Automatic object recognition within point clouds in clustered or scattered scenes

Bae, Egil

Egil Bae, "Automatic object recognition within point clouds in clustered or scattered scenes," Proc. SPIE 11538, Electro-Optical Remote Sensing XIV, 1153804 (20 September 2020); doi: 10.1117/12.2574119

SPIE.

Event: SPIE Security + Defence, 2020, Online Only

Automatic object recognition within point clouds in clustered or scattered scenes

Egil Bae

Norwegian Defence Research Establishment (FFI), P.O. Box 25, 2027 Kjeller, Norway

ABSTRACT

We consider the problem of automatically locating, classifying and identifying an object within a point cloud that has been acquired by scanning a scene with a ladar. The recent work [E. Bae, Automatic scene understanding and object identification in point clouds, Proceedings of SPIE Volume 11160, 2019] approached the problem by first segmenting the point cloud into multiple classes of similar objects, before a more sophisticated and computationally demanding algorithm attempted to recognize/identify individual objects within the relevant class. The overall approach could find and identify partially visible objects with high confidence, but has the potential of failing if the object of interest is placed right next to other objects from the same class, or if the object in interest is scattered into several disjoint parts due to occlusions or slant view angles. This paper proposes an improvement of the algorithm that allows it to handle both clustered and scattered scenarios in a unified way. It proposes an intermediate step between segmentation and recognition that extracts objects from the relevant class based on similarity between their distance function and the distance function of a reference shape for different view angles. The similarity measure accounts for occlusions and partial visibility naturally, and can be expressed analytically in the distance coordinate for azimuth and elevation angles within the field of view (FOV). This reduces the dimensions for which to search from three to two. Furthermore, calculations can be limited to parts of the FOV corresponding to the relevant segmented region. In consequence, the computational efficiency of the algorithm is high and it is possible to match against the reference shape for multiple discrete view angles. The subsequent recognition step analyzes the extracted objects in more details and avoids suffering from discretization and conversion errors. The algorithm is demonstrated in various maritime examples.

Keywords: segmentation, object recognition, scene analysis, point cloud, ladar

1. INTRODUCTION

Laser detection and ranging (ladar) is an imaging technique for generating detailed three-dimensional (3D) information of a scene. Point clouds convey accurate information of the shape and scale of objects in the scene. This makes them useful for classifying and identifying such objects, either by a human or an algorithm running on a computer. Algorithms are particularly well-suited for analyzing point clouds due to their ability to accurately sense relative distance in 3D space. In contrast, the human two-dimensional visual system is more inclined to suffer from perspective bias when analyzing such 3D point clouds. Ladar imaging have found use cases among a wide range of applications, such as autonomous automobiles, picking robots, airborne surveillance and satellite imaging.

Our previous paper ref.[1] presented algorithms for locating, segmenting and recognizing objects within point clouds acquired by ladar in maritime, urban and rural scenes. The point cloud was first segmented into multiple classes, where one of the classes contained objects with similar characteristics to the object of interest. Afterwards, individual objects within the relevant class were analyzed by a more computationally demanding algorithm that utilized the full 3D shape of the object, its simulated shadow and optionally certain intrinsic reflectance properties. The algorithms were designed for complex point clouds with irregular neighborhood structures, such as those acquired by the line scanning principle, and were based on solving various minimization problems over graphs. Although the algorithm works well in most cases, it has two main limitations that will be dealt with in this paper:

Further author information:
E-mail: Egil.Bae@ffi.no, Telephone: +47 63 80 73 99

Electro-Optical Remote Sensing XIV, edited by Gary W. Kamerman, Ove Steinvall, Proc. of SPIE
Vol. 11538, 1153804 · © 2020 SPIE · CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2574119

1. *Objects merged with other objects from the same class:* If the object of interest is placed in close contact with one or more objects of the same class, the segmentation step may not be able to separate those objects, instead regarding them as one single larger object. This may occur, for instance, if two vehicles are placed right next to each other. Another example is a ship at quay.

2. *Objects divided into disjoint parts:* Under certain circumstances, the object of interest may be divided into two or more disjoint parts that are separated in distance and appear scattered in the point cloud. For instance, if a ship is being observed straight from the front, points will get reflected from the hull and the area around the bridge, but due to their large distance from each other the hull and bridge will appear as two different objects in the point cloud. Another example would be an object whose middle part is occluded from the line of sight. These scenarios make it challenging to determine which object parts belong together.

This paper proposes improvements that allows the overall algorithm to handle both clustered and scattered scenes in a unified way. It consists of an intermediate step between segmentation and recognition that extracts objects from the relevant class based on similarity between their distance function and the distance function of a reference shape for different view angles. The similarity measure accounts for occlusions and partial visibility naturally, and is formulated as an optimization problem in the distance coordinate that can be solved analytically for each azimuth and elevation angle within the field of view (FOV). This implicit handling of the distance coordinate reduces the dimensions for which one must search from three to two. In consequence, the computational efficiency of the algorithm is high and it is possible to match against the reference shape for multiple discrete view angles. Only the parts of the FOV corresponding to the relevant segmented region of the point cloud need to be considered in the computation. The subsequent recognition step analyzes the extracted objects in more details and avoids suffering from discretization and conversion errors. The algorithm is demonstrated on some illustrative examples.

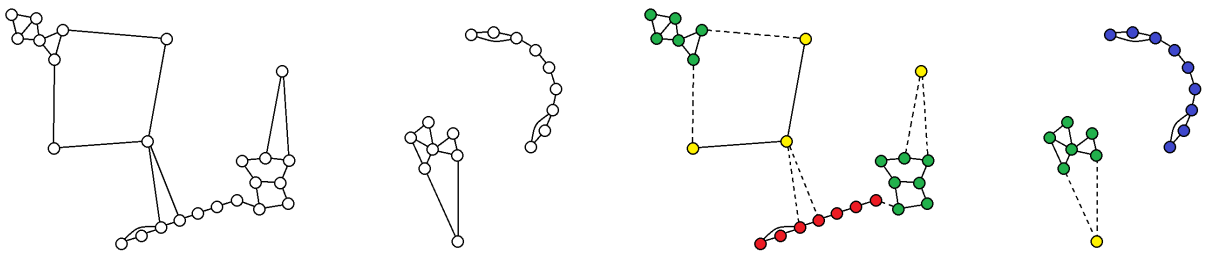
There has also been a lot of earlier work on point cloud segmentation and object recognition, although the above scenarios are common challenges. A closer review will be given in the technical section. The article[2] proposed to segment objects in ladar images from the ground plane based on morphological techniques. In the end, the objects were classified into clutter and human-made objects based on their geometrical structure, and the human-made objects were further classified based on their dimensions. Another closely related work[3] is on segmentation and recognition of vessels in maritime environments. Their data set was acquired by a flash ladar and the vessels were first extracted from the sea based on 2D image segmentation techniques. In the end, the vessels were classified based on their full 3D shape. There has been proposed several approaches for point cloud segmentation, e.g. ref.[4–8] to name a few. Most of them either interprets the point cloud as an image of 2D pixels or 3D voxels, and applies established image analysis techniques, such of morphological operations. Convolutional neural networks (deep learning) have recently been applied on point clouds, but they also require a regular 2D grid ref.[9–11] or 3D grid ref.,[11,12] as well as a large amount of training data.

This article is organized into four main sections. Section 2 gives a brief review of the segmentation step, Section 3 presents the new object extraction step and Section 4 gives a brief review of the recognition step. Experimental results are summarized in Section 5, but are also presented in the other sections.

2. POINT CLOUD SEGMENTATION

The point cloud is initially segmented into different regions containing different classes of objects, according to our previous work.[1] A brief review of the method is given in order to identify some key limitations that will be dealt with in the next section. The segmentation approach falls within a newer class of methods that process the 3D points directly on a graph, without first converting them to a regular 2D or 3D grid. Consequently, no information is lost and situations with depth overlap and motion are naturally handled. The algorithm is based on solving a so-called variational problem on the graph, which is a recent concept that is known to give very good solutions to various tasks in pattern recognition. A particularly efficient and robust algorithm was developed in.[13]

Assume that we are given the coordinates of N points in 3D space, obtained by a ladar or another similar imaging technique. The N points will be denoted as $\mathbf{p}^1, \mathbf{p}^2, ..., \mathbf{p}^N$ in the rest of this article. In order to formulate



(a) graph (b) graph after segmentation

Figure 1: Example of a graph with 2D points as vertices, where each vertex is connected to its two nearest neighbors. Note that some vertices may still be connected to more than two vertices and the graph need not be connected, but may consist of several connected components. The images are reproduced from the author's previous work[13] under the terms of the Creative Commons Attribution Licence.

segmentation of the points as a minimization problem, the points are first organized in an undirected graph. A graph is a set of vertices, of which 3D points is a special case, together with a pairwise interconnection between the set of vertices. The set of all vertices is denoted by V. A pair of vertices that are connected is called an edge in the graph. The set of all edges is denoted by E. Two vertices $\mathbf{p}$ and $\mathbf{q}$ are connected if and only if their vertex pair $(\mathbf{p}, \mathbf{q})$ is contained in the set of edges E.

Figure 1 depicts a graph where each vertex is a 2D point. The points are visualized as round balls, whose centers indicate the point coordinates. The edges are visualized as lines connecting two points with each other. In this work, the graph is constructed such that two points are connected if one of them is among the "k-nearest" neighbors of the other. That is, for each point p an edge (p, q) is constructed if q is one of the k closest points of p. We set $k = 20$ in experiments. Constructing a k nearest neighbors graph can be achieved efficiently by exploiting the redundancy between the point coordinates. A common algorithm involves first building a k-d search tree over the points and then looking for nearest neighbors by traversing the branches of the tree, see ref.[14,15] for details.

A graph that is constructed based on the k nearest neighbors may consist of several "connected components". A connected component is a subset of the graph where any two vertices (points) are connected by a path of edges, that is, it is possible to travel from one vertex to the other by only moving along the edges. The graph in Figure 1 contains three such connected components. Vertices in two different connected components are not connected by a path of edges. Therefore, constructing the graph provides some basic segmentation of the point cloud, where each connected component can be regarded as a region. Such a segmentation may be meaningful if the object is completely separated in distance from other points in the point cloud. However, this can only rarely be expected to be the case.

Our previous papers[1,13] explained how to obtain a much finer and more meaningful segmentation by solving an energy minimization problem over the graph. The main idea is to divide the points into several different regions based similarity between point pairs and how individual points relate geometrically and statistically to neighboring points. An illustration is given in Figure 1 (b), where the points are grouped into a region of low point density (yellow), and regions of high (red), medium (blue) and low (green) correlation between the coordinates of each point and its neighboring points. A similar segmentation can be constructed for 3D point clouds, where human-made structures typically have a high degree of correlation between the point coordinates and vegetation have a low degree of correlation. Individual objects from the segmented point cloud can now be extracted from the point cloud. For example the red object is separated from the surrounding objects due to their different class memberships.

Some examples on recorded 3D point clouds are shown in Figure 2. On the left is a land scene that is segmented into vegetation with leaves (light green), vegetation without leaves (dark green), the ground surface (brown) and human-made structures (blue). Marked in red is human-made structures with the size of a car.

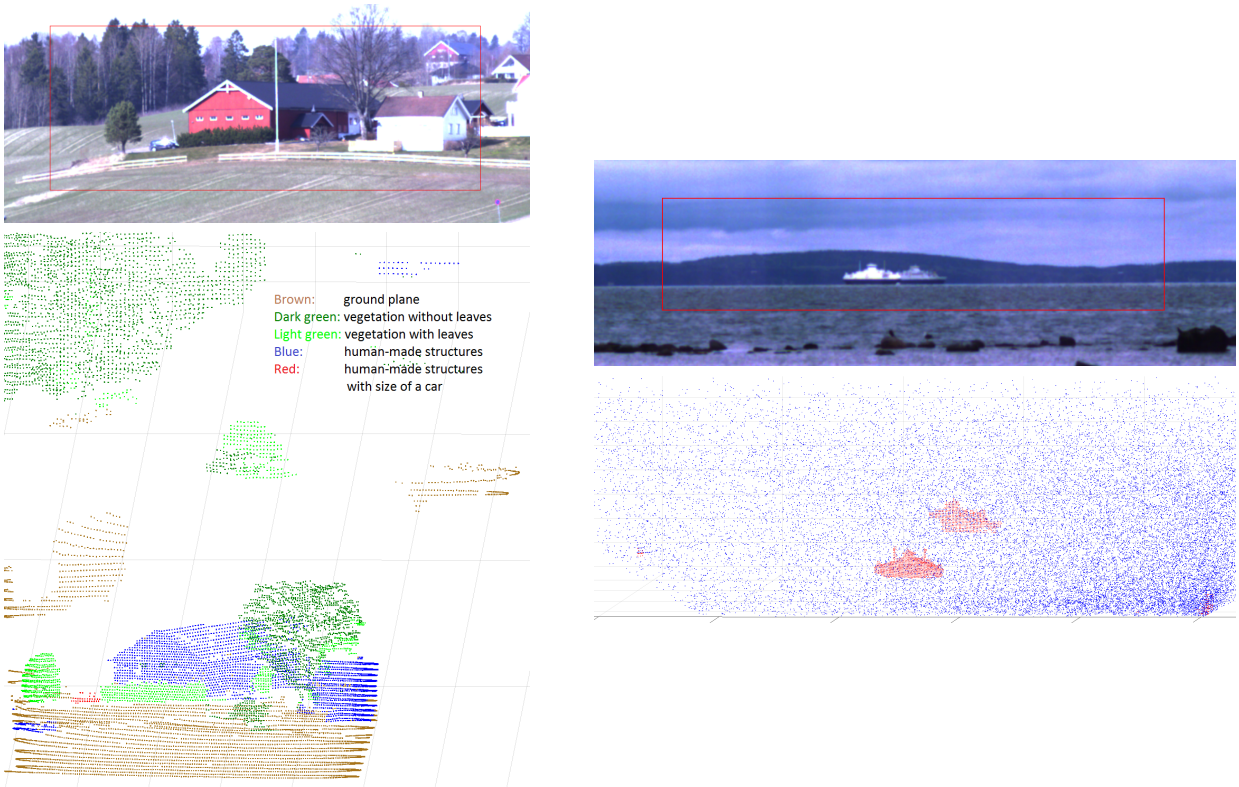


Figure 2: Left: Segmentation of point cloud acquired in land scene into vegetation, the ground surface, human-made structures and vehicles. Right: Segmentation of point cloud acquired in maritime scene into solid objects and noise.

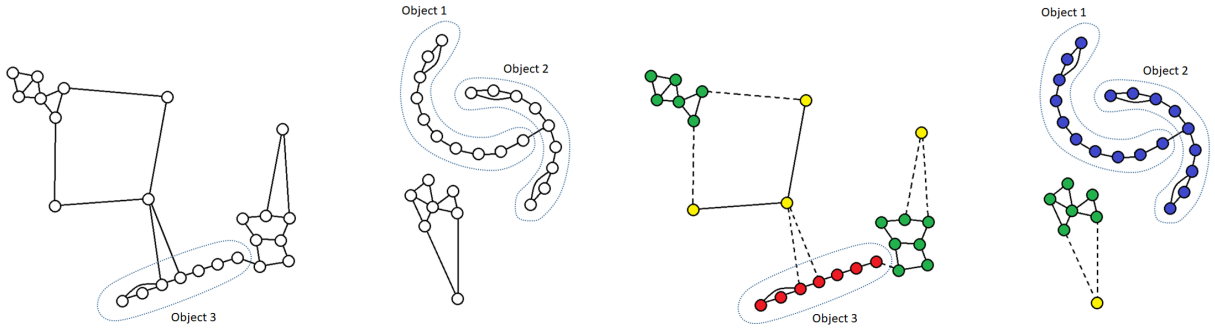
Since the car is only surrounded by objects from other classes (a bush, a tree and the ground surface) it can be extracted from the point cloud based on connected component analysis. On the left is a maritime scene that is segmented into solid objects and noisy points detected from the atmosphere (blue). Marked in red are solid objects with the size of a ship.

3. OBJECT EXTRACTION IN CLUSTERED AND SCATTERED SCENES

Under most circumstances the segmentation process itself will separate individual objects from the rest of the point cloud. By removing edges between points that are assigned to different regions (dashed lines in Figure 3), the graph gets split into several different connected components, each of which can be regarded as a separate object.

There are, however, two different scenarios that may cause the previous simple procedure to fail:

- Clustered scenes: If the object of interest is placed in close contact with one or more objects of the same class, the segmentation step may not be able to separate those objects, instead regarding them as one single larger object. This may occur, for instance, if two vehicles are placed right next to each other. Another example is a ship at quay. An illustrative example is shown in Figure 3, where object 1 and object 2 each belong to the same class. Because of their close vicinity to each other, the segmentation process merges both objects together as seen in subfigure (b). Since both objects are now connected by a path of edges, they will be regarded as one single larger object by the simple procedure described in the previous section. Another example is shown in Figure 4, where the ship at quay gets merged with the surrounding buildings in the point cloud.



(a) graph (b) graph after segmentation

Figure 3: (a) Example of a graph where two objects belonging to the same class (object 1 and object 2) are located in close vicinity to each other. This causes both objects to get connected by a path of edges when creating the graph. (b) Segmentation of graph of 2D points into regions of low density (yellow), high degree of statistical correlation of coordinates between neighboring points (red), medium correlation (blue) and low correlation (green). Due to the similar characteristics or object 1 and object 2, the segmentation algorithm is unable to separate them from each other.

- Scattered scenes: Under certain circumstances, the object of interest may be divided into two or more disjoint parts that are separated in distance and appear scattered in the point cloud. From slant view angles, the horizontal parts of an object may not be visible, and the visible vertical parts may therefore appear disjoint. Another example would be an object whose middle part is occluded from the line of sight. These scenarios make it challenging to determine which object parts belong together. See the rightmost ship in figure 4 for an example. Since the ship is viewed straight from the front, only frontmost parts of the hull and the bridge is visible. The hull and bridge appear as two different connected components in the point cloud.

This section describes a more advanced procedure for extracted objects from the relevant segmented region also under these circumstances. A point $\mathbf{P} = (p_1, p_2, p_3)$ can be represented in spherical coordinates as (d, θ, ϕ), where θ is the altitude angle, ϕ is the azimuth angle and d is the distance from the origin to the 3D point. See Figure 5 for an illustration.

$$d = \sqrt{p_1^2 + p_2^2 + p_3^2} \tag{1}$$

$$\theta = \arctan(\frac{p_2}{p_1}) \tag{2}$$

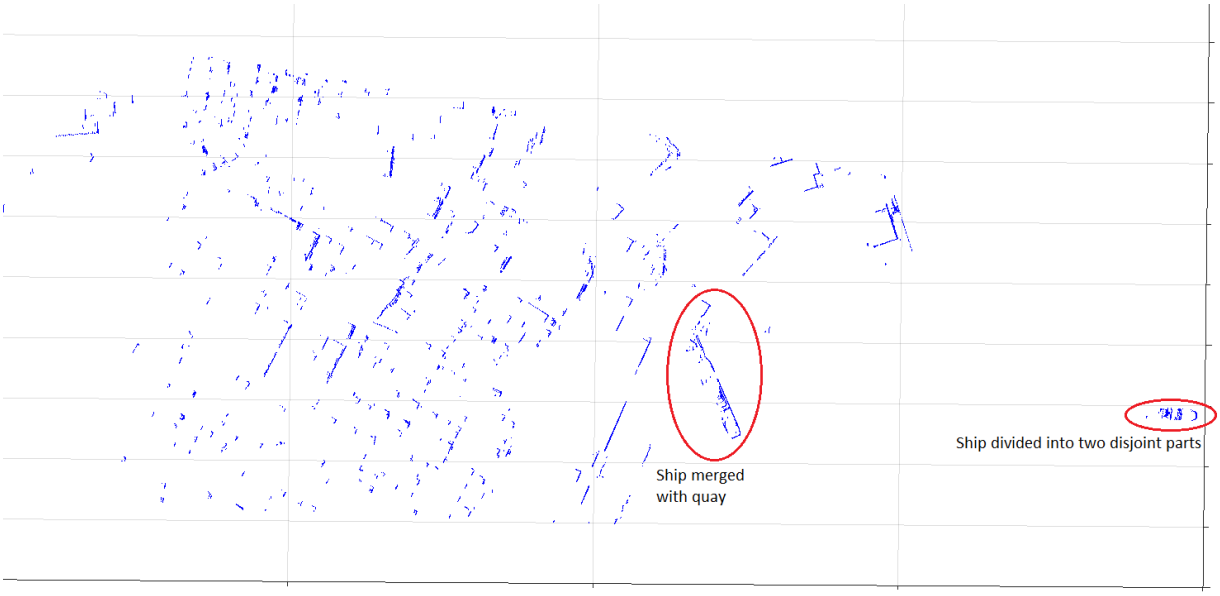
$$\phi = \arccos(\frac{p_3}{\sqrt{p_1^2 + p_2^2 + p_3^2}}) \tag{3}$$

The discrete set of points in the point cloud $\mathrm{P} = \{p_1^i, p_2^i, p_3^i\}_{i=1}^m$ can be represented in spherical coordinates as $\{d^i, \theta^i, \phi^i\}_{i=1}^m$. We would like to define the distance coordinate d implicitly as a function of the azimuth angle ϕ and elevation angle θ. The resulting distance function, called $D^P(\phi, \theta)$, will also approximate the surface that the 3D points are sampled from between the points and is defined over the entire the field of view (FOV)

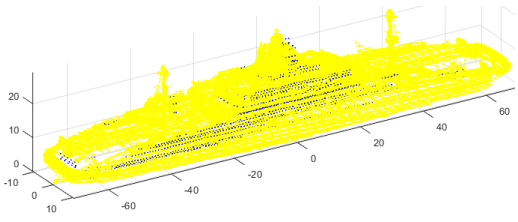
$$\mathrm{FOV} = [\theta_{\min}, \theta_{\max}] \times [\phi_{\min}, \phi_{\max}].$$

It is defined to take the value zero at areas of the FOV where no points are observed. It was explained earlier that several points may share the same azimuth and elevation angle, for instance if the laser pulse partially penetrates through leaves, thus reflecting both from the leaves and objects behind. However, under the assumption that the object of interest is non-transparent, it can in general be assumed that no points should be observed behind it. In case of ambiguity, we therefore let $D^P(\theta, \phi)$ be the distance to the point farthest away. For computational

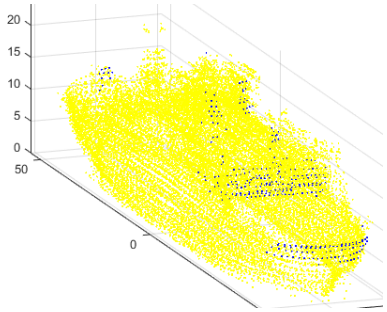
(a) Visual light image



(b) Segmented point cloud (top down view)

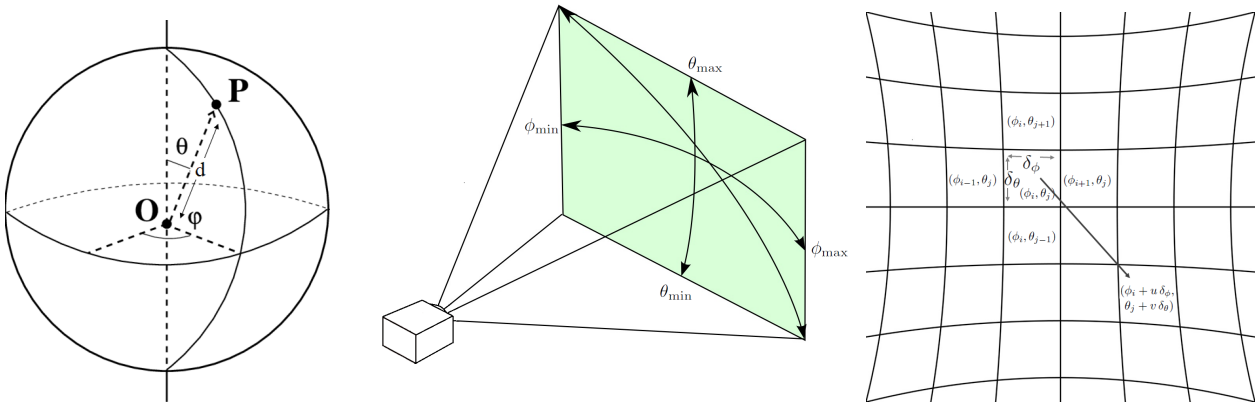


(c) **Identified ship (at quay): Bastø VI**



(d) **Identified ship viewed from front: Bastø I**

Figure 4: Example of a point cloud that illustrates both a clustered and scattered scene. A top down view of the point cloud is shown in (b). The leftmost ship is merged together with the quay and the rightmost ship is separated into two disjoint parts due to the slant view direction straight from the front. The rest of the subfigures show results of the algorithms to be presented in this paper. The algorithms are able to extract the points on the ships from the rest of the point cloud and compare them to 3D models, among other features. In subfigures (c) and (d) the observed points are marked in blue and points of the 3D model are marked in yellow.



(a) (b) (c)

Figure 5: (a) Representation of the point $\mathbf{P} = (p_1, p_2, p_3)$ using spherical coordinates (d, θ, ϕ). (b) Visualization of the field of view (FOV). (c) Discretization of the field of view.

purposes, the distance function is defined discretely over a set of "pixels" which are evenly distributed rectangularly shaped solid angles over the field of view. The azimuth and altitude angles are discretized into the intervals

$$\phi_i = [\phi_{\min} + \delta_\phi \cdot (i-1), \phi_{\min} + \delta_\phi \cdot i], \quad i = 1, ..., m \tag{4}$$

$$\theta_j = [\theta_{\min} + \delta_\theta \cdot (j-1), \theta_{\min} + \delta_\theta \cdot j], \quad i = 1, ..., n \tag{5}$$

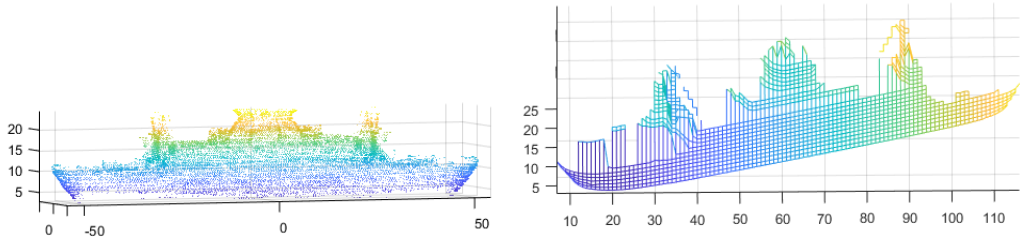
$$\tag{6}$$

The set of all "pixels" thus consists of all solid angles

$$[\phi_i \times \theta_j]_{i=1,...,m,\ j=1,...,n}$$

See Figure 5(c) for an illustration. The discrete distance function is defined to take a constant value within solid angle $\phi_i \times \theta_j$. For notational convenience, we let $D^P(i,j)$ denote the value of D^P within $\phi_i \times \theta_j$. There are several ways to compute the value of D^P for each solid angle $\phi_i \times \theta_j$, depending on whether the discrete angular resolution is higher or lower than the sampling resolution. We assume a slightly higher discrete resolution in most experiments. For solid angles $\phi_i \times \theta_j$ where sampling points are observed, we define $D^P(i,j)$ to be the largest distance of those points. Surrounding empty pixels are subsequently interpolated using a second order polynomial based on information in the filled pixels. At areas of the FOV more than two pixels away from observed points, $D^P(i,j)$ is set to zero.

We want to match the distance function $D^P(i,j)$ representation of the point cloud against a similarly constructed distance function representation of the object shape $D^M(i,j)$ for different view angles. $D^M(i,j)$ is constructed by first creating a synthetic point cloud by placing a 3D model of the object in the scene, removing all points on the 3D model that are occluded from the line of sight, calculating the spherical coordinates of all points, and then proceeding as described above for constructing the distance function. An illustration of a 3D model and the resulting distance function is shown in Figure 6. By convention, the 3D model is placed near the origin of the coordinate system by setting $\phi_{\min} = 0$ and $\theta_{\min} = 0$ when constructing $D^M(i,j)$. We now consider the issue on how to select the distance of the 3D model from the origin. We assume that relevant objects are so far away relative to their size that their shapes project orthographically down to the image plane. In consequence, varying the distance to the 3D model corresponds to enlarging or shrinking the corresponding distance function while adding or subtracting the shift in distance. To give an example, consider the distance function of the 3D model calculated from a range of, say, 2000 meters. By shrinking the distance function by



(a) (b)

Figure 6: (a) Visualization of a point cloud obtained by scanning the ferry Bastø I. (b) the corresponding distance function.

a half and subtracting 2000 meters, one then obtains the distance function of the same 3D model from a range 4000 m. The size of the distance function varies little between ranges of, say, 4000 m and 4100 m. Consequently, the distance function at 4100 m can be well approximated by adding 100 m to the distance function at 4000 m. This is especially true considering that the discretization of the distance function results in discretization errors that tend to dominate. For the same reason, the orientation of object in the scene and the orientation of the 3D model do not need to match exactly either, but only approximately. The objective of the object extraction step is not high precision, but to be able to extract objects of approximately similar shapes to the object of interest that can be further analyzed by the recognition step.

Assume first that an estimate of the object distance (within 200 meters) and the object orientation (within 10 degrees) is known. Let $D^M(i,j)$ be the distance function to the 3D model placed in the scene using the estimated object distance and orientation. We want to calculate the similarity between $D^M(i,j)$ and the distance function of the observed point cloud $D^P(i,j)$ in order to detect and extract possible candidate objects. There are two sets of unknown parameters:

- A translation of the model D^M from its default position by azimuth angle $u\,\delta_\phi$ and altitude angle $v\,\delta_\theta$, where u and v are integers. See Figure 5(c) for an illustration. Using the same notational convention as above, the translated distance function of the model is $D^M(i+u, j+v)$.
- A shift D^s of distance function $D^M(\theta+u, \phi+v) + D^s$, where D^s is the difference between the estimated and real object distance.

We will now focus on several similarity measures with increasing complexity that all handle the parameter D^s implicitly. Assume first that the translation parameters u, v are known. For notational convenience, the set of all pixels that contain points from both the translated model and the observed point cloud, i.e. where both their distance functions $D^M(i+u, j+v)$ and $D^P(i,j)$ are non-zero, is defined as

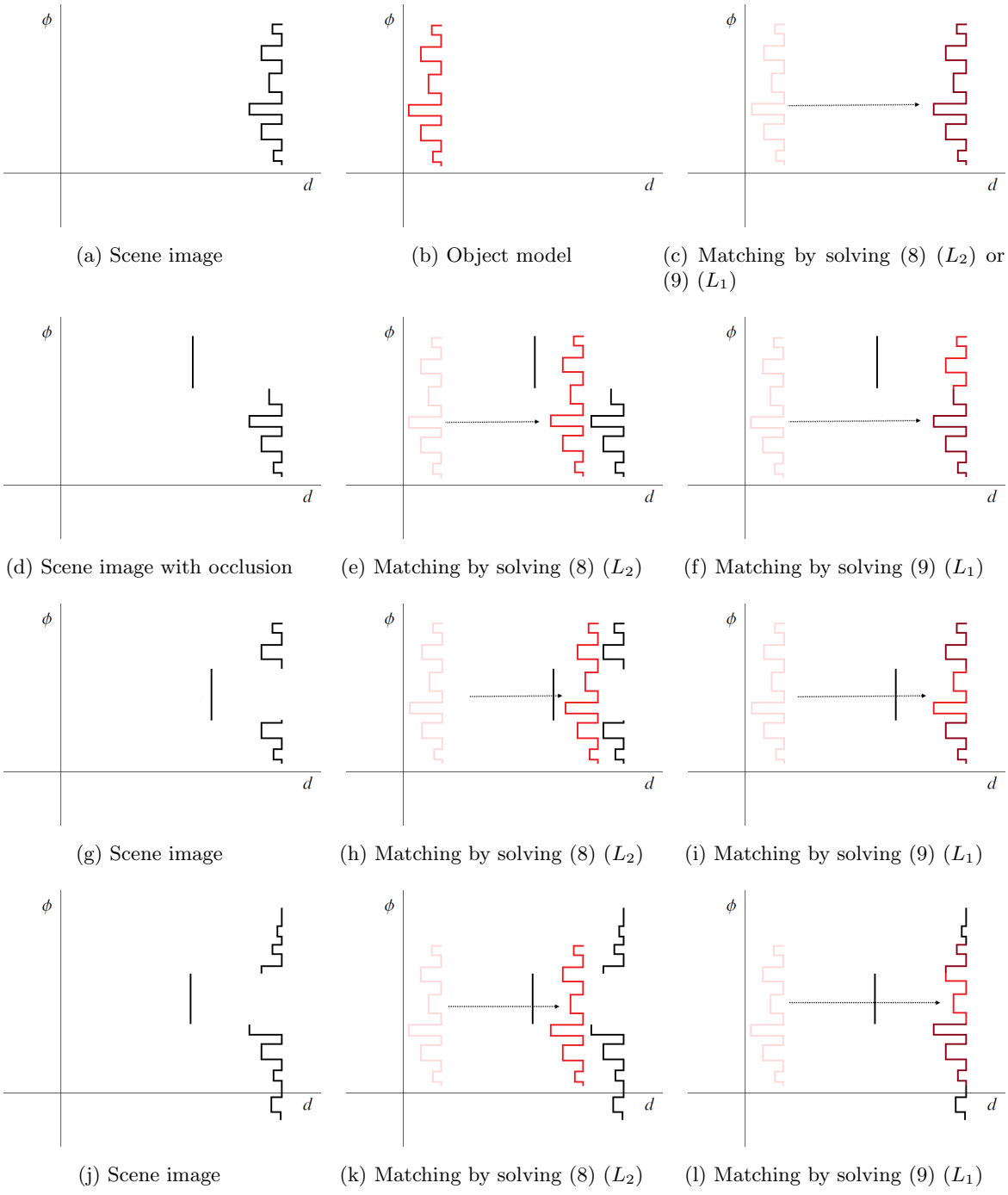
$$I_{u,v}^{M,P} = \Big\{ \text{All } (i,j) \in \mathbb{N}^2 \text{ such that } D^M(i+u, j+v) \neq 0 \text{ and } D^P(i,j) \neq 0 \Big\} \tag{7}$$

A simple way to measure the similarity is by minimizing the 2-norm between $D^M(i+u, j+v) + D^s$ and $D^P(i,j)$, weighted by the number of observed points

$$L_2(D^M, D^P; u, v) = \min_{D^s \in \mathbb{R}} \frac{\sum_{(i,j)\in I_{u,v}^{M,P}} \left| \big(D^M(i+u, j+v) + D^s\big) - D^P(i,j) \right|^2}{\sum_{(i,j)\in I_{u,v}^{M,P}} 1}. \tag{8}$$

The denominator counts the total number of contributing pixels where both D^M and D^P are non-zero. To solve the minimization problem for D^s is particularly easy in this case, as the minimizer is just the mean over non-zero pixels

$$D^s = \text{mean}_{(i,j)\in I_{u,v}^{M,P}} \big(D^P(i,j) - D^M(i+u, j+v) \big)$$



(a) Scene image (b) Object model (c) Matching by solving (8) (L_2) or (9) (L_1)

(d) Scene image with occlusion (e) Matching by solving (8) (L_2) (f) Matching by solving (9) (L_1)

(g) Scene image (h) Matching by solving (8) (L_2) (i) Matching by solving (9) (L_1)

(j) Scene image (k) Matching by solving (8) (L_2) (l) Matching by solving (9) (L_1)

Figure 7: The object of interest, shown in (b), is matched against the scene images shown in the left columns by computing the distance shift D^s according to the models (9) (L_1) and (8) (L_2). Dark red indicate areas where the object model and scene image overlap exactly. The model (9) (L_1) is superior in case the object of interest is partially occluded or merged with the surroundings.

An illustration is given in Figure 7. In case the scene object is fully visible, the model gets shifted to the correct distance as seen in Figure 7(c). However, in case the scene object is partially occluded, the model instead gets shifted to the mean distance between the occlusion pattern and the scene object as seen in Figure 7(e), (h) and (k). A similarity measure that is more robust to occlusions can be constructed by instead minimizing the 1-norm between $D^M(i+u, j+v) + D^s$ and $D^P(i,j)$, weighted by the number of observed points

$$L_1(D^M, D^P; u, v) = \min_{D^s \in \mathbb{R}} \frac{\sum_{i,j \in I_{u,v}^{M,P}} \left| \left(D^M(i+u, j+v) + D^s \right) - D^P(i,j) \right|}{\sum_{i,j \in I_{u,v}^{M,P}} 1}. \tag{9}$$

Instead of being shifted to the mean, the minimizer D^s will shift the model distance function in such a way that it is aligned with the scene distance function for the maximum number of pixels. If at most 50% of the area of the object is occluded, it will ignore the occlusion pattern and align the model correctly as seen in Figure 7(f),(i) and (l). The statement will be made more mathematically concise and proved in the following theorem.

THEOREM 3.1. *Assume there exists a subset $S \subset I_{u,v}^{M,P}$, where $|S| > 0.5|I_{u,v}^{M,P}|$, such that $D^M(i+u, j+v) + D^s = D^P(i,j)$ for all $(i,j) \in I_{u,v}^{M,P}$ for some real number D^s. Then D^s is the unique minimizer of the L_1 norm defined in 9.*

Proof: The statement will be proved by contradiction. Assume that for some real non-zero number $a \in \mathbb{R}$, $D^s + a$ is a minimizer of (9). The objective function in (9) can be factored as $O(D^s) = E(D^s) \frac{1}{\sum_{i,j \in I_{u,v}^{M,P}} 1}$, where

$$E(D^s) = \sum_{i,j \in I_{u,v}^{M,P}} \left| \left(D^M(i+u, j+v) + D^s \right) - D^P(i,j) \right|$$

and $\frac{1}{\sum_{i,j \in I_{u,v}^{M,P}} 1}$ is the normalization factor.

Denote the set $\bar{S} = I_{u,v}^{M,P} \backslash S$. Then $E(D^s)$ can be decomposed as a sum over S and $\bar{S}$ as

$$E(D^s) = \sum_{i,j \in S} \left| \left(D^M(i+u, j+v) + D^s \right) - D^P(i,j) \right| + \sum_{i,j \in \bar{S}} \left| \left(D^M(i+u, j+v) + D^s \right) - D^P(i,j) \right| \tag{10}$$

$$= \sum_{i,j \in \bar{S}} \left| \left(D^M(i+u, j+v) + D^s \right) - D^P(i,j) \right| \tag{11}$$

where we have used that $\left| \left(D^M(i+u, j+v) + D^s \right) - D^P(i,j) \right| = 0$ for all $(i,j) \in S$. Consider now

$$E(D^s + a) = \sum_{i,j \in S} \left| \left(D^M(i+u, j+v) + D^s + a \right) - D^P(i,j) \right| \tag{12}$$

$$+ \sum_{i,j \in \bar{S}} \left| \left(D^M(i+u, j+v) + D^s + a \right) - D^P(i,j) \right| \tag{13}$$

$$= |a||S| + \sum_{i,j \in \bar{S}} \left| \left(D^M(i+u, j+v) + D^s + a \right) - D^P(i,j) \right| \tag{14}$$

$$\geq |a||S| - |a||\bar{S}| = E(D^s) + |a|(|S| - |\bar{S}|) > E(D^s) \tag{15}$$

where the last inequality follows by the assumption that $|S| > |\bar{S}|$. Since $E(D^s + a) > E(D^s)$ then $O(D^s + a) > O(D^s)$ and this contradicts the assumption that $D^s + a$ is a minimizer of (9). It follows that D^s must be the unique minimizer of (9). □

In this case, there is also a closed form solution to the minimization problem, which is the median over non-zero pixels

$$D^s = \text{median}_{(i,j) \in I_{u,v}^{M,P}} \left(D^P(i,j) - D^M(i+u, j+v) \right) \tag{16}$$

While this approach better handles occlusions, it has two limitations. First, if more than 50% of the projected area of the scene object is occluded, the shift D^s will be calculated in such a way that the model gets matched

with the occlusion pattern instead of the scene object. Secondly, the similarity measure $L_1(D^M, D^P; u, v)$ is calculated over both the visible and occluded parts of the scene object, which reduces the accuracy.

Motivated by the above results, a similarity measure will be constructed that handles more severe occlusions and excludes occluded parts of the scene points from calculation. Denote the function

$$H(i,j) = D^M(i+u, j+v) - D^P(i,j)$$

Hence, the model (9) can be formulated in terms of $H(i,j)$ as

$$L_1(D^M, D^P; u, v) = \min_{D^s \in \mathbb{R}} \frac{\sum_{(i,j)\in I_{u,v}^{M,P}} \left|H(i,j) + D^s\right|}{\sum_{(i,j)\in I_{u,v}^{M,P}} 1} \tag{17}$$

Define the one-dimensional function $\tilde{H}(k)$ for $k = 1, ..., K$ as the sort of all non-zero values in $H(i,j)$ in descending order. Here $K = \sum_{(i,j)\in I_{u,v}^{M,P}} 1$ is the total number of pixels where both D^M and D^P are non-zero. We use the algorithm merge-sort, which has a worst case complexity of $O(K * \log(K))$, to efficiently construct $\tilde{H}$. The equation (17) can now be written in terms of $\tilde{H}$ as

$$L_1(D^M, D^P; u, v) = \min_{D^s \in \mathbb{R}} \frac{1}{K} \sum_{k=1}^{K} \left|\tilde{H}(k) + D^s\right| \tag{18}$$

In case the scene object is partially occluded, the distance to the scene $D^P(i,j)$ will be lower over occluded parts than visible parts of the object. In consequence, the value of $H(i,j)$ will be higher at occluded parts than visible parts of the object. Since $\tilde{H}$ is a sort of the values of $H(i,j)$ in ascending order, it therefore follows that there must exist an integer $K_{\text{vis}} < K$ such that $\tilde{H}(i)$ for $i = 1, 2, ..., K_{\text{vis}}$ corresponds to the scene object and $\tilde{H}(i)$ for $i = K_{\text{vis}} + 1, K_2 + 2, ..., K$ corresponds to the occlusion pattern. We would like to calculate the sum in the nominator of (18) only over the scene object, i.e. as $\sum_{k=1}^{K_{\text{vis}}} \tilde{H}(k) + D^s$. Since K_{vis} is an unknown variable, we could make it part of the optimization problem as

$$\min_{K_{\text{vis}}} \min_{D^s \in \mathbb{R}} \frac{1}{K_{\text{vis}}} \sum_{k=1}^{K_{\text{vis}}} \left|\tilde{H}(k) + D^s\right| \tag{19}$$

An immediate problem with this approach is its bias towards small objects. In fact, by setting $K_{\text{vis}} = 1$ and assuming only one object pixel is visible, the minimum value can always be made zero. In order to avoid the bias towards small objects, it is necessary to incorporate a factor that reduces the detection confidence in case of few object pixels and increases the confidence gradually for larger number of visible pixels. The factor is formulated as a logarithmic function in the number of object pixels $\log_t(K_{\text{vis}})$ and will be discussed in more details in the next chapter on object recognition. By including this factor results in the new model

$$L_1^{\text{vis}}(D^M, D^P; u, v) = \min_{K_{\text{vis}}} \min_{D^s \in \mathbb{R}} \frac{\log_t(K_{\text{vis}})}{K_{\text{vis}}} \sum_{k=1}^{K_{\text{vis}}} \left|\tilde{H}(k) + D^s\right| \tag{20}$$

Although seemingly complex, the minimization problem (20) can be solved in different ways, where the second is particularly efficient.

Define the auxiliary function

$$E(\ell) = \min_{D^s \in \mathbb{R}} \frac{\log_t(\ell)}{\ell} \sum_{k=1}^{\ell} \left|\tilde{H}(k) + D^s\right| \tag{21}$$

The minimizer D^s is the median of $\tilde{H}(1), ..., \tilde{H}(\ell)$, and since $\tilde{H}$ is arranged in ascending order this is just

$$D^s = \begin{pmatrix} \tilde{H}((\ell+1)/2) & \text{if } \ell \text{ is odd} \\ \frac{\tilde{H}(\ell/2) + \tilde{H}(\ell/2+1)}{2} & \text{if } \ell \text{ is even} \end{pmatrix}. \tag{22}$$

The first algorithm obtains the minimizer of (20) by computing

Algorithm 1 for solving (20)

Set $E_{\min}^{\text{temp}} = \infty$, $E(0) = \infty$.
For $\ell = 1$ to K

- Compute $E(\ell)$ from (21) using the formula (16) for D^s
- **If** $E(\ell) < E(\ell - 1)$, **set** $E_{\min}^{\text{temp}} = E(\ell)$, $\ell_{\min}^{\text{temp}} = \ell$

Output $L_1^{\text{vis}}(D^M, D^P; u, v) = E_{\min}^{\text{temp}}$, $K_{\text{vis}} = \ell_{\min}^{\text{temp}}$.

Instead of computing $E(\ell)$ from scratch for every ℓ, the second algorithm computes $E(\ell)$ recursively in a very efficient manner. This is possible due to the observation that D^s can be fixed in advance in a certain way, while solving the simpler minimization problem

$$L_1^{\text{vis}}(D^M, D^P; u, v) = \min_{K_{\text{vis}}} \frac{\log_t(K_{\text{vis}})}{K_{\text{vis}}} \sum_{k=1}^{K_{\text{vis}}} \left| \tilde{H}(k) + D^s \right| \tag{23}$$

Note that if the model is perfectly aligned with the scene object along the azimuth and elevation angles, then the difference $D^M(i+u, j+v) - D^P(i, j)$ is equal for every visible pixel (i, j) on the scene object. Consequently, if n pixels on the object are visible then $\tilde{H}(1) = \tilde{H}(2) = \tilde{H}(n)$, and if n is greater than half of the number of model pixels then

$$\text{median}(\tilde{H}) = \text{percentile}_p(\tilde{H}) \tag{24}$$

for any percentile p less than or equal to 50. We have now established that if at least 50 percent of the projected area of the scene object is visible, then $D^s = \text{percentile}_p(\tilde{H})$ is a solution to the inner minimization problem (20) for every $K_{\text{vis}} >= \frac{2p}{100} * K$. Assume now that less than 50 percent of the projected area of the scene object is visible, i.e. that some fraction $C < 1/2$ of the number of object pixels K are visible. Then $D^s = \text{percentile}_p(\tilde{H})$ is still a solution to the inner minimization problem (23) for every $K_{\text{vis}} \leq 2 \cdot C \cdot K$, whereas for $K_{\text{vis}} > 2 \cdot C \cdot K$ it must hold that

$$\sum_{k=1}^{K_{\text{vis}}} \left| \tilde{H}(k) + \text{percentile}_p(\tilde{H}) \right| \geq \min_{D^s} \sum_{k=1}^{K_{\text{vis}}} \left| \tilde{H}(k) + D^s \right|.$$

The minimization problem (23) will therefore penalize choices of K_{vis} greater than the number of visible object pixels even more than the model (20). In practice, the model will never be aligned perfectly with the scene object due to inaccuracies caused by the 3D model and the discretization. This may cause some scene points behind the object to get encompassed within the domain of the model and thus for very small p (24) may be violated. In practice, we choose $p = 10$ and set $D^s = \text{percentile}_{10}(\tilde{H})$.

Recall the definition of $E(\ell)$ in (21). By fixing D^s as above, $E(\ell)$ can now be computed recursively by noting that

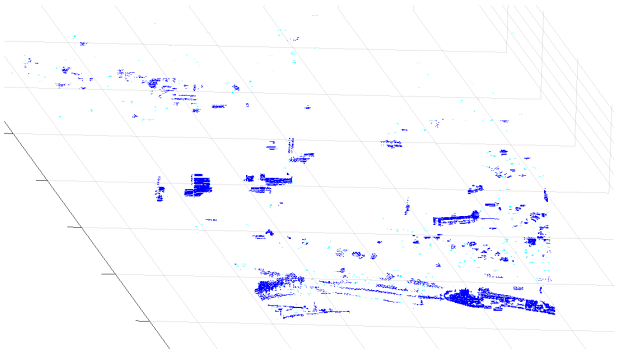
$$E(\ell + 1) = \frac{\log_t(\ell + 1)}{\ell + 1} \Big(\frac{\ell}{\log_t(\ell)} E(\ell) + \left| \tilde{H}(\ell + 1) + D^s \right| \Big). \tag{25}$$

Due to the convexity of the function $E(\ell)$, if for some ℓ, $E(\ell+1) > E(\ell)$, then $K_{\text{vis}} = \ell$ is the minimizer of (20).

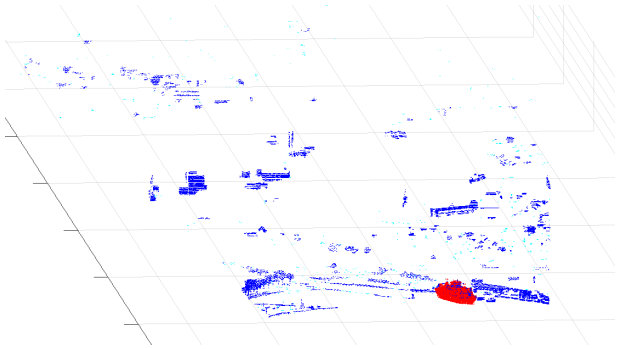
Algorithm 2 for solving (20)

Set $\ell = 0$, $E(0) = \infty$, D^s according to formula (16).
While $E(\ell + 1) < E(\ell)$ **and** $\ell < K$:

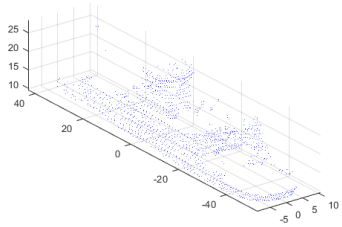
(a) Scene image



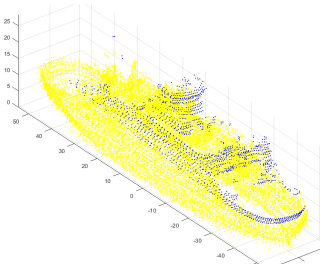
(b) Segmented point cloud



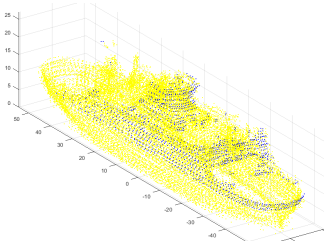
(c) Estimated pose of 3D model (red) inserted in segmented point cloud



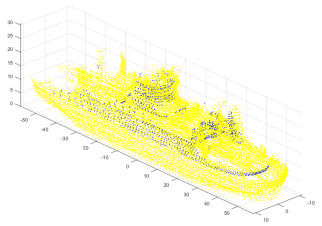
(d) Extracted object



(e) Extracted object aligned with 3D model (Imperfections are due to the discretization)



(f) Refined alignment (see next section) of object with 3D model of Bastø I. Mean point distance: 0.47 m



(g) Refined alignment (see next section) of object with 3D model of Bastø III, mean point distance: 0.85 m

Figure 8: Ladar scan of a ferry at quay. The ferry is extracted from the point cloud and identified as Bastø I. Details will follow in the next sections.

- Calculate $E(\ell+1)$ according to (25)
- $\ell \leftarrow \ell + 1$

Output $L_1^{\text{vis}}(D^M, D^P; u, v) = E(\ell-1)$, $K_{\text{vis}} = \ell - 1$

Recall that an estimate of the distance to the scene object within 200 meters was assumed to be known in advance, and the distance function D^M to the 3D model was created based on this estimate. We now describe a procedure to estimate the approximate distance to the scene object automatically. Let D_r^M, $r = 1, ..., R$ be distance functions to the 3D model of the object, created from ranges between some lower and upper ranges, each of them about 10% farther away than the previous. Define the indicator functions $\mathbb{1}_{D_r^M(i,j)}$ which take the value 1 when $D_r^M(i,j)$ is positive and 0 otherwise

$$\mathbb{1}_{D_r^M}(i,j) = \begin{pmatrix} 1 & \text{if} & D_r^M(i,j) > 0 \\ 0 & \text{otherwise} & \end{pmatrix}.$$

By utilizing the hat shaped function $G(i,j) = \sum_{r=1}^{R} \mathbb{1}_{D_r^M}(i,j)$, the distance can be estimated as the weighted mean

$$r^* = \frac{\sum_{i,j} G(i+u, j+v) D^P(i,j)}{\sum_{i,j} G(i+u, j+v)}$$

Instead of creating a new distance function, using the estimated range r^*, the one of the precomputed distance function $D_r^M(i,j)$ with range closest to r^* is used for efficiency.

We want to find the translation (u,v) and orientation (ψ) of the distance function of the 3D model that gives the best similarity with the scene image D^P. For each (u,v) we let $D_r^\psi(i+u, j+v)$ denote distance function to the 3D model using the estimated range, rounded to the nearest precomputed range r as described above. The set of possible translations are restricted to the points where there is an overlap between the 3D model and the scene image, i.e. the set

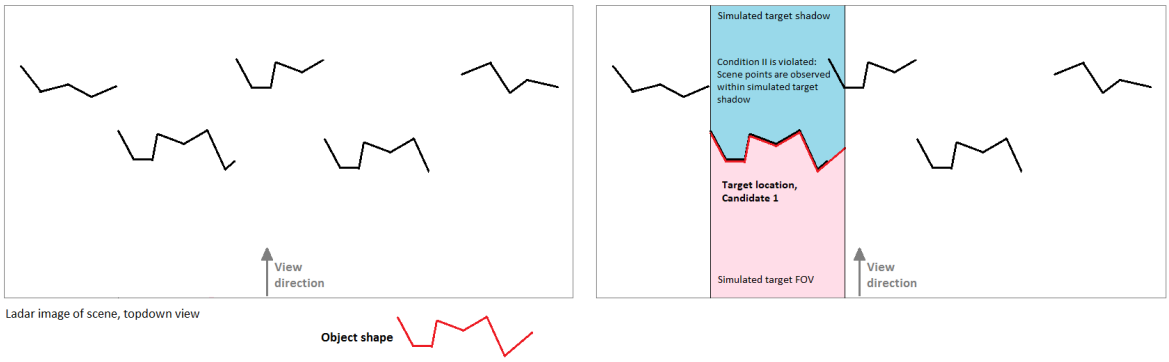
$$J^{\psi,P} = \Big\{ \text{all } (u,v) \in \mathbb{N}^2 \text{ such that there exists } (i,j) \in \mathbb{N}^2 \text{ where} D^M(i+u, j+v) \neq 0 \text{ and} D^P(i) \neq 0 \Big\}$$

For every such translation $(u,v) \in J^{\psi,P}$ and every discrete view angle ψ Algorithm 2 is used to compute $L_1^{\text{vis}}(D^{M^\psi}, D^P; u, v)$ defined in (23).

Locations (u,v) where $L_1^{\text{vis}}(D^{M^\psi}, D^P; u, v)$ is lower than a given threshold indicate potential locations of the scene object. The 3D model can now be inserted in the point cloud by converting the spherical coordinates $(\phi_{\min} + u\delta_\phi, \theta_{\min} + v\delta_\theta, d)$, where d is the estimated distance to the scene object, into Euclidean coordinates, using the estimated orientation ψ. Points within the relevant class of the segmented point cloud that are within a certain distance to the 3D model are extracted and regarded as a single object to be further analyzed in the next step.

4. IDENTIFICATION OF EXTRACTED OBJECTS

The final "recognition"/"identification" step proceeds by calculating a recognition confidence measure for relevant objects extracted by the previous step by a more computationally demanding algorithm. The object extraction step gives an approximate pose of the object 3D model in the scene, but suffers from reduced precision due to discretization of the point cloud distance function and the orientation of the 3D model. The recognition step refines the alignment between the extracted object and 3D model, by using the estimated pose of the 3D model as initial condition. See Figure 8 (c) for an illustration. The algorithm also utilizes other criteria, such as the consistency between the shadows of the observed point cloud and of the 3D model, in order to create a recognition confidence measure for various objects. Extended mathematical details were given in our previous paper.[1] Here we provide a brief summary of those criteria:



(a) 2D point cloud and shape of the object of interest

(b) Criterion II is violated

(c) Less likely to be the object by criterion III

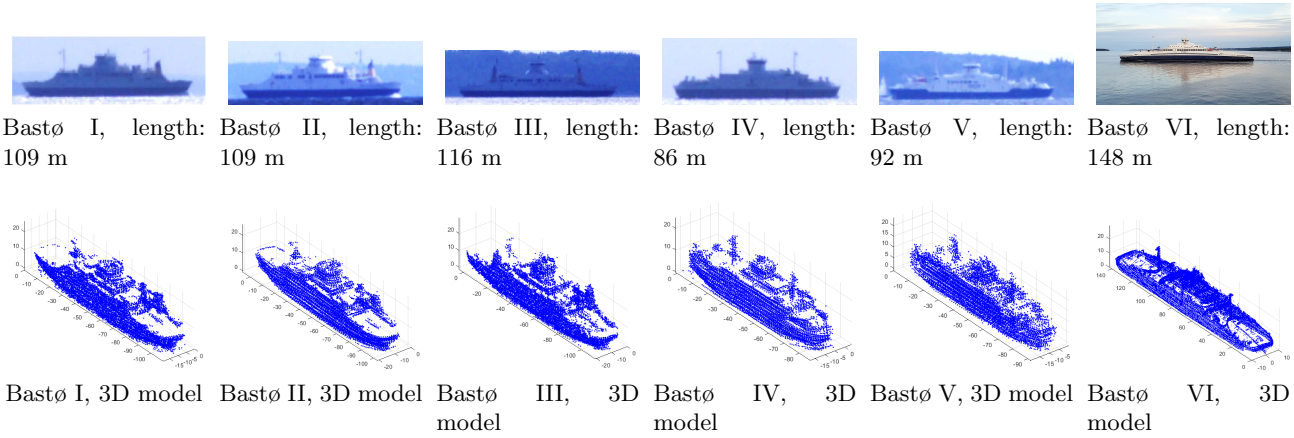
(d) All criteria I-III are fulfilled

Figure 9: Illustration of recognition criterion I-IV. Shown in (a) is a dense two-dimensional point cloud obtained by scanning the scene by a ladar in the indicated view direction. Imagine for example that the ladar has scanned a single line across the scene. We wish to find and recognize the red object under (a) within the point cloud. (b)-(d) analyzes various scene objects in terms of the recognition criteria.

(I) How closely can the observed object points be aligned with a 3D model of the object of interest? To answer this question while also accounting for objects that may only be partially visible, the algorithm rotates and translates the observed object in such a way that the average distance between each observed point and its closest corresponding point on the 3D model becomes as small as possible. The algorithm processes the observed 3D points directly, and measures their distance to the nearest triangle on the 3D model. Results are depicted in subplots (d)-(e) of Figures 11, 12 and subplots (c) and (d) of Figure 4, where observed points are marked in blue and model points are marked in yellow. The mean point distance is indicated by C_1 in the rest of this article, and is shown below each figure.

(II) How well does the 3D model match with the point cloud shadows? Assuming the object of interest is constructed of non-transparent material, points are not expected to appear behind it along the line of sight. This criterion is useful for ruling out other scene objects, when searching for an object that may be partially occluded, or partially non-reflective. In Figures 11 and 12 (f)-(g), dark and red indicate the total projected area of the simulated shadow of the 3D model from the view point of the ladar. Red indicates points that are observed within the shadow, violating the non-transparency assumption. The fraction of the total expected shadow area containing such red violating points is indicated by C_2 in the rest of this article, and is shown below each figure.

(III) How well does the distribution of detected points over the projected area of the 3D model fit with expectations? Occluded parts of the area are not expected to reflect pulses. Over visible parts of the projected area, the likelihood of absence of detected points may be estimated rather well from the incidence angle of the laser beam with the surface of the 3D model, the distance and the strength of other nearby return



Bastø I, length: 109 m | Bastø II, length: 109 m | Bastø III, length: 116 m | Bastø IV, length: 86 m | Bastø V, length: 92 m | Bastø VI, length: 148 m

Bastø I, 3D model | Bastø II, 3D model | Bastø III, 3D model | Bastø IV, 3D model | Bastø V, 3D model | Bastø VI, 3D model

Figure 10: The six Bastø ferries and their 3D models obtained by mixing the point clouds acquired by ladar scans from different view angles. Bastø I and II are sister ships with the same shape and will be regarded as the same ship.

pulses. Some knowledge about reflectance properties of the object surface would of course make it possible to tune this criterion. This criterion is optional and will not be utilized in the experiments of this paper.

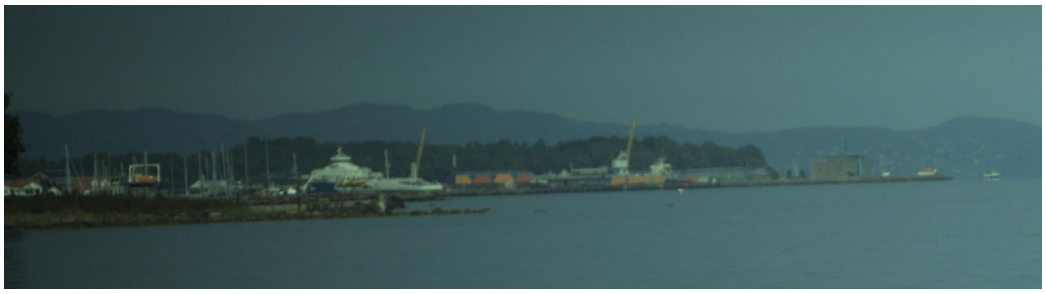
(IV) How many object points are observed? The confidence measure is weighted by a function that decreases (logarithmically) as the number of observed points decreases. This factor counteracts the fact that objects with few points can more easily match with the 3D model by coincidence.

An illustration of criterion I-IV is given in figure 9. Earlier work such as ref.[3,16,17] have proposed to recognize objects in point clouds based on how well they match with some 3D model. Utilization of the shadows in ladar point clouds has been a surprisingly scarce topic in previous literature as far as we are aware. Ref.[18] analyzed shadows in ladar point cloud from a physical perspective and demonstrated how they can aid in manual object detection, classification and recognition. In ref.[19,20] the point cloud shadows were used for automatic object detection.

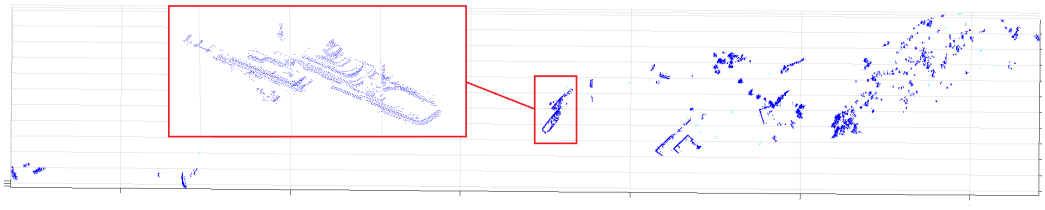
5. EXPERIMENTS

This section presents some sample experimental results on some datasets recorded of the Bastø ferries that travel across the Oslo fjord in Norway. There are a total of 4 different ferries in operation, and 2 additional ferries that have recently been phased out. All the 6 ferries have rather similar shapes and sizes and are shown in Figure 10, together with 3D models that have been constructed by merging several ladar scans from different view angles.

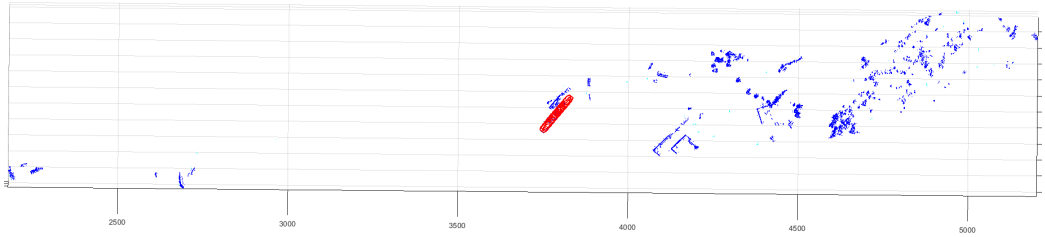
Given a ladar scan of the scene, we would like to test the ability of the algorithm to estimate if and where a Bastø ferry is located, and identify the ferry among all the Bastø ferries. Results were shown in our previous paper[1] in an open sea environment. Here we focus on challenging cases where the ferries are at quay or observed straight from the front. The point clouds are only segmented into two classes: solid objects (dark blue) and noise (light blue). This will challenge the object extraction method the most. Figure 11 shows a ladar scan of the ferry Bastø VI at quay while also being partially occluded by the nearby building. As one can see in (b), the ferry, quay and building appear as one large object in the point cloud. The object extraction step proposed in this paper is able to extract the 3D points on the ferry from the surrounding objects. In subfigure (d) the alignment of the observed object with the 3D model of Bastø VI according to criterion I is shown. The mean point distance (C_1) between the observed object and 3D model is 0.527. For comparison, the second best match is with the ferry Bastø III and is shown in subfigure (d), resulting in a significantly worse mean point distance of 1.58 m. Subfigures (e) and (f) show the inconsistencies between the point cloud shadows and the simulated shadow of each of the 3D models when inserted in the scene. Bastø VI gives only 1.9 percent inconsistences,



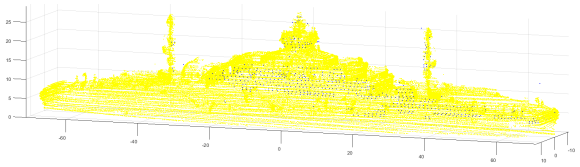
(a) Visual light image



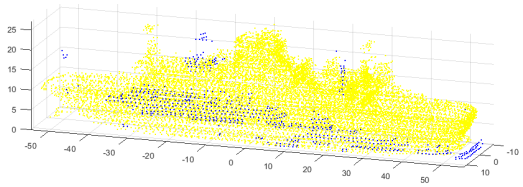
(b) Segmented point cloud (top down view)



(c) Estimated pose of 3D model (red) inserted in segmented point cloud



(d) **Identified ship: Bastø VI**, mean point distance : 0.527 m



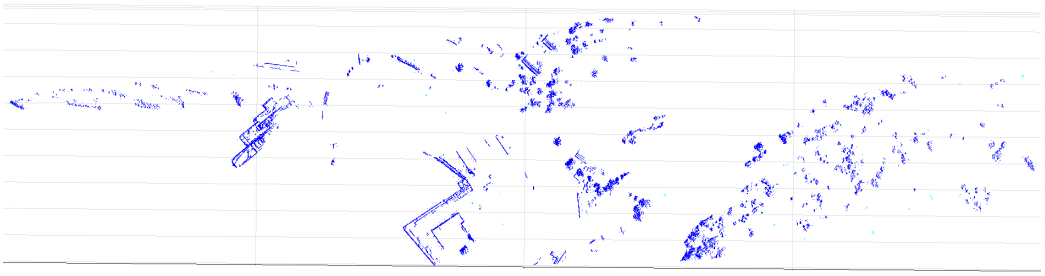
(e) 2nd best match: Bastø II, mean point distance : 1.584 m

(f) Bastø VI, shadow inconsistency : 1.88 percent

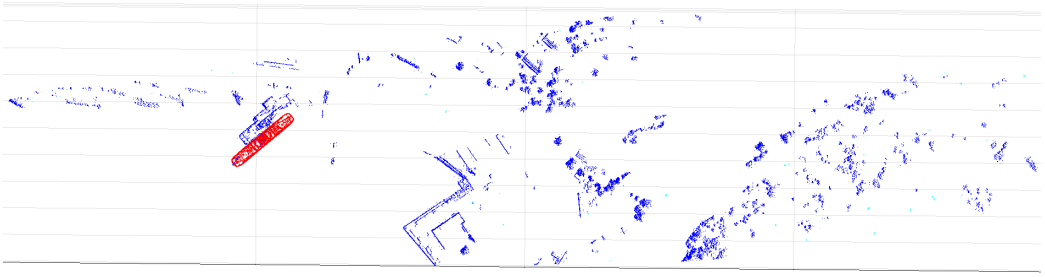
(g) Bastø II, shadow inconsistency : 35.3 percent

Figure 11: Extraction and identification of a Bastø ferry at quay.

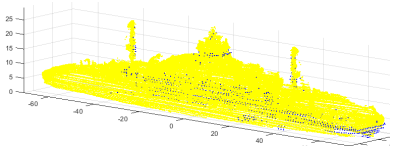
(a) Visual light image



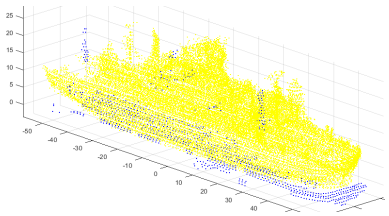
(b) Segmented point cloud (top down view)



(c) Estimated pose of 3D model (red) inserted in segmented point cloud



(d) **Identified ship Bastø VI**, mean point distance: 0.627 m



(e) Bastø III, mean point distance: 1.913 m

(f) Bastø VI, shadow consistency: 2.30 percent

(g) Bastø II, shadow inconsistency: 43.4 percent

Figure 12: Extraction and identification of a Bastø ferry at quay.

which is significantly less than Bastø III. In spite of the relatively fewer observed points due to the occlusion pattern, the object gets identified as Bastø VI with high confidence according to the recognition confidence measure. A similar result is shown in Figure 12 from another view point, in which case the ferry is identified with high confidence. The results are consistent with our previous paper,[1] which indicated that a mean point distance below 0.6 and shadow inconsistency below 4 percent are excellent for objects of this size.

Further results were shown in Figure 4, where Bastø VI is at quay and Bastø I is scattered into two pieces due to the view direction straight from the front, and in Figure 8 where Bastø I is observed at the quay. The ferries get correctly identified in all cases.

6. CONCLUSIONS

This paper proposed an algorithm for locating and identifying objects in points clouds acquired by a ladar in clustered or scattered environments. Our previous work approached object detection and recognition by first segmenting the point cloud into multiple classes of similar objects, before a more sophisticated and computationally demanding algorithm attempted to recognize/identify individual objects within the relevant class. This approach has limitations in case the object of interest is placed within a cluster of objects from the same class, or in case the object of interest is divided into several disjoint parts due, for instance, to slant view angles. An intermediate step between segmentation and recognition was proposed that extracted objects from the relevant class based on similarity between their distance function and the distance function of a reference shape for different view angles. The similarity measure accounted for occlusions and partial visibility naturally, and could be expressed analytically in the distance coordinate for azimuth and elevation angles within the field of view (FOV). This reduced the dimensions for which to search from three to two, which was a crucial property for maintaining a high computational efficiency of the overall algorithm. Experimental results showed that partially occluded ships at quay and ships observed straight from the front could be identified with a high confidence.

REFERENCES

[1] Bae, E., "Automatic scene understanding and object identification in point clouds," *Proc. SPIE* **11160**, Electro–Optical Remote Sensing XIII, 111600M (2019).

[2] Palm, H. C., Haavardsholm., T., Ajer, H., and Jensen, C. V., "Extraction and classification of vehicles in ladar imagery," *Proc. SPIE* **8731**, Laser Radar Technology and Applications XVIII, 873102 (2013).

[3] Armbruster, W. and Hammer, M., "Maritime target identification in flash-ladar imagery," *Proc. SPIE* **8391**, Automatic Target Recognition XXII, 83910C (2012).

[4] Jiang, X. and Bunke, H., "Fast segmentation of range images into planar regions by scan line grouping," *Machine Vision and Applications* **7**, 115–122 (1994).

[5] Felip, R., Ferrandans, S., Diaz-Caro, J., and Binefa, X., "Target detection in ladar data using robust statistics," *Proc. SPIE* **5988**, Electro–Optical Remote Sensing, 59880J (2005).

[6] Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A. Y., "Discriminative learning of markov random fields for segmentation of 3d scan data," in [*IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA*], 169–176 (2005).

[7] Triebel, R., Kersting, K., and Burgard, W., "Robust 3d scan point classification using associative markov networks," in [*Proc. of the International Conference on Robotics and Automation(ICRA)*], 2603–2608 (2006).

[8] Golovinskiy, A., Kim, V. G., and Funkhouser, T., "Shape-based recognition of 3D point clouds in urban environments," in [*International Conference on Computer Vision (ICCV)*], 2154–2161 (2009).

[9] Z. Wu, S. S., A. Khosla, F. Y., Zhang, L., Tang, X., and Xiao., J., "3d shapenets: A deep representation for volumetric shapes," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 1912–1920 (2015).

[10] Maturana, D. and Scherer, S., "Voxnet : A 3d convolutional neural network for real-time object recognition," in [*2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*], 922–928 (2015).

[11] Qi, C. R., H. Su, M. N., Dai, A., Yan, M., and Guibas, L., "Volumetric and multi-view cnns for object classification on 3d data," in [*IEEE Proc. Computer Vision and Pattern Recognition (CVPR)*], 5648–5656 (2016).

[12] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. G., "Multi-view convolutional neural networks for 3d shape recognition," in [*IEEE International Conference on Computer Vision (ICCV)*], 945 – 953 (2015).
[13] Bae, E. and Merkurjev, E., "Convex variational methods on graphs for multiclass segmentation of high-dimensional data and point clouds," *Journal of Mathematical Imaging and Vision* **58**(3), 468–493 (2017).
[14] Bentley, J. L., "Multidimensional binary search trees used for associative searching," *Communications of the ACM* **18**, 509–517 (1975).
[15] Indyk, P., "Chapter 39 : Nearest neighbours in high-dimensional spaces," in [*Handbook of Discrete and Computational Geometry (2nd ed.). CRC Press.*], 1–16 (2004).
[16] Chen, Y. and Medioni, G., "Object modeling by registration of multiple range images," in [*Proc. IEEE International Conference on Robotics and Automation*], 2724–2729 (1991).
[17] Besl, P. J. and McKay, H. D., "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(2), 239–256 (1992).
[18] Steinvall, O., Sjökvist, L., Jonsson, P., and Edström, S., "Shadows in laser imaging and mapping," *Proc. SPIE* **10796**, Electro–Optical Remote Sensing XII, 1079609 (2018).
[19] Kuntimad, G. and Delashmit, W., "Target detection in ladar range images using shadow analysis," *Proc. SPIE* **6214**, Laser Radar Technology and Applications XI, 621405 (2006).
[20] Grönwall, C. A., Tolt, G., Chevalier, T., and Larsson, H., "Spatial filtering for detection of partly occluded targets," *Optical Engineering* **50**, 047201 (2011).