Recognition of 3D Package Shapes for Single Camera Metrology

Ryan Lloyd, Scott McCloskey
Ryan.Lloyd@Honeywell.com, Scott.McCloskey@Honeywell.com
Honeywell ACS Labs, Golden Valley, MN

Abstract

Many applications of 3D object measurement have become commercially viable due to the recent availability of low-cost range cameras such as the Microsoft Kinect. We address the application of measuring an object's dimensions for the purpose of billing in shipping transactions, where high accuracy is required for certification. In particular, we address cases where an object's pose reduces the accuracy with which we can estimate dimensions from a single camera. Because the class of object shapes is limited in the shipping domain, we perform a closed-world recognition in order to determine a shape model which can account for missing parts, and/or to induce the user to reposition the object for higher accuracy. Our experiments demonstrate that the addition of this recognition step significantly improves system accuracy.

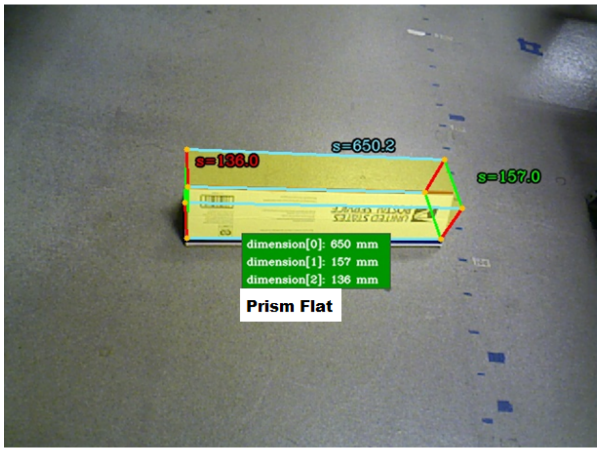


Figure 1: A prism with self-occlusion. The back half of the object is not visible. Our system recognizes the package as a prism laying flat and exploits symmetry of the prism to include the absent back half in measurements. The system fits a minimum volume bounding box around the package. The ground truth dimensions are length=648 mm, width=153, and height=136.

1. Introduction

The estimation of an object's dimensions has long been of interest in many commercial applications, but the high costs of 3D cameras has - to date - precluded their use in retail settings with lower budgets. As such, the advent of low-cost range cameras like the Microsoft Kinect has reinvigorated the interest in vision-based metrology for retail settings. We address the application of Microsoft Kinect-like active stereo cameras to object metrology for retail shipping operations, where the dimensions of an object are used - in part - to determine the shipping fare. Because the estimated dimensions are used for a commercial transaction, most countries require certification of the system's accuracy under the intended use case. This includes, in particular, accuracy testing under general object poses relative to the camera.

There are several object poses which are problematic for single-camera metrology. Consider the case of a prism lying on the ground such that only one face is visible to the camera, as shown in Figure 1. In this case, the three dimensions of the smallest cuboid bounding the observed points on the object are smaller than the dimensions of the smallest cuboid bounding *all* of the object, so a naive algorithm would produce a significant under-estimate of the object's depth. The same issue exists for cylinders in a similar pose, due to self-occlusion of the object. While it is conceptually straightforward to add a second camera which provides complementary views of missing surfaces, the associated doubling of cost and installation complexity weighs against such an approach, and engineering issues such as the interference of two cameras' active illumination also makes it unappealing.

We note that cases of self-occlusion can be handled by extrapolating the missing surfaces *if the shape of the object is known*. Instead of adding a second camera, then, we take advantage of the limited domain of shipped objects (the vast majority of them are either cuboids, cylinders, or prisms). Starting from the 3D data produced by an active stereo camera, we construct a 2D feature histogram which is classified using a Support Vector Machine (SVM) to determine a combination of object shape and pose. As in the figure above of the prism, shape models improve the dimensioning accuracy of cases with self-occlusion.

In addition to handling self-occlusion, we use shape recognition to identify cases where limits of the sensor's accuracy may lead to high dimensioning errors. As in other 3D sensing modalities, active stereo systems suffer from 'mixed pixels' [4], where the depths of two different parts of the scene combine to give erroneous 3D points that don't correspond to real world objects. By detecting object configurations that give rise to large depth discontinuities - namely upright cylinders and prisms - we tell the user to re-position a package for a better estimate.

Our method is evaluated at both the component and system level. At the component level, we test on a set of about 1500 images, and find a recognition accuracy of 99%. At the system level, we demonstrate that the use of shape-based models and rejection of bad poses significantly improves measurement accuracy in the presence of self-occlusion. On a subset of around 500 test images with known ground truth dimensions, mean absolute error of the three dimensions is reduced by 72/72% for prisms/cylinders/boxes, respectively, with a small (5%) increase in mean absolute error for boxes arising from a few cases of mis-classification. On balance, this brings the system performance on prisms and cylinders in line with the box performance, and should allow for certification.

2. Related Work

Hetzel et al. present a method for recognizing objects in range images by organizing features into histograms [5]. They use three local features at each measured point: shape index (derived from curvature) [7], surface normal in spherical coordinates (θ, ϕ), and "pixel depth" normalized to [0, 256). The "pixel depth" of a point is the depth from the sensor to the point, but the depths over an object are normalized to the byte scale. They compute these features at all points for a particular object and then build a histogram from the features. However, the "pixel depth" feature is too sensitive to changes in object pose, and thus requires a larger training set than is practical in our case. Given a test object, they compute the χ^2-divergence between the test object and all histograms residing in a training database. The trained object with the lowest χ^2-divergence indicates the test object's class (and even pose). That is, they use 1-NN classification, so both the accuracy and runtime are dependant on the number of labeled training samples.

Other popular object recognition schemes with 3D data include spin images from Johnson and Hebert [6]. Körtgen et al. [8] use 3D shape contexts, extending the 2D shape contexts of Belongie et al. [1]. Radu et al. describe 3D objects with several different feature histograms, including fast point feature histograms [12] and viewpoint feature histograms [13]. Their features are based on relationships between an anchor point and points in its neighborhood. Histograms are local; each anchor point has its own histogram. These histograms are invariant to point density, and they satisfy other invariants.

Many of these methods provide finer grain recognition than the method we present, at an unacceptably high computational cost. We have a relatively small number of classes, and have no need to distinguish fine pose differences as in previous work. Our method needs to be fast, due to the high throughput of retail shipping centers, and produce an accurate classification in under 300ms.

Like our system, several previous methods use SVMs in object recognition. In [11], Roobaert et al. use SVMs to recognize objects from 2D RGB images. They use a 3-dimensional vector per image, which simply equals the average RGB intensities. Similarly, Chapelle et al. classify 2D color images by forming color model histograms of the images (in HSV space) and then using the histograms as feature vectors in SVMs [3]. In the case of packages, methods like these that use color features do not help in classification. Most packages, regardless of shape, are either cardboard brown or white. Packages often have logos and writing in unpredictable colors.

3. Method

In this section, we discuss our approach to segmenting, describing, recognizing, and estimating the size of packages. This method works with any sensor that provides information from which a computer can build an organized 3D point cloud, i.e. 3D points arranged in a 2D matrix corresponding to the rasterization of the range image. For each case, we use an organized point cloud derived from a single disparity image from the range camera. Note that we only consider the depth data produced by the sensor, ignoring color information.

In light of the shipping domain and the differing performance observed between upright and lying packages, we recognize the following five classes:

- Rectangular boxes, i.e. cuboids
- Right circular cylinders laying flat on the ground
- Right circular cylinders standing vertically
- Right regular prisms with triangular bases laying flat on the ground
- Right regular prisms with triangular bases standing vertically

Much of the needed discrimination can be achieved by considering two features of points on the object: curvature c and orientation θ relative to the ground plane. Regular boxes, for instance, will be dominated by points with zero curvature, and with orientations parallel or orthogonal to the ground (for the top and lateral faces, respectively). Surface

Class	**Surface**	θ **(rad)**	**c**
Box			
	Lateral faces	$\frac{\pi}{2}$	0
	Top face	0	0
Cylinder Flat			
	Lateral surface	0 to π	> 0
	Circular bases	$\frac{\pi}{2}$	0
Cylinder Vertical			
	Lateral surface	$\frac{\pi}{2}$	> 0
	Circular base	0	0
Prism Flat			
	Lateral faces	$\frac{\pi}{3}$	0
	Triangular bases	$\frac{\pi}{2}$	0
Prism Vertical			
	Lateral faces	$\frac{\pi}{2}$	0
	Triangular base	0	0

Table 1: Expected curvature (c) and orientation (θ) features over the surfaces of ideal geometric shapes.

points on a cylinder lying flat will have non-zero curvature and a continuous range of orientations w.r.t. the ground. Table 1 breaks down the features we expect over ideal package shapes. The table considers only smooth surfaces and not edges or corners, since these constitute a negligible fraction of surface points.

Considering just the histograms of c and θ, boxes and vertical prisms are likely to be confused. We add another feature describing the shape of the package's occluding contour, as viewed from above, which is one of three simple geometric shapes: equilateral triangle, rectangle, or circle. Table 2 lists the likely categorical value of this extra trait for each class.

Class	**Top Face Shape**
Box	Rectangle
Cylinder Flat	Rectangle
Cylinder Vertical	Circle
Prism Flat	Rectangle
Prism Vertical	Equilateral Triangle

Table 2: Expected top face shape for each class of packages.

3.1. Algorithm Overview

The algorithm must first locate the ground plane and segment bodies above the plane. For each segmented package, the algorithm follows this outline:

1. Smooth the object with moving least squares
2. Compute the surface normal at each point
3. Determine the angle, θ, between each point's surface normal and the ground normal
4. Compute curvature, c, at each point
5. Fill a 2D histogram with dimensions c and θ
6. Determine the 2D shape of the top of the package
7. Create a 201-dimensional feature vector containing the histogram elements and the top shape descriptor
8. Classify the feature vector with a SVM

3.2. Estimating a Ground Plane

Finding a ground plane accurately is important because we examine objects relative to it. Identifying some surface as the ground provides an early step in segmenting bodies. We define the ground plane as the largest plane in the cloud, which usually is the floor around the package(s). The largest plane has the most points near it (that is, in the plane's support) among all other planes.

We search for the largest plane in a structured manner. We break the 2D image plane into a regular grid of $n \times n$ pixel (px) cells. We use n=32 px, which gives a suitable tradeoff between the number of cells (and thus speed) and the likelihood that at least one 32×32 patch contains mostly ground. For each cell, we locally find the 3D points at positions $(0, 0)$, $(0, n - 1)$ and $(n - 1, 0)$. We compute the 3D plane through these three points and determine the support. The support mask contains all points within a maximum distance (such as 20 mm) from the plane. After finding the plane of biggest support (in terms of px), we compute a plane of best fit through the support points via least squares.

3.3. Segmenting Above-Ground Objects

After finding the ground plane, we find above-ground points and segment them into bodies. Above-ground points are those in the direction of the ground's normal.

Euclidean clustering segments above-ground objects in an organized point cloud. The algorithm is based purely on clustering points that are spatially close in XYZ space and near each other on the image plane. We start searches from seed points on a regular grid. A search proceeds in a depth-first manner, using 4-connectivity. When checking a neighbor of the current node, the algorithm adds the neighbor to the stack if its L1 distance from the current node is below a threshold. Euclidean clustering can help segment bodies that overlap on the image plane but are otherwise separated in 3D space. Figure 2 gives an example of segmenting packages.

3.4. Smoothing the Object

After isolating the 3D points belonging to a package, we use moving least squares (MLS) [9] to smooth the surface of the package. Levin gives an overview of how MLS works [10]. We use MLS from the Point Cloud Library [14]. Smoothing the cloud is essential for accurately computing local features. Without the smoothing step, the sur-

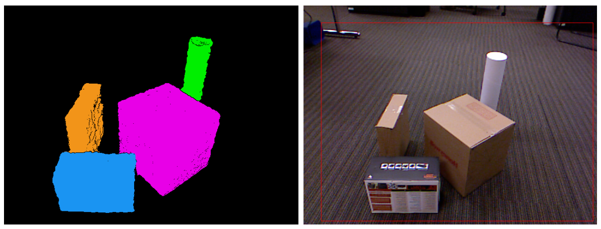
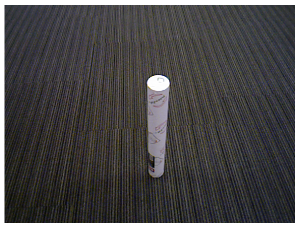
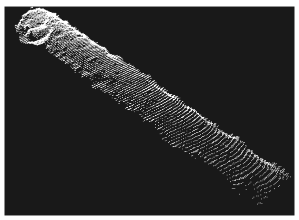


Figure 2: Result of Euclidean clustering.


(a) Test cylinder


(b) Before MLS

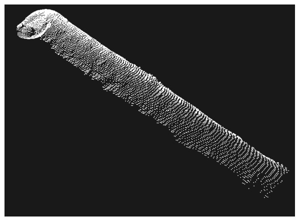

(c) After MLS

Figure 3: Before and after images of applying moving least squares to a point cloud of a cylindrical mailing tube.

face appears rough close-up with abrupt changes between points. These abrupt changes cause erratic surface normals and inaccurate curvature values. Figure 3 shows a result of applying MLS.

MLS adjusts each measured point according to a locally fit polynomial surface. For each point, the algorithm finds other points in its neighborhood. MLS defines a local coordinate frame at the anchor point. It defines the frame by first finding a plane of best fit through the points in the neighborhood. The plane's normal provides one basis vector, and the other two basis vectors are coincident with the plane. The origin of the frame is at the anchor point. The algorithm transforms the points in the neighborhood to the local frame. It then computes an approximating bivariate polynomial of best fit through the neighborhood in a least squares sense. If the anchor point is at $(0, 0, 0)$ and the polynomial is f, then its new position is at $(0, 0, f(0, 0))$. A transformation maps the smoothed point back to the global frame. MLS is a slow part of our object recognition scheme because finding the neighbors of each anchor point (via an octree) is time-consuming. The run-time for MLS depends on the quantity of points on the package, and it is generally 10% to 35% of the total run-time for object recognition.

3.5. Building a Feature Histogram

The feature histograms we build are based on two features at each point over an object and then combine them into a histogram. The first feature is the angle, θ, between the local surface normal and the ground's normal. Let the estimated ground's normal be $\vec{g}$, and let the local surface normal estimated at an anchor point be $\vec{n}$. To determine $\vec{n}$, we build a covariance matrix for the XYZ positions of the anchor point and points in its neighborhood of a certain radius. Suppose the center of gravity for the local region is $\vec{m} = (\bar{x}, \bar{y}, \bar{z})$. Define the 4x4 transformation matrix T as:

$$T = \begin{bmatrix} I_3 & -\vec{m} \\ \vec{0}^T & 1 \end{bmatrix} \tag{1}$$

Let A be a homogeneous $4 \times n$ matrix of the following form with one point from the region per column, where n is the number of points in the region:

$$A = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \\ y_0 & y_1 & \cdots & y_{n-1} \\ z_0 & z_1 & \cdots & z_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{2}$$

Set B to the non-homogeneous form of TA. Then, the 3x3 covariance matrix Σ is:

$$\Sigma = \frac{BB^T}{n} \tag{3}$$

Next, we run PCA on Σ. The eigenvector associated with the smallest eigenvalue provides an estimate for the surface normal at the anchor point. That is, $\vec{n} = \vec{v_1}$, where $\vec{v_1}$ is associated with the smallest eigenvalue λ_1 of Σ among the three eigenvalues.

We need to ensure all normals are consistent and point towards the optical axis. Initially, each surface normal could point in one of two directions, and a normal could wrongly point into a surface rather than up from the surface. Let $\vec{u} \in \mathrm{R}^3$ be a real-world point described in the camera's frame, with origin at the focal point. Let $\vec{n} \in \mathrm{R}^3$ be a surface normal at $\vec{u}$. Then, $\vec{n}$ must be flipped if

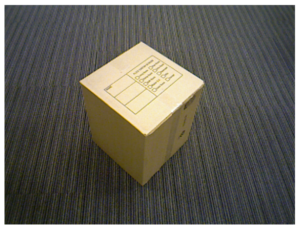
$$cos(\phi) = \frac{-\vec{u} \cdot \vec{n}}{\|\vec{u}\| \cdot \|\vec{n}\|} < 0 \tag{4}$$

Correspondingly, let $\vec{n} = -\vec{n}$ if $-\vec{u} \cdot \vec{n} < 0$.

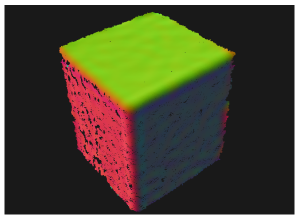
Finally, the first feature, orientation w.r.t. ground, is found via:

$$\theta = cos^{-1}\left(\frac{\vec{g} \cdot \vec{n}}{\|\vec{g}\| \cdot \|\vec{n}\|}\right) \tag{5}$$

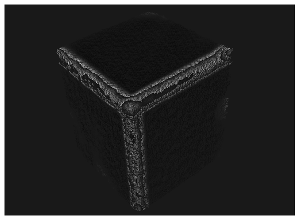
The second feature is curvature. From the previous PCA on Σ, we define curvature as the ratio of the smallest eigenvalue to the sum of the eigenvalues. That is,



(a) Test box



(b) Surface normals mapped to RGB space



(c) Curvature (brighter regions mean higher curvature)

Figure 4: Surface normals and curvature for a box.

$$c = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{6}$$

We use PCL's routines for computing the local surface normal and curvature. Refer to Figure 4 for a visual representation of surface normals and curvature.

We compute the two features at each of the object's points and put the features in a 2D relative histogram. We use 10 uniform bins for curvature, ranging from 0 to 0.08. We use 20 uniform bins for θ, which varies from 0 to π radians. Figure 5 shows typical histograms for the classes of packages. Note the similarities between histograms in the same class.

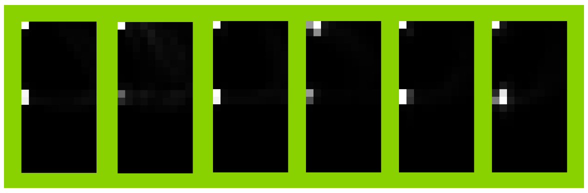
3.6. Top Shape Categorization

The module that classifies the top shape of the package projects its points to the ground plane. The projected points form a binary "projection image," where a non-zero pixel corresponds to a projected point. Each projected point forms a circle of radius 2 pixels on the projection image. We use a relatively large radius to fill in gaps between projected points. In the projection image, 1 pixel = 1 mm^2.

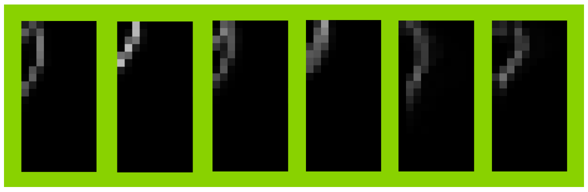
We first test if the package's top resembles an equilateral triangle. We find the largest contour among the clusters of non-zero points in the projection image. We approximate the contour with a polygon. The polygon needs three vertices to continue. The three interior angles are $\angle A$, $\angle B$, $\angle C$. Define the triangle score as:

$$t = |\pi/3 - \angle A| + |\pi/3 - \angle B| + |\pi/3 - \angle C| \tag{7}$$

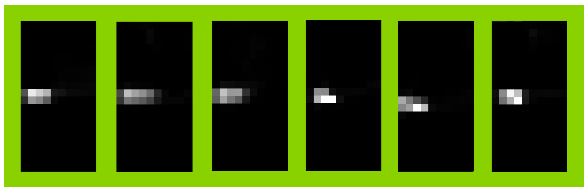
If $t < 1$, and package $height > 3 \cdot \max(length, width)$, then return "equilateral triangle." In essence, the top shape of the package must be a triangle with interior angles near $60°$ and the package must be tall in order for the top shape to be an equilateral triangle.



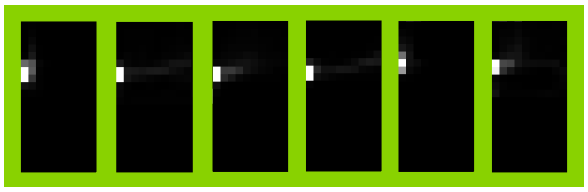
(a) Boxes



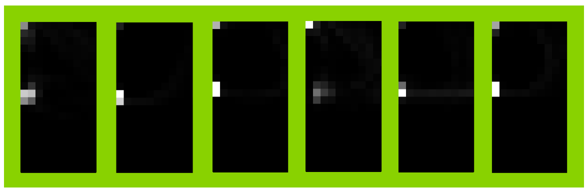
(b) Flat cylinders



(c) Vertical cylinders



(d) Flat prisms



(e) Vertical prisms

Figure 5: Relative histogram features for classes of packages with six examples per class. θ is on the vertical axis, and c is on the horizontal axis. (0, 0) is in the upper-left corner of each histogram. Brighter cells correspond to higher frequencies.

If the package fails the test for an equilateral triangle as the top shape, then it tests for a circle. Let n be the number of non-zero pixels in the projection image. Fit a minimum area bounding rectangle around the projected points, and suppose the bounding rectangle has dimensions w and h. Define the rectangle score as the proportion of the bounding rectangle's area that contains points:

$$r = \frac{n}{wh} \tag{8}$$

If $r < 0.88$, and package $height > 3{\cdot}\max(length, width)$, then return "circle." Otherwise, return "rectangle."

3.7. Training and Testing a SVM

We use libsvm for training a support vector machine to perform package classification [2]. We train a 5-class SVM, using the 1-versus-1 approach to training a SVM with more than two classes. A binary model exists between every possible pair of classes for a total of 10 models. We use equal weights for all classes. Additionally, we use the histogram intersection kernel, which we add to libsvm's source code. The histogram intersection kernel has this form, where $\vec{u}, \vec{v} \in \mathrm{R}^n$ are feature vectors:

$$k(\vec{u}, \vec{v}) = \sum_{i=1}^{n} min(u_i, v_i) \tag{9}$$

Our feature vectors are 201-dimensional, corresponding to the number of cells in a histogram plus the descriptor for the top-down package shape. We scale the feature vectors, which include the categorical descriptor of a package's top shape, so each cell is in $[0, 1]$.

Given a point cloud containing test package(s), the computer follows the above steps to build a feature vector for each package. The machine then uses libsvm and the trained support vector machine to classify the new package. The class with the most votes when classifying the package with the 10 models serves as a prediction of the test package's class.

4. Experimental Design

Figure 6 shows a RGB image from the PrimeSense Carmine 1.082 device with example outputs from our object recognition module. Beyond this anectdotal example, this section describes the experimental setup used to evaluate both the object recognition performance and its impact on the accuracy of the overall system.

To create a package shape recognition module, we train a SVM and test it using the previously mentioned method. We grab images from PrimeSense Carmine 1.08x devices. For our large-scale testing, each range image contains exactly one package with limited clutter. A planar floor consumes most of the image around the package. We find the largest plane in the image, which serves as the ground. We then iteratively compute a plane of best fit for the ground using least squares. Although each image has one package, we run Euclidean clustering to extract a complete mask for the package and ignore any background clutter. After isolating a package, we build a 201-D feature vector for it.

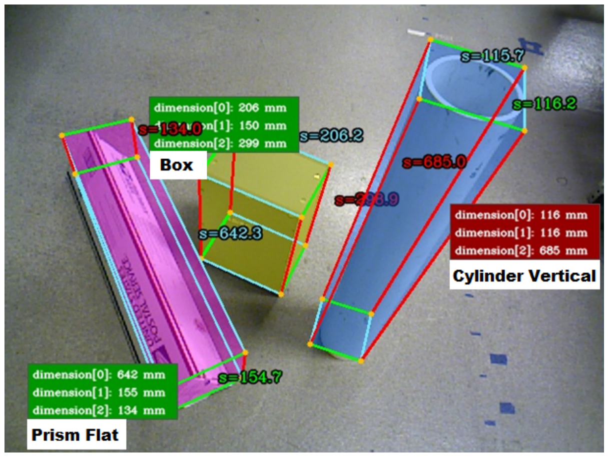


Figure 6: Packages displayed with measurements and labels.

For both training and testing, we capture diverse packages from a wide variety of viewpoints. The objects are diverse in scale and shape. Some boxes are cardboard, while others are machined from plastic. We use cylinders of various radii and heights. The packages and ground have diverse rotations relative to the camera. The camera is also at various distances from the packages (between about 0.7 and 2.5 m).

4.1. Training

We train a SVM with 3,427 examples of real packages. Table 3 gives counts of the objects used. A large number of cases are boxes because we had gathered range images of boxes for other analysis. During training, we discover an optimal cost of C=32.

Class	**Count**
Boxes	2654
Cylinders Flat	432
Cylinders Vertical	144
Prisms Flat	107
Prisms Vertical	90

Table 3: Counts of objects used in training

4.2. Testing Classification Accuracy

We test the 5-class SVM with a set of 1,487 packages. All these images are new and do not appear in training. For each package, we create a 201-dimensional vector and classify it using the trained SVM. Classification accuracy of this component test is presented in Section 5.

4.3. Testing Dimensioning Accuracy

Because the purpose of our object recognition module is to improve the performance of object metrology, it is more important to determine how object recognition impacts the accuracy of measuring objects. Our dimensioning system fits a minimum volume bounding box (or MVBB) around a package. This bounding box is the smallest box that contains all the package's 3D points, after suitable outlier rejection for stray points. The bottom face of the MVBB is always coincident with the ground, which reduces the number of degrees of freedom of the box when creating it. When a cylinder or prism is self-occluded, the naive dimensioning algorithm is unaware of the hidden points on the package and often initially underestimates the dimensions of the MVBB. However, we can adjust the dimensioning method after determining the package type. If the system labels a package as a flat cylinder or flat prism, then it adjusts the dimensions of the MVBB to better fit the object. If the system discovers that a package is a vertical cylinder or vertical prism, it prompts the user to lay the package flat so it can dimension the object more accurately. We use 474 packages having groundtruth dimensions of MVBBs–a subset of the 1,487 packages–for testing dimensioning accuracy.

5. Results

Classification is accurate. The correct classification rate is 99.06% for the test set of 1,487 packages. Table 4 gives a confusion matrix. Most of the confusion arises from the system incorrectly classifying flat cylinders and flat prisms as boxes. This confusion could be due to the fact that these three classes have a rectangular top shape.

Classifying and measuring packages is fast. Run time depends on the number of points in a package and thus the package size. Our program is single-threaded. We test the run time on a desktop PC with two Intel Xeon quad-core processors (each core runs at 2.93 GHz) and 12 GB of main memory. The object recognition module typically takes 0.16 to 0.29 seconds. The entire measuring process usually takes 0.6 to 1.1 seconds.

Package shape recognition improves dimensioning accuracy overall. Figure 7 gives relative histograms of the dimensioning error with and without object recognition. The system misclassifies one box, which leads to a negligible drop in overall dimensioning accuracy for boxes. Measurement errors for cylinders are overall smaller after object recognition. After recognizing vertical cylinders, we exclude the estimates because such detections would induce the user to re-position the object. Identifying flat cylinders allows us to consider the often hidden lateral surface when dimensioning. Measurements for prisms show much more significant improvement. Prior to object recognition, the dimensioner does not account for the hidden surfaces of a prism, and it often underestimates the size of a bounding box. When the system identifies a flat prism, it exploits symmetry and obtains a more accurate MVBB. We ignore vertical prisms upon recognizing them. Table 5 shows the mean errors for boxes, cylinders, and prisms with and without object recognition.

Class	**Mean abs error before recognition**	**Mean abs error after recognition**
Cylinders	8.14	2.26
Prisms	21.57	6.14
Boxes	4.31	4.53

Table 5: Mean absolute dimensioning error (in mm) before and after using object recognition

6. Conclusions, Limitations, Future Work

We presented a 3D object recognition method which improves the accuracy of dimensions estimated from mailing packages. With respect to our key criteria - classification accuracy and speed - we have demonstrated the utility of SVM-based classification of a histogram feature containing the orientation and curvature of surface points, along with a categorical feature describing the contour of the object's projection. Relative to previous work on 3D object recognition, our method is faster and is tailored to the particular application domain. We seek the smallest combination of features helpful in classifying objects in our application.

The method we present for object recognition works well in the limited domain of shipped packages, where only a few classes of basic geometric shapes are needed. Whereas our method may not extend to more general categories of 3D objects, other methods may be more useful for applications involving diverse object types. In the future, we could incrementally improve the recognition accuracy by updating the SVM given new data on observed packages.

We still need to determine how camera-object distance affects recognition accuracy. We notice that histograms appear blurrier as the distance between the camera and package increases. A greater distance results in an object having fewer points, and each point has a neighborhood with fewer neighbors. However, our intended operating range in commercial environments is small (between about 0.75 and 2.5 m), and we obtain high recognition accuracy with cases in this range.

Acknowledgements

KINECT is a registered trademark of Microsoft Corporation. XEON is a registered trademark of Intel Corporation. All other trademarks used herein are the trademarks of their respective owners.

References

[1] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and*

		Predicted Label				
		Box	F. Cyl.	V. Cyl.	F. Prism	V. Prism
Groundtruth Class	Box	390	1	0	0	0
	F. Cyl.	7	317	0	0	0
	V. Cyl.	1	0	223	0	0
	F. Prism	5	0	0	303	0
	V. Prism	0	0	0	0	240

Table 4: Confusion matrix from testing a 5-class SVM on 1,487 packages.

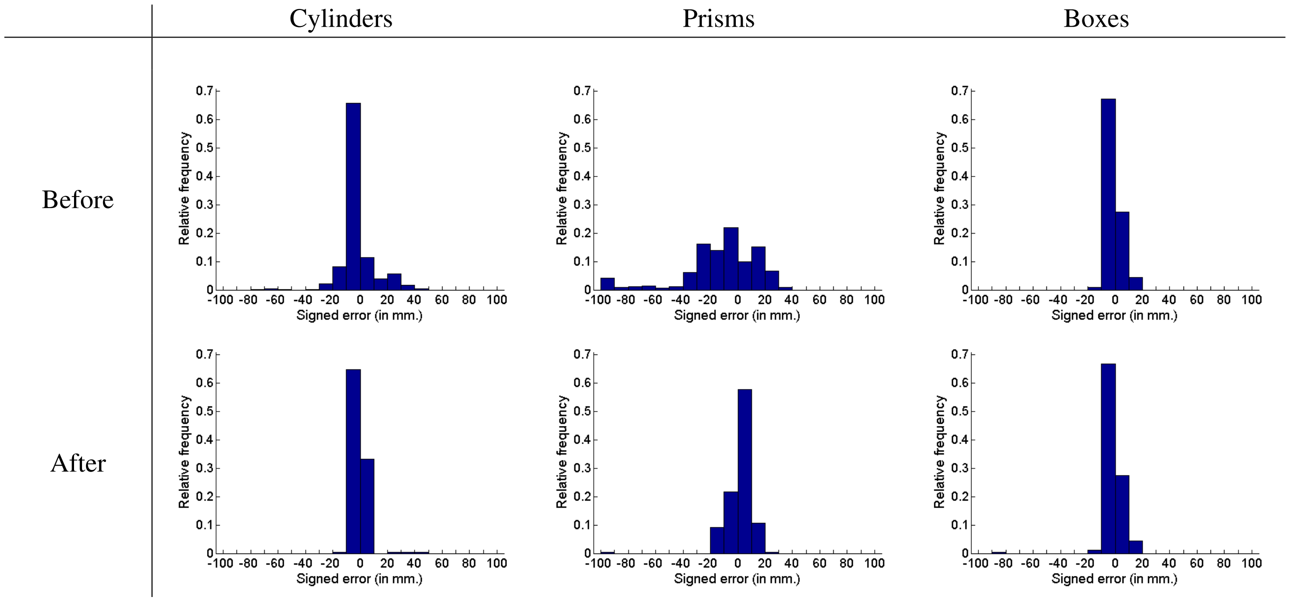


Figure 7: Relative histograms of dimensioning error, organized by package shape and before and after using object recognition.

Machine Intelligence, IEEE Transactions on, 24(4):509–522, 2002. 2

[2] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 6

[3] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *Neural Networks, IEEE Transactions on*, 10(5):1055–1064, 1999. 2

[4] Martial Hebert and Eric Krotkov. 3d measurements from imaging laser radars: how good are they? *Image and Vision Computing*, 10(3):170–178, 1992. 2

[5] Günter Hetzel, Bastian Leibe, Paul Levi, and Bernt Schiele. 3d object recognition from range images using local feature histograms. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–394. IEEE, 2001. 2

[6] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999. 2

[7] Jan J Koenderink and Andrea J van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–564, 1992. 2

[8] Marcel Körtgen, Gil-Joo Park, Marcin Novotni, and Reinhard Klein. 3d shape matching with 3d shape contexts. In *The 7th central European seminar on computer graphics*, volume 3, pages 5–17, 2003. 2

[9] Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of computation*, 37(155):141–158, 1981. 3

[10] David Levin. Mesh-independent surface interpolation. *Geometric modeling for scientific visualization*, 3:37–49, 2003. 3

[11] Danny Roobaert and Marc M Van Hulle. View-based 3d object recognition with support vector machines. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, pages 77–84. IEEE, 1999. 2

[12] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 2

[13] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010. 2

[14] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011. 3