Covariance Based Point Cloud Descriptors for Object Detection and Recognition

Duc Fehr^a, William J. Beksi^{a,*}, Dimitris Zermas^a, Nikolaos Papanikolopoulos^a

^a *Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 55455 USA*

Abstract

Processing 3D point cloud data is of primary interest in many areas of computer vision, including object grasping, robot navigation, and object recognition. The introduction of affordable RGB-D sensors has created a great interest in the computer vision community towards developing efficient algorithms for point cloud processing. Previously, capturing a point cloud required expensive specialized sensors such as lasers or dedicated range imaging devices; now, range data is readily available from low-cost sensors that provide easily extractable point clouds from a depth map. From here, an interesting challenge is to find different objects in the point cloud. Various descriptors have been introduced to match features in a point cloud. Cheap sensors are not necessarily designed to produce precise measurements, which means that the data is not as accurate as a point cloud provided from a laser or a dedicated range finder. Although some feature descriptors have been shown to be successful in recognizing objects from point clouds, there still exists opportunities for improvement. The aim of this article is to introduce techniques from other fields, such as image processing, into 3D point cloud processing in order to improve rendering, classification, and recognition. Covariances have proven to be a success not only in image processing, but in other domains as well. This work develops the application of covariances in conjunction with 3D point cloud data.

Keywords: RGB-D data, colored point clouds, classification, object recognition

1. Introduction

Three dimensional (3D) point clouds provide important cues to analyze objects or environments. They are, for instance, heavily used in topographical mapping, where an airplane or a satellite passes over an area and takes several snapshots with a laser range finder. Researchers like Schwalbe et al. [1] used such laser range finders to build a height map of a scanned area. In the area of mobile robotics, Weingarten et al. [2] have developed a sensor that generates point clouds usable by a robot to sense its environment, detect obstacles, and navigate without hitting them. Ip and Gupta [3] use dense point clouds of models for computer aided design (CAD) purposes. Allen et al. [4] use point clouds to accurately reconstruct historic monuments.

One of the most important objectives for a mobile robot is to intelligently process information from the surrounding environment. In order to perform its tasks, the robot needs to be able to perceive its environment. For example, when navigating a robot has to recognize certain features that can be used as landmarks. During the DARPA Grand Challenge and Urban Challenge,

laser range finders have been used extensively on-board different autonomous vehicles to build a 3D map of the immediate environment. When trying to grasp an object, a robot first needs to find the object in the scene. To execute this task, 3D object detection and classification have to be performed.

There are also several uses of 3D point clouds in the medical domain. 3D scans of patients can be taken for the purpose of medical diagnosis. In orthotics, CAD models can be designed from the patients' scans in order to develop and manufacture adapted orthoses (efficiently). In dentistry, 3D models of teeth can help to plan and design dental repairs.

Common to almost all of these applications is the basic task of identifying and analyzing objects in a 3D point cloud. Object recognition using only depth data is a challenging task as it lacks many other informational cues such as color, intensity, and texture. The problem is exacerbated when multiple objects are cluttered in the point cloud as finding the different object boundaries becomes hard. Furthermore, sensor noise, surface reflectivity of the sensor beams, limited range of the 3D sensors, and the divergence of the sensor beams as the depth of the point increases, make object recognition in

*Corresponding author email: beksi@cs.umn.edu (W.J. Beksi).

this setting extremely difficult.

1.1. Motivation

The release of the Microsoft Kinect [5] has had a revolutionary effect on the research community as it provides not only a regular RGB camera, but also an affordable range camera accessible to everyone. This device constitutes one of the first so-called RGB-D sensors that provides a depth image as well as a regular RGB picture. Previously, point cloud capture involved expensive specialized sensors, such as lasers or dedicated range imaging equipment. Today, range data is readily available from cheap sensors supplying point clouds that can easily be extracted from a depth map. Many different works exist for applications ranging from controlling a robot with human motion to actually using the Kinect sensor on a mobile robot as the environment measuring device. In this work, an RGB-D sensor is used to capture a depth map from which a point cloud can be computed for analysis. Since this sensor has not been designed for very accurate measurements compared to the accuracy offered by a laser range finder, it is prone to some amount of noise. However, since RGB-D sensors provide a depth map in addition to a regular RGB image, the colored picture can be used to “color” the point cloud. This “colored” point cloud contains even more information at each point and can thus be used to enhance the object classification performance.

The problem of object recognition in point clouds is much harder than just rendering the collected data. Several works have successfully dealt with object recognition and point cloud classification. In particular, the method of using *spin images* introduced by Johnson and Hebert [6], work well in describing local surfaces. There have been several extensions to the initial idea of spin images. A significant limitation observed with state of the art techniques is that their performance heavily depends on the configuration settings, e.g., the size of the mesh used for analysis or the histogram bin size. Another difficulty is their high dimensionality, making nearest neighbor matching on large 3D point cloud databases computationally prohibitive. In addition, existing literature on spin images focuses on evaluating their performance on noiseless synthetic data or on point clouds generated from inherently less noisy laser data.

1.2. Related Work

Regarding the use of laser range finders and scanners, a lot of work in the area of reconstruction has been done. Carr et al. [7] discuss the use of Radial Basis Functions (RBF) in order to reduce the number of

points in a point cloud of an object. Minor flaws in the measurement point cloud due to self occlusions can be corrected. Similar work has been done by Bajaj et al. [8] where the authors present algorithms to reconstruct objects from a point cloud. These works emphasize the fact that the reconstruction algorithms are efficient. Using fewer points for reconstruction would speed up the process since less data needs to be processed.

Carmichael et al. [9] extends this work to take into account the nonuniform distribution of the sensed points at close and long ranges. This approach seeks to answer the problem of selecting the correct scale for spin images. Assfalg et al. [10] use spin image signatures in which the spin images are divided into sectors and the signatures correspond to the histogram of points according to these sectors. Li et al. [11] combine the spin image descriptors with a dictionary learning approach to quickly compare parts of a CAD model when information is only partially available.

In 2004, Tangelder and Velcamp [12] published a comprehensive survey on different point cloud descriptors. In more recent developments, Rusu et al. [13] use a histogram based approach to find matching points to facilitate the alignment and merging of point clouds. Similar to Assfalg et al. [10], who construct signatures on top of spin images, Tombari et al. [14] show that building signatures on top of histograms provides good results for feature detection.

The common denominator of these methods is that they have been developed for “simple” point clouds. Now that access to “colored” point clouds has been drastically simplified with the introduction of cheap RGB-D sensors, a lot of work has been done in this area. Lai et al. [15] introduce an RGB-D image database of objects that have been pre-segmented and on which different algorithms can be tested. The present work will use this database for the experiments. Alongside the introduction of the benchmark, the authors introduce a descriptor that is able to classify and detect objects from RGB-D data. The same authors establish a second approach to classification [16] using sparsity techniques in order to classify objects. Their work also shows that shape, combined with appearance information, provides better classification results than classifying for shape or appearance separately.

Bo et al. [17] introduce kernel descriptors for object recognition. In further work [18], they compute and evaluate descriptors at different patch sizes and thus build a hierarchical model of depth kernels. With a similar focus, Blum et al. [19] use dictionary learning in order to find relevant features for classification. In their work, Bo et al. [20] merge the previous ideas of hier-

archical models with dictionary learning. Similar work has been performed by Tang et al. [21] on a dataset provided by Willow Garage for the 2011 ICRA Solutions in Perception Instance Recognition Challenge.

The applicability of a covariance matrix for describing object appearances was introduced by Tuzel et al. [22] for the task of detection and classification. In their work, the descriptors are essentially the covariance matrices of feature vectors from the data.

Tuzel et al. [22] use a nine dimensional feature vector containing RGB information of the pixel and the gradients at the pixel. The computed descriptor is then a 9x9 matrix. Porikli et al. [23] continue to build on the covariance descriptor idea to track objects in videos from static and moving cameras. The same nine features are used in the static setup.

While the discriminative power of a histogram based descriptor, similar to spin images, is dependent on the bin size of its parameters, the representational power of covariance matrices arises from their inherent structure that captures the correlation between the dimensions of the feature vectors. Thus, on the one hand, covariances help to eliminate the parameters that need to be tuned as in spin images. However, on the other hand, their inherent structure poses significant computational difficulties; these matrices do not adhere to the Euclidean geometry, but span a Riemannian space with an associated Riemannian distance metric. One such metric is the geodesic distance defined by Förstner and Moonen [24]. Another metric has been developed by Arsigny et al. [25] to speed up the distance measurement between covariance matrices.

Covariance descriptors have found successful applications in many other areas of computer vision such as emotion classification [26] and action recognition [27]. Pang et al. [28] use the responses to Gabor filters as features for face recognition. Even areas such as acoustics, have seen the introduction of covariances [29, 30].

2. Covariances

As described in the previous section, covariances have been introduced as descriptors in image processing by Tuzel et al. [22] for detection and classification tasks.

2.1. Concept

In order to build the covariance descriptor, different features need to be extracted. Let I be an image and let ϕ be a function that maps I to a p channeled image $\mathcal{F}$ containing a p dimensional feature vector $F_p \in \mathbb{R}^p$ at each pixel:

$$\begin{aligned} \phi : \quad I &\rightarrow \mathcal{F} \\ I(x, y) &\mapsto \mathcal{F}(x, y). \end{aligned} \quad (1)$$

I is an image with dimensions $w \times h \times c$ where $c = 1$ for a gray scale image and $c = 3$ for a colored image. $\mathcal{F}$ is the feature image having the same width and height as I , thus having dimension $w \times h \times p$, where p is the number of considered features. ϕ can be any function such as intensity, gradients, Fourier transforms or a combination of these.

Let $z_i = \mathcal{F}(x_i, y_i)$, $z_i \in \mathbb{R}^p$, for $i = 1, 2, \dots, N$, be the feature vectors from a region of interest (ROI) of the image, then the covariance descriptor of this region $C_{ROI} \in \mathcal{S}_{++}^p$ is defined as:

$$C_{ROI} = \frac{1}{N-1} \sum_{i=1}^N (z_i - \mu_z)(z_i - \mu_z)^T, \quad (2)$$

where μ_z is the mean feature vector, and $\mathcal{S}_{++}^p$ is the space of $p \times p$ symmetric positive definite (SPD) matrices.

2.2. Distance

One of the difficulties with covariance matrices lies in determining how “close” they are to each other since positive definite matrices do not adhere to Euclidean geometry. Intuitively, this simply means that the closest distance between two covariance matrices is not a straight line.

One metric that can be used, the geodesic distance, has been introduced by Förstner and Moonen [24] and is defined as follows:

$$d_{GD}(X, Y) = \sqrt{\text{trace}(\log^2(X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}))}, \quad (3)$$

where $X, Y \in \mathcal{S}_{++}^p$ and $\log(\cdot)$ is the matrix logarithm operator.

This metric is affine invariant and has been the most widely used Riemannian metric over covariance descriptors. However, it is computationally expensive to use this metric and therefore recent research proposes the use of the log Euclidean Riemannian metric [25] which has the following form:

$$d_{LE}(X, Y) = \|\log(X) - \log(Y)\|_F, \quad (4)$$

where $\log(\cdot)$ is the matrix logarithm operator and $\|\cdot\|_F$ is the Frobenius norm. Since this metric decouples the two matrices involved in the distance computation as in Eq. (3), and since the matrix logarithms can be computed offline, this metric is very fast and will be used throughout this paper.

As described in the previous section, covariances are computed over predefined regions of interest. This provides a degree of flexibility in the type of covariance descriptor that can be defined. There are two main approaches that can be used:

1. Local approach: At each point in the point cloud all the points in a neighborhood are considered to be in the region of interest. A disadvantage is the inherent increase in data. For each point, not only does the color and position need to be saved, but also the covariance.
2. Global approach: All the points in an object are considered to be part of the neighborhood. One covariance is computed per object. The entire point cloud corresponding to one object is represented as a single matrix.

3. Local Descriptor

For this phase of the work, pointwise (local) descriptors are considered. Each point in the point cloud is described with a descriptor, augmenting the cloud with covariances, and hence making this approach memory intensive.

3.1. Dataset

The Kinect [5] sensor is used to capture point clouds from the surrounding environment. The experimental setup is designed towards creating a real world 3D object recognition scenario with arbitrary objects found around the laboratory. Thus, data from eight different classes of objects was collected. The classes are the following:

• Bottle (Bot)	• Person (Per)
• Box (Box)	• Robot (Rob)
• Chair (Cha)	• Soccer ball (Soc)
• Loudspeaker (Lou)	• Tennis ball (Ten)

Sample images and the respective range data for each class are illustrated in Fig. 1.

3.2. Features

A total of eight features are explored. Let p be the point at which the covariance is computed and let p' be a point in the neighborhood of p whose features are used in the computation of the covariance. P is the tangent plane to the surface at p , and is characterized by the

Figure 2: The different features that are used in the covariance descriptor are illustrated. The features are computed similarly to the spin images by measuring the distances in the plane defined by the normal n at point p and vector pp' . α , β , and θ are defined the same way as previously. ρ is the norm of vector pp' , while ψ is the angle between the normals. P is the tangent plane at p .

normal n , which is estimated by performing an eigendecomposition of the covariance matrix created from the points in the neighborhood of p . The definitions of the four initial features are the following: α is the l_2 norm of the projection of pp' onto the plane P , β is the l_2 norm of the projection of pp' onto n , θ defines the angle between vector pp' and the normal n , and ρ is the l_2 norm of vector pp' . These four features are complemented by ψ , the angle between the normals n and n' at their respective points p and p' . Therefore, a total of eight features are available. The computation of these features is given next:

$$\alpha = \|pp' - (pp' \cdot n)n\|_2 \quad (5)$$

$$\beta = \|(pp' \cdot n)n\|_2 \quad (6)$$

$$\theta = \arccos\left(\frac{pp' \cdot n}{\|pp'\|_2}\right) \quad (7)$$

$$\rho = \|pp'\|_2 \quad (8)$$

$$\psi = \arccos(n \cdot n') \quad (9)$$

$$n' = [n'_x, n'_y, n'_z]^T. \quad (10)$$

3.3. Results

500 random sample points out of each class are taken and queried against an increasing amount of randomly chosen descriptors from the classes. In the case that there are not enough descriptors available in the class, the entire set is used. A k-nearest neighbor approach was used for classification, with the log Euclidean (LE) distance (Eq. (4)) to compare the covariances. The results for four different feature vectors, defined in Eqs. (11) – (14), are given in Fig. 3. These vectors differ by the number of features used. Vectors F_i , $i \in \{3, 4, 5, 8\}$ show which features were used to

Figure 1: Different objects in which we compute pointwise covariance descriptors. The top row contains the RGB image of each object while the bottom row is the corresponding depth image.

Figure 3: Pointwise classification for different feature vectors. The black solid line corresponds to the results of the spin image. The dashed lines represent the results for the covariance descriptors using the log Euclidean distance (LE).

compute the covariances. For comparison, the spin image performance is also included.

$$F_3 = [n'_x, n'_y, n'_z]^T \quad (11)$$

$$F_4 = [\alpha, \beta, \theta, \rho]^T \quad (12)$$

$$F_5 = [\alpha, \beta, \theta, \rho, \psi]^T \quad (13)$$

$$F_8 = [\alpha, \beta, \theta, \rho, \psi, n'_x, n'_y, n'_z]^T. \quad (14)$$

The trend in Fig. 3 is for the most part expected. The more features used, the more discriminative the feature descriptors become. An interesting observation is that the results for the covariance using only the normals give better results when compared to using all eight features. There is information contained in the normals that spin images do not take advantage of, which prompted the use of the normals' coordinates as features. The angle between the normals at point p and point p' also contains information along those lines, therefore they

were used in the feature vector to increase the discriminative power of the descriptor.

Nevertheless, this approach presents the following disadvantages. First, since a covariance is computed at each point, the amount of data that needs to be stored is drastically increased. Second, two objects being locally similar is a potential cause of problems. Therefore, a “global” descriptor is introduced in which the covariance over a whole object is computed by compressing the complete point cloud into a single matrix.

4. Global Descriptor

In order to make comparisons faster and more stable, a “global” approach is tested. Instead of having a covariance descriptor computed for each point of an object, each object is characterized by a single descriptor. An additional benefit of this approach is the memory savings achieved through the compression of the data.

4.1. Dataset

Lai et al. [15] introduce an RGB-D image dataset of objects that have been pre-segmented and on which different algorithms can be tested (Fig. 4). The present work uses this database for the experiments.

4.2. Features

The RGB-D dataset provides its r, g, b color channel values at each point (“C” features) as well as its x, y, z coordinates (“P” features). This allows access to six different features, Figs. 5(b) and 5(c).

This number is increased by computing the normal (n_x, n_y, n_z) at each point to get three additional features (“N” features), Fig. 5(d). The image features (“IMI” features) correspond to the RGB image intensity, Fig 5(e), along with the outputs of the Sobel operator [31], Figs. 5(f) and 5(g). I_x and I_y correspond

Figure 4: A subset of the different objects contained in the database introduced by Lai et al. [15].

to the output when the operator is applied to the patch along x and y respectively. For I_{xx}, I_{yy}, I_{xy} , the Sobel operator is applied a second time on the patch. Figs. 5(i), 5(j), and 5(k) display the output of these operations. $Mag = \sqrt{I_x^2 + I_y^2}$ corresponds to the magnitude of the gradient of the image patch, a sample of which is shown in Fig. 5(h).

Once the normals are computed at each point, the curvature can be estimated, yielding two values along the main curvature axes (c_1, c_2) (“ k ” features). Finally, the product of these curvatures provides a “total” curvature, $C = c_1 \cdot c_2$, at the point (“ K ” feature), Fig. 5(l). The use of the local curvature at a point has been inspired by work on non-rigid shapes by Bronstein et al. [32], where the Bronstein brothers show that curvatures provide a good feature to classify non-rigid objects. Since the RGB-D sensor provides a depth “image” (D) at the same time as an RGB image, similar operations on the depth image can be performed. This image D is essentially the same as the z -coordinate of the different points. The images can be passed through a Sobel operator to find the derivatives along the x and y -axis (Figs. 5(n) and 5(o)), and to compute the magnitude of the gradient at each point (Fig. 5(p)).

The features can be organized as follows:

- position: **P** x, y, z ,
- color: **C** r, g, b ,
- image: **IMI** $I, I_x, I_y, I_{xx}, I_{yy}, I_{xy}, I_{mag}$,
- depth: **D** D_x, D_y, D_{mag} ,
- normal: **N** n_x, n_y, n_z ,
- curvature: **Kk** C, c_1, c_2 .

Figure 5: Shown here are the different features used on the objects from the Lai et al. dataset [15]. (a) The initial image. (b) The segmented image. (c) The data points in the point cloud. (d) The normals at the points color coded in RGB space. (e) The intensity image. (f) The output of the Sobel operator along the x -axis on I . (g) The output of the Sobel operator along the y -axis on I . (h) The magnitude of the gradient of I at each point. (i) The output of the Sobel operator along the x -axis applied to I_x . (j) The output of the Sobel operator along the y -axis applied to I_x . (k) The output of the Sobel operator along the y -axis applied to I_y . (l) The curvature. (m) The depth map output D from the sensor. This corresponds to the z -coordinate of the point. (n) The output of the Sobel operator along the x -axis on D . (o) The output of the Sobel operator along the y -axis on D . (p) The magnitude of the gradient of D at each point.

Different feature vectors can be defined by combining these features, for instance:

$$\begin{aligned}
 F_C &= [r, g, b], \\
 F_P &= [x, y, z], \\
 F_{PCN} &= [x, y, z, r, g, b, n_x, n_y, n_z], \\
 F_{PCNk} &= [x, y, z, r, g, b, n_x, n_y, n_z, C, c_1, c_2], \\
 F_{PCIM} &= [x, y, z, r, g, b, I, I_x, I_y, I_{xx}, I_{yy}, I_{xy}, M], \\
 F_{PCIMDNK} &= [x, y, z, r, g, b, I, I_x, I_y, I_{xx}, I_{yy}, I_{xy}, I_{mag}, \\
 &\quad D_x, D_y, D_{mag}, n_x, n_y, n_z, C].
 \end{aligned}$$

The classification has been run with 15 different feature

vectors.

4.3. Results

The experimental procedure in [15] and [19] is followed: the database is subsampled by taking every fifth frame in order to have around 45,000 frames on which a classification is performed. During the experimental runs, the procedure by Lai et al. [15, Section V] is done as follows. For category recognition, one object is randomly left out from each category for testing and the classifier is trained on all views of the remaining objects. For instance recognition, two scenarios are considered:

- **Alternating Contiguous Frames:** Divide each video into 3 contiguous sequences of equal length. There are 3 heights (videos) for each object, therefore we have 9 video sequences for each instance. 7 of these are randomly selected for training and the test is run on the remaining 2.
- **Leave-Sequence-Out:** Train on the video sequences of each object where the camera is mounted 30° and 60° above the horizon and evaluate on the 45° video sequence.

An additional test on category recognition was carried out to check the correlation between the classification accuracy and the number of categories. In this experiment, for every chosen number $n \in [2, \dots, 45]$, n categories out of 51 are chosen randomly and the same category recognition procedure is applied in which for every category one object is randomly removed from the entire set for testing purposes and the learning is done on the remaining objects. Where the frames are randomly chosen, the experiments are run 10 times and the average is reported.

For each set of features, the covariance for each frame is computed as well as the matrix logarithm for the purpose of computing the log Euclidean distance described in Section 2.2. Classification is performed with an RBF kernel Support Vector Machine (SVM) classifier, implemented by Chang et al. [33], using the log Euclidean distance.

4.3.1. Instance Classification

Fig. 6 shows the overall results for the Leave-Sequence-Out experiment. Fig. 7 shows the results for each class, individually. Fig. 8 shows the results for the Alternating Contiguous Sequence experiment. Here as well, the results over each class are shown in Fig. 9.

The results of the experiments for instance classification are interesting in that they give insight into which

Figure 6: The results of the Leave-Sequence-Out experiment.

Figure 7: The results of the Leave-Sequence-Out experiment for each individual class.

Figure 8: The results of the Alternating Contiguous Sequence experiment. The standard deviation over 10 runs is displayed in black.

Figure 9: The results of the Alternating Contiguous Sequence experiment (contrary to Fig. 7) for each individual class.

features are important. In this scenario, the descriptors solely using color (C) information, although not producing optimal results, outperform the shape based feature descriptors (P, N, NK, NKk), Figs. 6 and 8. This was somewhat expected as the objects in each category usually present very similar shapes. The different objects have a similar overall shape inside each category which makes it difficult to classify on shape features. Color, on the other hand, is a better classifier in this case. This fact explains why the descriptor containing the most visual information ($PCIM$) performs best, a trend visible in both scenarios, leave-sequence-out and alternating contiguous frames. When combining the different features, the results become better as hypothesized.

It should be noted that the results for the Alternating Contiguous Sequences classification are slightly better than the Leave-Sequence-Out experiment. This can be explained by how the different samples are drawn in both runs. For the Leave-Sequence-Out run, all the samples from the 45° view were used for the evaluation. The training was not able to catch any samples from this angle, which explains the slightly worse performance. However, the covariance descriptor is still able to produce acceptable results.

Looking at these results, adding more features does not help in the classification. This means that for instance classification, adding more information on shape does not noticeably increase the separability. Color information is more important than shape information. Furthermore, there are some classes (3, 28, 30, 47) that produce consistently poor instance classification results over all the different feature descriptors.

To explain this phenomenon, a closer look can be taken at two objects from class 3, “banana”. Fig. 10 shows two different bananas in the database. Since in-

Figure 10: Two different objects from the class “banana”, they are very similar and thus difficult to differentiate.

Table 1: Classification accuracy. (a) Leave-Sequence-Out, (b) Alternating Contiguous Frames. Accuracies are averaged over 10 trials.

Method	Instance		Category
	(a)	(b)	
Linear SVM [15]	73.9	90.2 ± 0.6	81.9 ± 2.8
Nonlinear SVM [15]	74.8	90.6 ± 0.6	83.8 ± 3.5
Random Forest [15]	73.1	90.5 ± 0.4	79.6 ± 4.0
IDL [16]	-	91.3 ± 0.3	85.4 ± 3.2
HKDES [18]	82.4	-	84.1 ± 2.2
Kernel Desc. [17]	84.5	-	86.2 ± 2.1
CKM Desc. [19]	90.4	92.1 ± 0.4	86.4 ± 2.3
upgraded HMP [20]	92.8	-	87.5 ± 2.9
Cov Desc. (this work)	90.7	94.4 ± 2.0	80.4 ± 1.9

class similarity is considered, the shape of both objects is similar. In this case, the color is also roughly the same, the yellow shade might be only slightly different. Neither the shape nor the color information gives a definite difference between the objects, hence the classification becomes accordingly poor. However, it should be noted that in this case even a human observer would probably have trouble identifying that these are two different bananas. The comparison with other state of the art methods is given in Table 1.

4.3.2. Category Classification

For category classification, the expectation is that using more features will separate the different descriptors in the classification space. This is indeed what is ob-

Table 2: Classification accuracy and dimensionality comparison. Accuracies are averaged over 10 trials.

Method	Inter Class	Dimensionality
Linear SVM [15]	81.9 ± 2.8	4203
Nonlinear SVM [15]	83.8 ± 3.5	4203
Random Forest [15]	79.6 ± 4.0	4203
IDL [16]	85.4 ± 3.2	4203
Kernel Desc. [18]	84.1 ± 2.2	7000
HKDES [17]	86.2 ± 2.1	39000
CKM Desc. [19]	86.4 ± 2.3	19200
upgraded HMP [20]	87.5 ± 2.9	188300
Cov Desc. (this work)	80.4 ± 1.9	190

Figure 11: The results of the category classification experiment. The standard deviation over 10 runs is displayed in black.

served from the results given in Fig. 11.

Using the feature vector $F_{PCIMIDN}$ produces the best results although adding the curvature data provides very similar results. Contrary to the previous runs, in the category classification test (Fig. 11), the shape features (P, N, NK, NKk) are more meaningful than the color descriptor (C). Intuitively, this again makes sense since different objects should have a different shape, but could have the same color. For example, if in the database there is a red apple and a red toothpaste tube, the system is prone to confusion when classification is only based on color. Combining the classifiers provides better results. Unsurprisingly, the augmented color feature descriptor ($PCIM$) performs slightly worse than the shape descriptors. The covariance descriptor produces results that are reasonable compared to other classifiers (Table 1).

When the dimensionality of the different descriptors is taken into account, the results are quite surprising. The dimensionality of the presented covariance descriptor is given by $\frac{(p+1)p}{2}$, where p is the number of features, which results in 190 values. These values are also compared against the other state of the art methods and reported in Table 2. In that comparison, covariance descriptors are the best performing feature descriptors.

4.3.3. Subsampled Category Classification

In an effort to explain the category classification results, another experiment was run in which the categories were subsampled. At each run, the used categories have been redrawn. The results are shown in Fig. 12.

It is noticeable that the addition of the curvature does not significantly increase the accuracy of the classification. This raises the point that some features might be more suitable than others for classification. In the scope

Figure 12: The classification accuracy results for the subsampled categories experimental run. The x-axis shows the number of subsampled categories for each run.

Figure 13: The classification accuracy results for the subsampled categories experimental run with additional features. The x-axis shows the number of subsampled categories for each run. A descriptor using these additional features consistently performs better than the initial feature vector (black).

of this experiment, (PCN), ($PCNK$) and ($PCNKk$) all perform similarly well indicating that information is contained in the normal distribution of the objects, but the curvatures do not add more separability. The augmented color descriptor ($PCIM$) does not greatly improve the classification either. This echoes the fact discussed in the previous section that for category classification, shape dominates over color. The performance of the additional features can be seen in Fig. 13.

A descriptor using these feature vectors consistently performs better than the initial best feature vector in Fig. 12 (black line). It is observable that the different descriptors with or without curvature, perform similarly well. Also, depth image information only slightly improves the performance compared to the descriptors that do not use this information. This shows that a combination of position, color, and surface normals provides reasonable results without necessarily using the curva-

ture or the depth image information. In the next section, the paradigm is shifted to a dictionary learning based classification. There has been significant research done using classifiers based on dictionary learning [34, 35]. This approach would be similar to the method proposed by Blum et al. [19]. Bo et al. [20] also use dictionary learning, but add an additional hierarchical model on top of their classification paradigm.

4.4. Dictionary Learning Classification

To consider another classification paradigm, a dictionary learning approach has been evaluated. In this scheme, the classification is realized by using the reconstruction error of a sample through dictionaries that have been trained on different classes. The smallest reconstruction error classifies the sample. Mairal et al. [36] have shown successful classification on images using this technique.

In this test, only the subsampled category classification was run. Three different dictionary sizes are used: 32, 64 and 128 atoms. The more atoms available in the dictionary, the better the reconstruction should be. The test also spans four different sparsity levels for the coding as well as the reconstruction: 1, 2, 4, and 8.

Figs. 14 and 15 show the classification results for the different feature vectors introduced in the previous sections. For comparison purposes, the curve for $F_{PCIMIDNKK}$ using the SVM classification is shown in black. Fig. 16 compiles the classification results using a dictionary of size 128 and a sparsity level of 8.

In Figs. 14 and 15, it is interesting to note that the sparsity level, not the dictionary size, regulates the classification outcome. A sparsity level of 8 consistently outperforms the other sparsity levels for each dictionary size. Furthermore, the same sparsity level on a smaller dictionary size outperforms lower sparsity levels on larger dictionaries. This is a reasonable result however, as having a higher sparsity allows for more dictionary atoms to be used which can lower the reconstruction error. For instance, the classification for a sparsity level of 1 can be viewed as a “one-nearest-neighbor” classification since it finds the atom in the dictionaries that is closest to the sample.

The dictionary classification does not provide a significant improvement in the performance compared to the previous SVM classification. To improve on the dictionary classification it would probably be necessary to adjust the learning of the dictionaries to become more discriminative following the ideas introduced by Mairal et al. [36].

Figure 14: The classification results using dictionary learning for dictionary sizes 32, 64, and 128 along with sparsity levels 1, 2, 4, and 8. The respective features used are $F_{PCIMIDNK}$, $F_{PCIMIDN}$, F_{PCIMID} , $F_{PCIMINK}$, $F_{PCIMINKk}$ and F_{PCIMIN} . For comparison, $F_{PCIMIDNKK}$ results using SVM classification are shown in black.

Figure 15: The classification results using dictionary learning for dictionary sizes 32, 64, and 128 along with sparsity levels 1, 2, 4, and 8 for $F_{PCIMIDNKK}$. For comparison, the $F_{PCIMIDNKK}$ descriptor results in the SVM setup are shown in black.

Figure 16: The category classification results using dictionary learning for a dictionary of size 128 and a sparsity level of 8 for different feature vectors. For comparison, $F_{PCIMIDNk}$ results in the SVM setup are shown in black.

5. Robotic Application

The previous dataset contained only noiseless data in which the objects were not occluded. When occlusions are present, there is a definite advantage of having a dynamically moving sensor compared to a statically placed camera. Mobile robots are ideal sensor carrying platforms. Their ability to move about the environment instead of remaining in a fixed location (e.g., industrial robots) allows them to dynamically sense their surroundings. As research expands in the area of robotics, mobile robots can be expected to increasingly adopt the role of performing object recognition in various scenarios.

5.1. Robot Description

The Microvision robot was designed at the University of Minnesota’s Center for Distributed Robotics. Its shape is close to that of previous robots such as the COTS Scouts [?] and the eROSIs [?]. It is composed of a cylindrical body with two wheels and a tail (Fig. 17). The tail allows the robot to tilt about the wheel axis. This is useful not only to register different slices of the environment from the laser range finder, but also to provide another angle for the mounted camera and RGB-D sensor.

The increase in size of the robot enables the mounting of more sensors that could not be mounted on the older models. The robot is equipped with a Hokuyo URG-04LX laser range finder. The field of view of the sensor is 240° with 0.36° resolution. It has a refresh rate of 10 Hz and the maximum range is approximately 4 m. The embarked camera is a half inch CCD camera with a resolution of 640×480 pixels.

Figure 17: The Microvision robot equipped with an RGB-D sensor.

The robot’s capabilities have been improved by Beksi et al. [39]. Mounted on top of the Microvision is the Asus Xtion Pro Live. The Xtion is a motion sensing device equipped with an RGB-D camera and a pair of microphones. The depth camera has a range of 0.8 m to 3.5 m. It has viewing angles of 45° in the vertical direction, and 58° in the horizontal direction. The Xtion is positioned at a 90° offset relative to the base of the robot, which points the sensor towards the y-axis of the reference frame attached to the robot.

The Microvision runs the Robot Operating System (ROS) [40]. To facilitate occlusion alleviation, a ROS node has been developed which implements the motion scheme from Section 5.2. This node performs three main services. First, it implements a callback function for receiving point cloud data from the RGB-D sensor. Second, the node processes the point cloud data and computes a new aiming point as described in the motion scheme. Lastly, the node sends linear and angular velocities to the Microvision ROS node based on the location of the computed aiming point.

The Microvision ROS node functions as the driver for the robot and provides control to the robot’s wheels and tail. The node subscribes to linear and angular velocity messages published by the motion scheme node. Throughout this process the point clouds are saved to disk for offline processing. The feature computation and classification are currently being optimized in order to perform the task in real-time.

5.2. Motion Strategy

The motion scheme involves the robot circumnavigating an object cluster to obtain different points of view of the objects within the cluster. This strategy evolved from work by Fehr and Papanikolopoulos [41], which

Figure 18: The points and values defined in Section 5.2. The robot at position R' aims for point A .

involved the use of a laser range finder to detect objects in a 180° field of view (FOV). The Xtion's smaller FOV requires certain adjustments in the strategy. The motion scheme simply controls the linear and rotational velocities of the robot to follow an aiming point that is recomputed at each sampling step.

Data from the RGB-D sensor is used to compute the projection of the detected objects, and an aiming point used for velocity controls. The RGB-D sensor is pointed along the y_R -axis of the robot. This pose enables the robot to smoothly circle the objects without having to rotate the camera in order to keep the objects of interest in the FOV.

First, the points on the objects are projected onto the ground plane. Next, their centroid is computed in the ground plane. Finally, the aiming point A is computed along with the different points and angles used in the computation. R' is the point at which the robot should circle the objects. C is the centroid of all detected objects. A is computed as the intersection of the tangent to a circle centered at C of radius d with another circle centered at C with radius $d + b$, where b is an arbitrary value. These points and values are illustrated in Fig. 18.

A choice is made for $y_{R'}$ to point towards C , in order to have the camera pointing towards the circled objects. In Fig. 18, the blue square represents one of the detected objects that is the farthest away from the centroid. This distance defines d_m . d_0 sets the distance at which the objects should be circled. The offset d_0 is introduced to avoid collisions with the objects to estimate. From here,

Figure 19: The lengths and angles used for the velocity computation.

the aiming point A can be defined in the robot centric coordinate frame as follows:

$${}^C A = \begin{pmatrix} r \cos \theta \\ r \sin \theta \end{pmatrix} \quad (15)$$

$${}^{R'} A = {}^{R'} C + {}^{R'} T({}^R C) {}^C A \quad (16)$$

$${}^{R'} A = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + \begin{pmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{pmatrix} \begin{pmatrix} r \cos \theta \\ r \sin \theta \end{pmatrix} \quad (17)$$

where

$$d = \|CR'\| \quad (18)$$

$$= d_0 + d_m \quad (19)$$

$$r = \|CA\| \quad (20)$$

$$= d + b \quad (21)$$

$$\psi = -\frac{\pi}{2} \quad (22)$$

$$\theta = \widehat{R'CA} \quad (23)$$

$$\widehat{AR'C} = \perp \quad (24)$$

$$\cos \theta = \frac{d}{r}. \quad (25)$$

After the aiming point A has been determined, the velocities for the robot can be determined. Fig. 19 shows the different lengths and angles involved in the computation.

The linear velocity V is set proportionally to the distance to the aiming point:

$$V = K_v \|RA\|. \quad (26)$$

The rotational velocity ω needs an additional constraint so that if the robot moves towards A , the objects remain in the sensor's FOV. To enforce this constraint, the rotational velocity is set proportionally to the weighted mean β of the angles ϕ and ψ (Fig. 19):

$$\omega = K_\omega \beta. \quad (27)$$

ϕ is the angle under which the robot sees the aim point A from the forward direction (robot x -axis). ψ is the angle that centroid C deviates from the sensor viewing axis (robot y -axis),

$$\phi = \tan^{-1}(y_A, x_A) \quad (28)$$

$$\psi = \tan^{-1}(y_C, x_C) - \frac{\pi}{2} \quad (29)$$

$$w \in [0, 1] \quad (30)$$

$$\beta = w\psi + (1 - w)\phi. \quad (31)$$

The weight w allows for modifying the emphasis on either angle. To keep the objects in the FOV, w has been arbitrarily set to 0.7. Fig. 20 describes of how the mobile robot determines its linear and rotational velocities.

The weighted average between the angles ϕ and ψ is used to compute the rotational velocity. Intuitively, the robot tries to reach the desired position R' by receiving linear and rotational velocity commands, while at the same time trying to keep the objects in the FOV.

5.3. Data Processing

5.3.1. Training

A mobile robot needs to navigate in an environment independently and cannot rely on human input to segment its surroundings; manual segmentation is therefore not an option. To perform classification, the data must be segmented. In an effort to perform automatic detection, a new strategy is implemented. For classifier training, the different objects are assumed to be on the floor with enough space between them to clearly and easily separate them. Fig. 21(a) shows a sample input following these assumptions. In later parts of this section, more objects are allowed to be in the scene. These assumptions greatly simplify automatic segmentation. From a point cloud scan (Fig. 21(a)), the points on the floor (Fig. 21(b)) are estimated by plane fitting using the Random Sample Consensus (RANSAC) method [42].

A by-product of the floor removal step is the computation of the normal to the floor plane. Once an estimation of this normal is available, the point cloud can be

Figure 20: A view from the robot with the aiming point and the different directions used to determine the rotational velocity. The origin of the robot's reference frame is positioned on the blue point. The x and y axes of this frame are drawn in red and green respectively. The detected objects are given in green. The objects' centroid is shown in red and the aiming circle is shown in yellow. The aiming point, computed according to Eqs. (15) – (25), is colored cyan. The cyan axis shows the direction of the aiming point from which ϕ can be determined. In yellow, the direction of angle ψ is given. This is the direction of the object centroid (red) rotated by -90° . Finally in magenta, the direction of β is provided, which is used to compute the rotational velocity.

transformed into a coordinate frame in which its z -axis and the normal line up. This produces the coordinates of the point cloud in the global frame, under the assumption that the global frame's x and y -axis lie in the floor plane. An arbitrary choice of origin can be achieved by a simple translation in the xy -plane. In this frame, the object points used in the computation of the aiming point (the green points in Fig. 20) are then simply a projection onto the xy -plane along z .

After the removal of the floor, only objects of interest remain in the scene (Fig. 21(c)). At this point, a simple clustering based on Euclidean distance is performed to separate the different groups of points. A point P_2 is considered to be part of the same cluster as point P_1 if the Euclidean distance between P_1 and P_2 , is smaller than an arbitrarily value r , i.e. $\|P_1 - P_2\|_2 < r$. In other words, P_2 is considered to belong to the same clusters as P_1 when $P_2 \in B(P_1, r)$, the ball centered at P_1 with

Figure 21: The different steps for the object segmentation are shown. (a) The initial input cloud. (b) The estimated floor. (c) The object minus the floor. (d) The object of interest.

radius r . This check is performed over all the points until they are all part of a cluster.

Since the object of interest is assumed to be the only object in the FOV, it is segmented as the object in the foreground (Fig. 21(d)) and closest to the sensor. The covariance descriptor is then computed and SVM training is performed on all the covariances from the views of the segmented objects. Only nine features contribute to the covariance in this experiment:

$$f = [x, y, z, R, G, B, n_x, n_y, n_z]. \quad (32)$$

The features used are the Cartesian coordinates (x, y, z) , the color channel values (R, G, B) , and the normal coordinates (n_x, n_y, n_z) at the specific point. The smaller feature vector makes the computation of the descriptor fast although there is some sacrifice in accuracy.

5.3.2. Objects

A new object database is created. The training data consists of views of six different objects collected by the circling Microvision robot. The objects used can be seen in Fig. 22. The robot is set up to navigate around the different objects following the strategy described in Section 5.2. Fig. 23 shows the setup of the experiment.

Fig. 24 shows the robot's computation of the aiming point, shown in cyan, from the scene in Fig. 23. The robot is represented by a blue dot and the objects are displayed in green. The centroid of the objects is drawn in red and the aiming circle in yellow. The different angles used for the rotational velocity computation are given in Section 5.2 with differently colored directions.

Figure 22: The different objects used in the experiment. The top row shows the robot's view of the object and the bottom row depicts the color coding corresponding to each object.

Figure 23: Robot and object setup.

5.4. Classification

Figs. 25 – 27 provide results for different experimental runs. While circling, the robot is able to capture and recognize the different objects. When there is no occlusion (Fig. 25), the segmentation of the objects is relatively easy and the covariance descriptor provides excellent results. In some cases where there is slight overlap (Figs. 26 and 27), the classification still provides good results.

A difficulty in this approach is encountered when the segmentation merges parts of the point cloud that should not be merged, or separates parts that should be together. Fig. 28 shows an example of this problem, which is from the same run as Fig. 27. In this figure, the coffee can and paper roll clusters get merged and as a consequence they are misclassified. A better and more robust segmentation scheme is necessary to address this issue.

When there is too much occlusion the classification breaks down. However, the circling robot can provide another point of view from which it is possible to classify correctly. Fig. 29 shows an instance in which occlusion leads to misclassification. This view is taken from the same run as in Fig. 27. Only the shuttle's nose is in

Figure 24: The aim point (cyan) computed by the robot (blue) along with the aiming circle (yellow) around the objects (green) and their centroid (red) from the scenario depicted in Fig. 23. As in Section 5.2, the lines show the angles used for the rotational velocity computation.

the robot's field of view. If the robot can only see parts of an object, then it can be problematic to categorize the object since the classifier has been trained on full, non-occluded views. As the robot moves around the target, the FOV changes to allow for better classification.

5.5. Scale-Invariance

For the six objects given in Fig. 22, an experiment was performed where a point cloud of each object was captured at four increasing distances from the RGB-D sensor corresponding to 2-8 ft (0.61-2.43 m). Fig. 30 shows one such set of images. At 8 feet, the object is near the far-end range of the sensor's capabilities. For each combination of features, we computed the norm of the geodesic distance between each object and itself, and object to object at the various positions. The object's feature vector consisted of the following parameters: $F_{PCN} = [x, y, z, r, g, b, n_x, n_y, n_z]$. The average norm of the geodesic distance between each object and itself, at different scales, was found to be 0.104. The average norm between each object and all other objects was 0.204. This corresponds to a difference of 48.8% between the different objects, and shows a clear distinction in the geodesic distance regardless of the scale.

A second experiment to test the classification performance of the scaled data set was performed. The scaled object was passed to the SVM classifier trained on the

Figure 25: The result of a classification run with the first object cluster. The objects' color code given in Fig. 22.

Figure 26: The result of one classification run with the second object cluster. The objects' color code given in Fig. 22.

data from Section 5.3 to predict the label. In all cases, the prediction performed flawlessly, i.e. a misclassification rate of zero. It is worthy to note that at the smallest scale, the point cloud composition of the object is less than 1% of the total point cloud size, yet the classification was consistently performed correctly. These runs provide strong empirical evidence that the covariance descriptor is scale invariant.

5.6. Majority Voting

To enhance the final classification results, an approach was taken based on a majority voting method. Since the computed covariance matrices are stored in memory from the previous frame, we use them to follow the classification progress of the object clusters throughout the experiment. Specifically, we track the different clusters that represent the same object in consecutive images.

An intuitive way of following the progress makes use of the logarithmic distances of the matrices in the covariance space. The shorter the distance, the greater the

Figure 27: The result of a classification run with the third object cluster. The objects' color code given in Fig. 22.

Figure 28: An instance where poor segmentation leads to misclassification. The coffee can and paper roll clusters are merged.

Figure 29: An instance where occlusion leads to misclassification. Most of the shuttle is occluded which leads to that segment being classified as part of the shoe.

(a) 2 ft

(b) 4 ft

(c) 6 ft

(d) 8 ft

Figure 30: The shuttle object at four different scales.

resemblance between two clusters, and thus there is a greater likelihood for the two instances to refer to the same object. We verified experimentally that this is a sufficient way of assigning clusters to their respective object.

The majority voting algorithm is described as follows. The segmentation clusters the points of the RGB-D image and the classifier assigns a label to every cluster. These labels are stored in a histogram that contains bins for every object of the scene the algorithm detects. Every time a new cluster appears, the cluster's covariance matrix is compared to the previous RGB-D image's clusters' covariance matrices. If this distance is greater than a prespecified threshold, then a new bin is added in the histogram to indicate that a new object has appeared. At the end of the experiment, the histogram has stored all the different decisions of the classifier regarding the objects and assigns to every object the label with the maximum votes.

As an example, assume object O_x and its assigned bin in the histogram B_x . B_x contains all the labels classifying the several instances of O_x . O_x is then classified with the label of the object that has the majority of entries in

Figure 31: These histograms show the results of the majority voting algorithm for the objects in Fig. 25. Cluster 1 is the coffee can (green), cluster 2 is the box (red) and cluster 3 is the shoe (magenta).

B_x . Fig. 31 shows the voting results for the scene given in Fig. 25.

6. Conclusion and Future Work

Contrary to image processing, where some algorithms are well established, there is still no clear consensus on how RGB-D data is best tackled, which provides interesting research opportunities in the area of point cloud data processing. One such opportunity has been presented in this work where a novel covariance based descriptor for object classification and detection in 3D “colored” point clouds has been introduced. This research, in establishing the foundation of a covariance approach in point cloud processing, has identified and begun to address some of the drawbacks of traditional histogram-based classifiers. The advantages in using covariances against other descriptors are the following:

1. **Compactness.** A dimensionality reduction without any significant loss in recall has been shown, which implies lower computational and storage requirements.
2. **Flexibility.** The addition of new features has been shown to be very easy, without any significant change in memory or computation time. This provides a big advantage compared to histogram-based approaches in which adding a feature amounts to the addition of a dimension in the histogram cube.
3. **Parameter-freeness.** Compared to other histogram-based models such as spin images, there is no need to tune parameters such as bin size or bin number.
4. **Distinctiveness.** The covariances possess an inherent ability to subtract out common features available in the data and consider only the discriminative information. For example, when used for pointwise detection, if a region is planar then the covariance will be rank deficient, which might be useful at times to prune out non-informative regions from the point clouds.

These covariance descriptors have been applied to different point cloud data. The results are comparable to other state of the art methods. In addition, the use of a covariance classifier in conjunction with a mobile robot has been demonstrated. A moving sensor has clear advantages over a static sensor in that it can position itself to alleviate occlusions.

While significant strides have been made in object classification using RGB-D sensors, the area is still in its early steps and there remains a multitude of avenues left open for exploration. Covariances provide a new way of looking at point clouds and the full potential of such an approach has yet to be identified. Different issues and problems remain to be addressed:

- **Theoretical analysis** — In this work, different features have been identified and tested. Some provided better separability than others. A theoretical description of the different features could answer the question of which features provide better results and the reason behind their success.
- **Feature development** — The features introduced in this work have mostly been obvious choices of selection. They were relatively easy and fast to compute. There might be other features that would produce better classification results. However, the computation time will have to be balanced. New features would not be of great benefit if their computation at each point of the point cloud incurs a high cost in terms of time and resources.
- **Next view problem** — The dynamic placement a sensor mounted on a mobile robot provides needs to be taken advantage of to handle occlusions. A simple circumnavigation of objects, although producing views from different angles, might not be sufficient. More directed motion planning, for instance towards areas of low sampling density, needs to be considered.

Acknowledgements

This material is based in part upon work supported by the National Science Foundation through grants #IIP-0934327, #IIS-1017344, #CNS-1061489, #CNS-1138020, #IIP-1332133, #IIS-1427014, and #IIP-1432957.

References

- [1] E. Schwalbe, H. Maas, and F. Seidel, "3D building model generation from airborne laser scanner data using 2D GIS data and orthogonal point cloud projections," in *Proceedings of the International Society for Photogrammetry and Remote Sensing WG III/3, III/4*, vol. 3, 2005, pp. 12–14.
- [2] J. Weingarten, G. Gruener, and R. Siegwart, "A state-of-the-art 3D sensor for robot navigation," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2004, pp. 2155–2160.
- [3] C. Ip and S. Gupta, "Retrieving matching CAD models by using partial 3D point clouds," *Computer Aided Design and Applications*, vol. 4, no. 5, pp. 629–638, 2007.
- [4] P. Allen, I. Stamos, A. Troccoli, B. Smith, M. Leordeanu, and Y. Hsu, "3D modeling of historic sites using range and image data," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA*, vol. 1, 2003, pp. 145–150.
- [5] Kinect. [Online]. Available: <http://www.xbox.com/en-US/xbox-one/accessories/kinect-for-xbox-one>
- [6] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 51, no. 5, pp. 433–449, May 1999.
- [7] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001, pp. 67–76.
- [8] C. Bajaj, F. Bernardini, and G. Xu, "Automatic reconstruction of surfaces and scalar fields from 3d scans," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 1995, pp. 109–118.
- [9] O. Carmichael, D. Huber, and M. Hebert, "Large data sets and confusing scenes in 3D surface matching and recognition," in *Proceedings of the Second International Conference on 3D Digital Imaging and Modeling*, 1999., Oct 1999, pp. 358–367.
- [10] J. Assfalg, M. Bertini, A. Bimbo, and P. Pala, "Content-based retrieval of 3D objects using spin image signatures," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 589–599, 2007.
- [11] X. Li, A. Godil, and A. Wagan, "3D part identification based on local shape descriptors," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, 2008, pp. 162–166.
- [12] J. W. Tangelde and R. C. Veltkamp, "A survey of content based 3D shape retrieval methods," *International Conference on Shape Modeling and Applications*, vol. 0, pp. 145–156, 2004.
- [13] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA2009)*, 2009, pp. 1848–1853.
- [14] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proceedings of the 11th European Conference on Computer Vision: Part III*. Springer-Verlag, 2010, pp. 356–369.
- [15] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1817–1824.
- [16] —, "Sparse distance learning for object recognition combining rgb and depth information," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 4007–4013.
- [17] L. Bo, X. Ren, and D. Fox, "Depth kernel descriptors for object recognition," in *Proceedings of the 2011 IEEE Conference on Intelligent Robots and Systems(IROS)*, 2011, pp. 821–826.
- [18] L. Bo, K. Lai, X. Ren, and D. Fox, "Object recognition with hierarchical kernel descriptors," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1729–1736.
- [19] M. Blum, J. Springenberg, J. Wulping, and M. Riedmiller, "A learned feature descriptor for object recognition in rgb-d data," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1298–1303.
- [20] L. Bo, X. Ren, and D. Fox, "Unsupervised feature learning for rgb-d based object recognition," *ISER*, 2012.
- [21] J. Tang, S. Miller, A. Singh, and P. Abbeel, "A textured object recognition pipeline for color and depth image data," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3467–3474.
- [22] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Computer Vision ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin / Heidelberg, 2006, vol. 3952, pp. 589–600.
- [23] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on lie algebra," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 728–735.
- [24] W. Förstner and B. Moonen, "A metric for covariance matrices," *Quo vadis geodesia*, pp. 113–128, 1999.
- [25] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Log-euclidean metrics for fast and simple calculus on diffusion tensors," *Magnetic Resonance in Medicine*, vol. 56, no. 2, pp. 411–421, 2006.
- [26] W. Zheng, H. Tang, Z. Lin, and T. Huang, "Emotion recognition from arbitrary view facial images," in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin / Heidelberg, 2010, vol. 6316, pp. 490–503.
- [27] C. Yuan, W. Hu, X. Li, S. Maybank, and G. Luo, "Human action recognition under log-euclidean riemannian metric," in *Computer Vision ACCV 2009*, ser. Lecture Notes in Computer Science, H. Zha, R.-i. Taniguchi, and S. Maybank, Eds. Springer Berlin / Heidelberg, 2010, vol. 5994, pp. 343–353.
- [28] Y. Pang, Y. Yuan, and X. Li, "Gabor-based region covariance matrices for face recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 7, pp. 989–993, 2008.
- [29] C. Ye, J. Liu, C. Chen, M. Song, and J. Bu, "Speech emotion classification on a riemannian manifold," in *Advances in Multimedia Information Processing - PCM 2008*, ser. Lecture Notes in Computer Science, Y.-M. Huang, C. Xu, K.-S. Cheng, J.-F. Yang, M. Swamy, S. Li, and J.-W. Ding, Eds. Springer Berlin / Heidelberg, 2008, vol. 5353, pp. 61–69.
- [30] Y. Shinohara, T. Masuko, and M. Akamine, "Covariance clustering on riemannian manifolds for acoustic model compression," in *IEEE International Conference on Acoustics Speech and Signal Processing ICASSP*, Mar. 2010, pp. 4326–4329.
- [31] R. Gonzales and R. Woods, *Digital Image Processing (Third Edition)*. Pearson Prentice Hall, 2008.
- [32] A. Bronstein, M. Bronstein, and R. Kimmel, *Numerical Geometry of Non-Rigid Shapes*, 1st ed. Springer Publishing Company, Incorporated, 2008.
- [33] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 27, pp. 1–27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [34] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and

- clustering via dictionary learning with structured incoherence and shared features,” in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 3501–3508.
- [35] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.
 - [36] J. Mairal, M. Elad, and G. Sapiro, “Sparse representation for color image restoration,” *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, Jan 2008.
 - [37] I. Burt, A. Drenner, C. Carlson, A. D. Kottas, and N. Papanikolopoulos, “Impact orientation invariant robot design: An approach to projectile deployed robotic platforms,” in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006.
 - [38] M. Walter, M. Anderson, I. Burt, and N. Papanikolopoulos, “The design and evolution of the erosi robot,” in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Apr 2007, pp. 2984–2989.
 - [39] W. J. Beksi, K. Choi, D. Canelon, and N. Papanikolopoulos, “The microvision robot and its capabilities,” *Department of Computer Science, University of Minnesota, Tech. Rep*, 2015.
 - [40] ROS. [Online]. Available: <http://www.ros.org>
 - [41] D. Fehr and N. Papanikolopoulos, “Using a laser range finder mounted on a microvision robot to estimate environmental parameters,” in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2009, p. 733211.
 - [42] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.