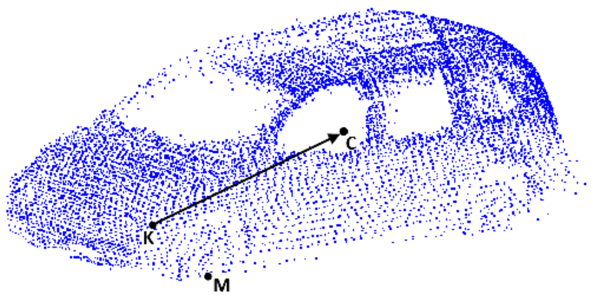


Figure 3: Illustration of keypoint representation. Point K is a keypoint, C the object center, M is the 3D point with minimal z coordinate. For each keypoint we store the descriptor matrix, the orientation vector, the keypoint's displacement from the object center $\vec{KC}$, and a relative height of the keypoint $dz = (C_z - M_z)$

ter of mass is shifted). We also could not use the sensor path because the point cloud often (and also in our data) consists of multiple co-registered scans. We thus mirror the normal vectors during descriptor computation and include *both* versions in the descriptor to achieve invariance against flipping.

In addition to the SI, we enrich the keypoint representation with further information, namely (i) the principal direction of the points in a local neighborhood, projected into the ground $(xy-)$ plane, (ii) the relative height of the keypoint (measured w.r.t. the lowest z-coordinate in the connected component), and (iii) the keypoint's offset from the object center. Please see Figure 3 for details. Clustering of the SI descriptors with k-means (Steinhaus, 1956) yields a dictionary of *geometric words*, i.e. clusters of mutually dissimilar local point configurations.

3.2.2 Voting-based localization

In the detection stage, keypoints are extracted at random locations in a given connected component, and their descriptors are computed as described above. Each descriptor is matched to a geometric word in the dictionary and casts votes for the location of the object center. Rather than let all keypoints vote equally, it has proven beneficial to weight the votes according to both the distribution of frequencies of the classes and the relative keypoint height. In details,

$$W(w_j, k_i, c_l) = W_{\text{st}}(w_j, c_l) \cdot W_{\text{height}}(w_j, k_i) \;, \qquad (1)$$

where c_l is the class label, $W_{\text{st}}(w_j, c_l)$ is the statistical weight and $W_{\text{height}}(w_j, k_i)$ is the height weight. In turn, the statistical weight is

$$W_{\text{st}}(w_j, c_l) = \frac{1}{N_w(c_l)} \cdot \frac{1}{N(w_j)} \cdot \frac{\frac{N(w_j, c_l)}{N(c_l)}}{\sum\limits_{c_n \in C} \frac{N(w_j, c_n)}{N(c_n)}} \;, \qquad (2)$$

where $N_w(c_l)$ is the number of clusters that contain votes for the class c_l, $N(w_j)$ is the number of elements in the cluster w_j, $N(w_j, c_l)$ the number of elements in the cluster w_j which vote for the class c_l, and finally $N(c_l)$ is the total number of votes for the class c_l in the whole vocabulary. The first term makes the weight value insensitive to the number of words (clusters) supporting a class in the training set, so helps to avoid bias towards classes with many different words, as in (Knopp et al., 2010). The second term normalizes the number of votes from each word. The last term estimates the probability that the word w_j votes for the class c_l. The height weight is defined as

$$W_{\text{height}}(w_j, k_i) = \exp\left(\frac{(h(k_i) - h(w_j))^2}{\sigma^2_{height}}\right). \qquad (3)$$

Here $h(k_i)$ is the relative height of the keypoint k_i from the lowest point of the whole connected component, $h(w_j)$ is the height of the cluster element w_j, σ_{height} is a smoothing parameter. This weight reflects the probability to encounter a specific geometric word at height h. Voting proceeds independently in a separate voting space for each class. After voting, local maxima are detected by mode search using mean-shift (Comaniciu and Meer, 2002) to obtain a list of potential object centers with their associated (pseudo-)probabilities for each class. Finally, we run inter-class non-maxima suppression in 3D space and discard modes with too low densities, to obtain a list of object center locations. Note that the procedure can extract in one connected component multiple objects belonging to either the same or different classes. It is also possible that no object center at all is found within a hypothesis, indicating that the connected component originates from an object of an unspecified class or from noise (e.g. pedestrians captured by the mapping system).

4 EXPERIMENTS

This section is organized as follows: first we describe the data, then we present our results separately for the hypotheses generation step and for the subsequent recognition step, and finally we discuss implementation details.

The experimental evaluation concentrates on the *Ohio* dataset, which has also been used by Golovinskiy et al. (2009) and is the most relevant public benchmark for the problem we tackle. The dataset includes 15 tiles, 100×100 meters each, scanned in Ottawa city. It has been obtained by stitching terrestrial and airborne laser scanning data (see Figure 4). A typical scene contains urban objects like buildings, trees, cars, poles etc. The authors kindly provided us with the ground truth annotation, so that we could compare directly against their method. The annotation includes a centroid for each object, its radius and its class label.

So far we have focused on *cars* and *light poles*. The *car* class is challenging because of high intraclass variation (we do not impose any limitations on the car type). Still, the centroid position is usually stable for cars, which is a prerequisite for ISM-type algorithms. Instances of the *light pole* class are quite similar to each other, however there are other "cylindrical" classes in the dataset (*lamp post*, *traffic light*, *traffic sign*), which make light pole detection non-trivial. In total the annotated portion of the dataset includes 235 cars and 73 light poles.

An important requirement for the hypotheses generation stage is not to miss any relevant objects, because at the next stage those

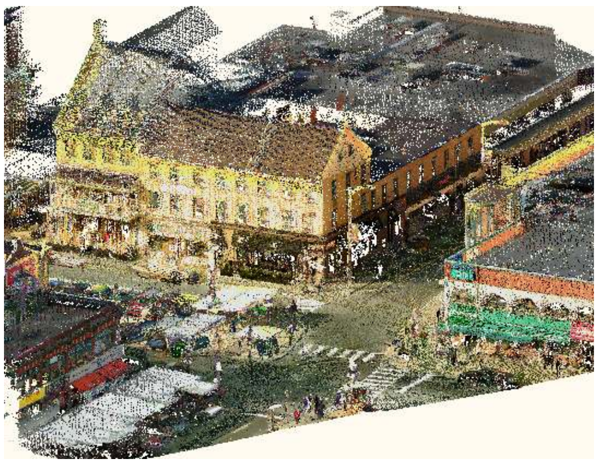


Figure 4: Typical example of an urban scene

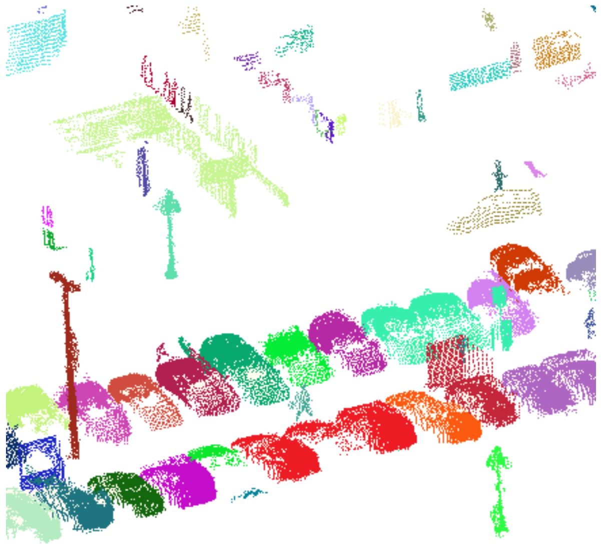


Figure 5: Typical result of the hypotheses generation stage. Different colors denote different connected components. The red hypothesis in the image center is caused by three different cars

mistakes could not be fixed. The rejected points (i.e. those which did not fall into any connected component) would not participate in the voting for the centroid. As a measure of how complete the hypotheses are, we compute the fraction of correct objects, which fall into at least one connected component. For each ground truth object we define a circular region by the given object center and radius. We consider detection correct if at least 50 of its points fall within the class-specific radius around the ground truth object center. In contrast to Golovinskiy et al. (2009), our hypotheses do not necessarily correspond to only one object. During the voting step we localize objects' centroids within the connected components. The experiment shows that the hypotheses generation stage retains 96% of the cars and 93% of the light poles. We conclude that this stage is not a bottleneck of the processing pipeline, which is in line with conclusions of Golovinskiy et al. (2009). Example connected components are shown in Figure 5.

To evaluate design choices for the voting stage we first test our spin image modification, then we investigate the influence of the vocabulary size and the effect of the new height weight. Finally we provide an end-to-end comparison of our approach against the state of the art for the dataset.

First one has to construct a dictionary of geometric words by clustering keypoint descriptors. The quality of detection depends on the type of descriptors. Our experiments show significant improvement when using the modified spin images (Section 3.2.1), see Figure 6(a).

We manually extract exemplars of the two object classes to construct the dictionary. Note we use only a few exemplars for each class (1–5), which significantly reduces the annotation effort for training. During voting, every keypoint *must* be associated with a geometric word. Therefore we have also added the background classes which had previously produced most false detections (*wall* and *tree*).

The size of the dictionary is an important parameter. Given 8000 keypoints extracted from the training objects, we conduct a series of experiments with different number of clusters (from 100 to 1000), see Figure 6(b). We conclude the a dictionary size of 100-300 (1-3% of the total number of keypoints) is the best choice.

Method	Dataset	*Cars*		*Light poles*	
		P	R	P	R
Golovinskiy	TEST	0.50	0.62	0.45	0.62
ISM	TEST	0.66	0.70	0.69	0.80
ISM	FULL	0.68	0.71	0.72	0.82

Table 1: Comparison between (Golovinskiy et al., 2009) and the proposed method (denoted as ISM, including both modified SIs and the height weight) in terms of [P]recision and [R]ecall. TEST denotes results for only the original test set, FULL for the combined training and test sets

An evaluation of the proposed height weight is presented in Figure 6(c). This weight helps to detect cars better due to their small range of height above the ground. However, the weight decreases the performance for light poles in the high-precision regime. We explain this by the presence of similar object parts at different heights.

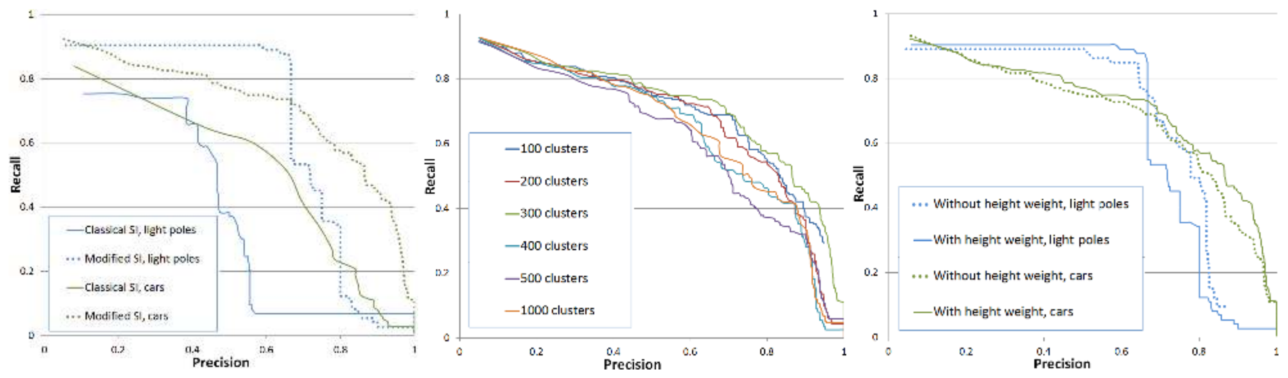
We compare the end-to-end results of our approach with those of Golovinskiy et al. (2009), on the same dataset, see Table 1. Overall, the dataset contains 235 cars and 73 light poles. It is split into a training set for learning classifiers (125 cars, 39 light poles) and a test set for evaluation (110 cars, 34 light poles). However, as mentioned before, our method does not require a large training set. A few exemplars are enough, so we also could test our approach on the full dataset (previous "training" and "test" parts combined). All our training instances have been extracted completely randomly from external data outside the annotated portion. They are present neither in the "test" nor in the "training" part.

The described approach for object detection in 3D point clouds has been implemented as a fully automatic processing pipeline. The system is completely written in C++, building on efficient libraries, most notably the Point Cloud Library[2], and can handle millions of 3D points. The total processing time for a tile (100×100 m^2, $\approx$4M points) is between 5 and 10 minutes on an Intel QuadCore 2.4 GHz machine with 4 GB of RAM. Empirically, the method is quite stable to parameters variation, so it does not require careful tuning. However, we point out the dependency between the number of training exemplars and the number of clusters in the dictionary. As more training objects are added, the number of geometric words should grow to accommodate the increased shape variability.

5 SUMMARY

We have presented a method for object detection and recognition in 3D point clouds. The method is based on implicit shape models, which recognize objects by voting for their centroid locations. We have introduced and evaluated extensions to the spin image descriptor as well as the weighting scheme for the voting stage. Experimental evaluation shows that these extensions significantly boost recognition performance. We compare our results with the state of the art on the *Ohio* dataset and get improvements of 8–24% in both precision and recall. Since the algorithm is based on a voting procedure for location of the object center, it is well-suited for exemplar-based detection. Small number of required training examples makes the approach useful for practical applications. On the downside, its main limitation are objects classes with strongly varying center and/or badly defined shape.

[2] `http://pointclouds.org`



(a) (b) (c)

Figure 6: Comparison of detection results for different system settings. (a) Classic vs. modified spin images. (b) Different numbers of geometric words (class *cars*). (c) Height weight vs. no height weight

References

Brenner, C., 2009. Extraction of features from mobile laser scanning data for future driver assistance systems. In: AGILE'09. 2

Comaniciu, D. and Meer, P., 2002. Mean Shift: A robust approach toward feature space analysis. IEEE T Pattern Anal 24(5), pp. 603–619. 4

Endres, F., Plagemann, C., Stachniss, C. and Burgard, W., 2009. Unsupervised discovery of object classes from range data using latent Dirichlet allocation. In: RSS'09. 2

Fadaifard, H. and Wolberg, G., 2011. Multiscale 3d feature extraction and matching. In: 3DimPVT'11. 2

Golovinskiy, A., Kim, V. G. and Funkhouser, T. A., 2009. Shape-based recognition of 3d point clouds in urban environments. In: ICCV'09. 1, 2, 3, 4, 5

Horn, B. K. P., 1984. Extended Gaussian images. Proc IEEE 72(2), pp. 1671–1686. 2

Johnson, A. E. and Hebert, M., 1999. Using spin images for efficient object recognition in cluttered 3d scenes. IEEE T Pattern Anal 21(5), pp. 433–449. 2, 3

Kluckner, S. and Bischof, H., 2010. Image-based building classification and 3d modeling with super-pixels. In: PCV'10. 2

Knopp, J., Prasad, M., Willems, G., Timofte, R. and Van Gool, L., 2010. Hough transform and 3d SURF for robust three dimensional classification. In: ECCV'10. 1, 2, 3, 4

Lafarge, F., Descombes, X., Zerubia, J. and Deseilligny, M. P., 2010. Structural approach for building reconstruction from a single DSM. IEEE T Pattern Anal 32(1), pp. 135–147. 2

Lam, J., Greenspan, M., Harrap, R., Kusevic, K. and Mrstik, P., 2010. Urban scene extraction from mobile ground based LiDAR data. In: 3DPVT'10. 2

Leibe, B. and Schiele, B., 2003. Interleaved object categorization and segmentation. In: BMVC'03. 3

McLain, D. H., 1976. Two dimensional interpolation from random data. Comput J 19(2), pp. 178–181. 3

Medioni, G., Lee, M.-S. and Tang, C.-K., 2000. Computational Framework for Segmentation and Grouping. Elsevier Science. 2

Mian, A., Bennamoun, M. and Owens, R., 2010. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. Int J Comput Vision 89(2-3), pp. 348–361. 2

Mian, A. S., Bennamoun, M. and Owens, R. A., 2005. Automatic correspondence for 3d modeling: an extensive review. Int J Shape Modeling 11(2), pp. 253–291. 2

Oude Elberink, S. and Vosselman, G., 2009. Building reconstruction by target based graph matching on incomplete laser data: Analysis and limitations. Sensors 9(8), pp. 6101–6118. 2

Patterson, A., Mordohai, P. and Daniilidis, K., 2008. Object detection from large-scale 3d datasets using bottom-up and top-down descriptors. In: ECCV'08. 2

Rusu, R. B., Blodow, N. and Beetz, M., 2009. Fast point feature histograms (FPFH) for 3d registration. In: ICRA'09. 2

Rutzinger, M., Oude Elberink, S., Pu, S. and Vosselman, G., 2009. Automatic extraction of vertical walls from mobile and airborne laser scanning data. In: ISPRS Laser scanning workshop '09. 2

Rutzinger, M., Pratihast, A., Oude Elberink, S. and Vosselman, G., 2010. Detection and modelling of 3d trees from mobile laser scanning data. In: Close Range Image Measurement Techniques, pp. 520–525. 2

Salti, S., Tombari, F. and Stefano, L. D., 2011. A performance evaluation of 3d keypoint detectors. In: 3DimPVT'11. 2

Shapovalov, R., Velizhev, A. and Barinova, O., 2010. Non-associative markov networks for 3d point cloud classification. In: PCV'10. 2

Sivic, J. and Zisserman, A., 2003. Video Google: A text retrieval approach to object matching in videos. In: ICCV'10. 3

Steder, B., Grisetti, G. and Burgard, W., 2010. Robust place recognition for 3d range data based on point features. In: ICRA'10. 2

Steinhaus, H., 1956. Sur la division des corps matériels en parties. Bull Acad Polon Sci, Cl III/4 pp. 801–804. 4

Toldo, R., Castellani, U. and Fusiello, A., 2009. A bag of words approach for 3d object categorization. In: MIRAGE'09. 1, 2

Triebel, R., Kersting, K. and Burgard, W., 2006. Robust 3d scan point classification using associative markov networks. In: ICRA'06. 2

Yu, T.-H., Woodford, O. J. and Cipolla, R., 2011. An evaluation of volumetric interest points. In: 3DIMPVT'11. 2