Object Recognition and Classification from 3D Point Clouds

Antoine El Daher Sehyuk Park

Abstract: in most modern robotic applications, having accurate object recognition and classification is becoming an ubiquitous requirement. To deal with the latter, several techniques have been proposed, using either monocular computer vision, stereo vision, and even laser arrays. Recently, new laser devices introduced into the market, are able to capture the whole environment as a sequence of several hundreds of thousands of points per second, sampled from the environment, along a cylinder around the laser device. Essentially, with correct pose estimation, those point clouds give us exactly 3D point coordinates for a very fine-grain sampling of the scene around us. In this paper, we present two techniques for recognizing objects based on those point clouds. The first one is based on using support vector machines, and extracting features from the scene; the second is based on boosting a set of weak decision tree classifiers to recognize the object. We also present a framework for performing accurate simulations of the laser output.

I. Introduction and Previous Work:

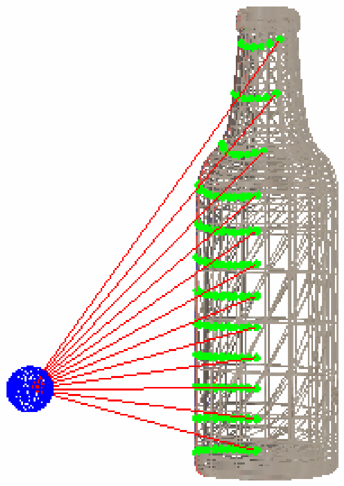
Several techniques have been used to recognize objects using computer vision. Currently, Viola-Jones advertise the most successful ones, which are based on making a boosted cascade of simple Haar-like feature classifiers [1],[2]. There are successful attempts at recognizing objects from images using SVM, as described in [3],[4] and [5], using typical Mercer or non-Mercer kernels. All those methods have proven to be highly accurate for simple 2D images, and under certain conditions. But the accuracy given by computer vision is not remotely sufficient for actual robotics. For example, a car detection module that used boosted classifiers, would always return several false positives, and several false negatives. And even if an object was detected using the camera, then it would be even more difficult to exactly pinpoint the object to 3D space, and find its bounds correctly. Many factors contribute to this inaccuracy; suffice to say that lighting conditions, colors, radial distortion, all affect the quality of the reading, and hence of the recognition. The idea behind lasers is that they are extremely accurate, and rarely off by more than 1mm. They emit their own light, so do not suffer from lighting fluctuations. We expect points coming out of a laser detection to be precise. Also, they are correctly scaled, which is different for the camera, since a same object can appear in different sizes for the latter, depending on its distance. There has been some work on recognition from range images [6],[7]. [8] describes a more recent approach, which uses the laser data in conjunction with camera data to "normalize" the camera data, in order to account for lighting changes. In this paper, we will describe our implemented simulator, as well as some of the techniques that we experimented with. For the simple case of testing whether an input object is a bottle or something else, we were able to get a classifier accuracy of 95.83% using 600 examples.

II. Simulating Readings:

Because such lasers are mostly at the prototype level, and still prohibitively expensive, we decided to design a simulator that would behave like the laser would. So we would feed in a scene of the environment, and then the simulator would give readings, as it would see them from its own point of view, if it were a set of 64 spinning lasers, along the y-axis.

The code for this was done mostly using OpenGL. We implemented a tool that would render scenes. We also implemented a laser simulator, which would spin around, and sample the scene at discrete intervals, and mark those points. We then ran the simulator on several 3D models of bottles, and recorded the outputs given by that simulator.

Because we only had around 10 models of bottles at our disposal, we recorded those same models in several other poses, sometimes applying some distortion in 3D space to them, in order to get more general results. We ended up with a few hundred 3D models of bottles, measured by 3D point clouds.



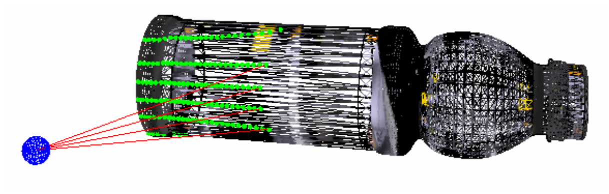


Figure 1: The simulator running on certain bottles.

The results given by the simulator were very satisfactory. The interface has also been made such that one can navigate through the scene, look around, and one can easily distort the points read out by the simulator, store them in file, and test classification on them based on the reading.

III. SVM for recognition

We have done a lot of work at the implementation level on SVM object recognition. Using SVMs, we tried to solve multi-class classification problem. 3D features (x,y,z) are not well-known to the image classification problems but there are many well-known methods for 2D images.

First of all, there are several ways of gathering features from 3D images. We devised a simple way of gathering features that is applicable to general learning algorithms easily.

As preprocessing, we shift the 3D points so that the mean of z value becomes 0. Simply, we can subtract mean z value from every point's z value. Using this way, we can also rescale every object to have similar sizes, basically by dividing every coordinate by the maximal coordinate that has been read throughout the reading.

Based directly on the range scan, we divided the 3D image into N by M regions and for each region, computed the maximum, minimum and mean z value among the points in the region. For example, taking into account a 3D bottle similar to the one before:

Similarly, we could compute minimum and average z value for each region. The three measures (max, min and mean) are a very good way to explain the features briefly.

Then, we have N*M*3 dimensional features. Using the features of the training examples, we can train an SVM model and predict on test examples. This is a first very simple approach at the problem.

We experimented with various kinds of kernels, but found that the most accurate results came when we used the radial basis kernel:

$$K(u,v) = \exp\left(-\frac{\|u-v\|^2}{2\sigma^2}\right)$$

The following picture shows the information that we'd extract out of such an image.

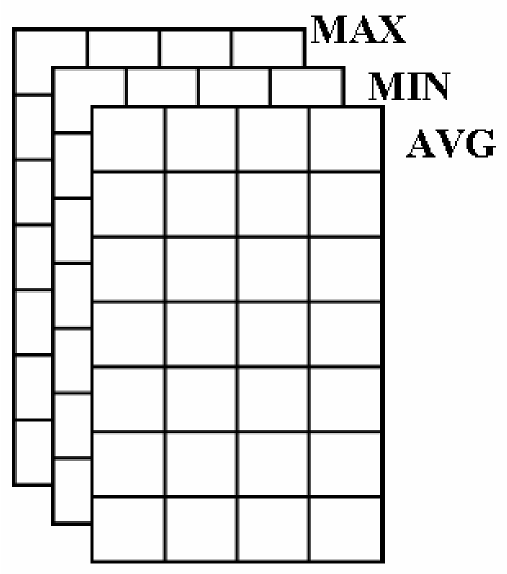


Figure 2: the max,min and mean features

And here is a an example that shows exactly what is currently going on with the inputs.

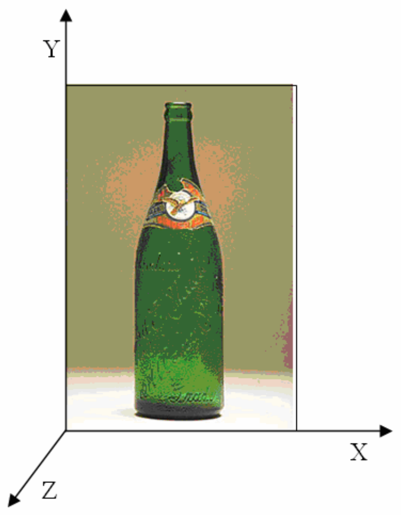
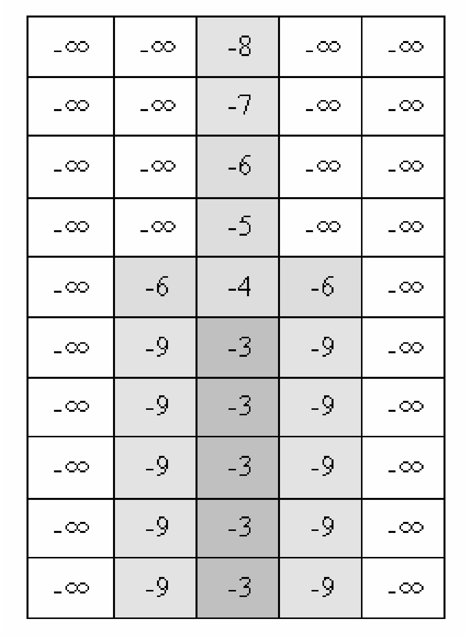


Figure 3: Sample bottle

And assuming that the camera reader is somewhere on the XZ plane, away from the YX plane, we get the following readings:



-∞	-∞	-8	-∞	-∞
-∞	-∞	-7	-∞	-∞
-∞	-∞	-6	-∞	-∞
-∞	-∞	-5	-∞	-∞
-∞	-6	-4	-6	-∞
-∞	-9	-3	-9	-∞
-∞	-9	-3	-9	-∞
-∞	-9	-3	-9	-∞
-∞	-9	-3	-9	-∞
-∞	-9	-3	-9	-∞

Figure 4: Depth map for the bottle

In this image, each reading is a depth reading, as given by the laser scanner.

In order to solve the scaling problem, we devise a hierarchical model, similarly to what is described in [9]. In other words, we gather features from M x N regions, (M/2) x (N/2), (M/4) x (N/4) and so on. In other words, we divide the image into several sub-samples, and get features from that.

The question one needs to ask oneself at this point is: how big should M and N be? Bigger regions can solve the locally translation invariant problem but will lose much information. On the other hand, small regions can contain more information and capture more important features. The latter might nevertheless be affected by small translations in the input.

Note that all of the above mostly refers to depth images, as opposed to actual (x,y,z) coordinate images. We will be working on extending the algorithm to take all of this into account; the scaling problem would disappear under those conditions, since we know that a bottle can only be of a specific size.

Thanks to Chih-Chung Chang and Chih-Jen Lin, we used some parts of the code LIBSVM (A Library for Support Vector Machines).

The experimentation procedure is described in the section about results.

IV. Boosted classifiers for recognition:

i. Technique:

In the spirit of what was done by Viola-Jones [2], and Nuchter [8], we thought about using boosted weak learners to classify 3D objects. We did not really do any implementation on this part, but gathered most of the ideas already.

For purely depth images, the idea is to use the same set of features as those found in 2D images classification, namely simple Haar wavelets; then one can run AdaBoost to combine those in order to correctly be able to classify the input. Note that the input would then simply be the depth reading.

However, this would basically be forcing the object to be recognized from this particular point of view, but we have much more information from the 3D points and the pose of the laser, than we have from depth information.

For this reason, it seems natural to extend the Haar wavelets typically used in 2D computer vision, to take depth into account; the basis is too big to fully list, here are a couple of vectors in the basis:

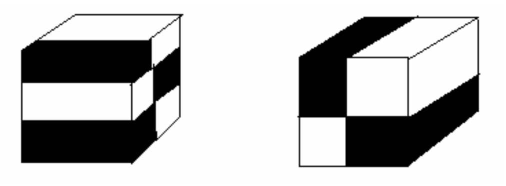


Figure 5: 3D Haar wavelets.

So given a particular input, whenever a point that is read falls within the "black" area, it is added to the sum, and whenever it falls within the "white" area it is subtracted, similarly to what the Haar features do.

The idea is that we have several such features, and so in the test cases, we find the feature which maximizes the information gain. Then we re-assign weights for the whole sample space using the standard AdaBoost technique, and find the next feature which maximizes information gain, and so on. In the end, we combine all of them to create a classifier.

Next, given a test image, all we need to do is evaluate that classifier for those images.

ii. Reducing the run-time:

To speed things up during training, as was done using the standard 2D Haar, we can use an "integral 3D image", which stores for any [x,y,z], the total number S[x,y,z] of points contained in the box [0..x] x [0..y] x [0..z], so that the number of points contained in [a..b] x [c..d] x [e..f] can be calculated using O(1) accesses to the [x,y,z] table.

Computing the [x,y,z] table can be made efficiently. We used the following formula:

$$\begin{aligned} S[x,y,z] = {} & S[x-1,y,z] + S[x,y-1,z] + S[x,y,z-1] \\ & - S[x-1,y-1,z] – S[x,y-1,z-1] – S[x-1,y,z-1] \\ & + S[x-1,y-1,z-1] + P[x,y,z] \end{aligned}$$

Where P[x,y,z] = 1 if there is a point inside the bin [x,y,z] and 0 otherwise. This clearly allows us to find the whole of S[x,y,z] in O(n^3) time.

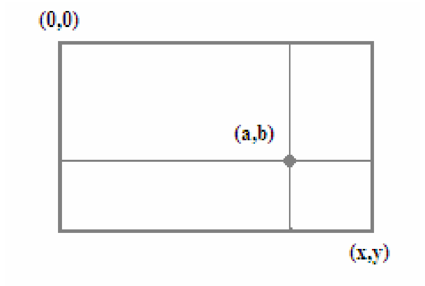


Figure 6: Finding S[x,y] in O(1)

Looking at the above table, which is a 2D simplification of the problem, we notice that the sum of the points in the range [a; x] x [b; y] is simply:

$$S[x,y] – S[a,y] – S[x,b] + S[a,b]$$

Extending this to the case of a 3D feature, it is easy to see that the sum of pixels in the range [a..x] x [b..y] x [c..z] is:

$$S[x,y,z] – S[a,y,z] – S[x,b,z] – S[x,y,c] + S[a,b,z] + S[a,y,c] + S[x,b,c] – S[a,b,c]$$

With that in mind, we can evaluate each feature at a specific location, and get a certain value for it.

iii. Training:

Training occurs as follows: for each possible feature, try placing the feature at every single location in the image grid. Evaluate the feature for all of the positive samples, and all of the negative samples. Use the feature as a classifier, and find the threshold which maximizes the information gain. Then find the feature which maximizes the information gain, which is simply given as:

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)-\frac{p_0+n_0}{p+n}I\left(\frac{p_0}{p_0+n_0},\frac{n_0}{p_0+n_0}\right)-\frac{p_1+n_1}{p+n}I\left(\frac{p_1}{p_1+n_1},\frac{n_1}{p_1+n_1}\right)$$

In the above, n_0 is the number of negative examples that would make the value bigger than the threshold, p_1 is the number of positive examples whose feature evaluation would yield a value smaller than the threshold.

So in the end, the problem is simply one that counts take into account the density of points, and their configuration, to be able to make accurate predictions. The training time for this method was understandably very slow, because we're forced to go over several combinations. For 4 kinds of features, and different kinds of scaling in (x,y,z) there are approximately O(n^6) combinations, each of which needs to be tested on the samples which number possibly as much as a thousand. In addition, storage for the sum tables tend to be also very large, occupying a cubic amount of memory for each of the test samples.

This made it difficult to train on a very large set, and so we had to restrict ourselves to sets with sizes as small as 20 elements, and test sets of size 20 elements. This only yielded accuracies of 70-75%. With more computing power, we could have trained it on much larger sets and seen what the results would look like, which is something that we have included in our future work.

iv. Accounting for various transformations:

Some factors need to be taken into account in order to make sure that objects are correctly recognized. First of all, the range readings are all converted to position readings in x,y,z; this makes sure that objects are correctly scaled, and that an object nearby won't look ten times bigger as it would in a camera.

Second, when an object needs to be classified, the mean depth is subtracted, so that the object now has a depth centered at z = 0. This allows for some translation issues to be resolved.

V. Results

Let us first describe the testing scenario that we used.

Because the results of the SVM algorithm were the most successful, we will report them here. The previous section describes the results of the boosting method. We generated 300 models of bottles, and 300 models of non-bottles, which included various random objects, such as cylinders, spheres, distorted planes, random sets of points; we then ran it with various cross validation coefficients, for example, with 70% of the samples used for training, and 30% of the data used for testing.

The initial accuracy using 60%-40% validation was 95.83%, with 230 out of 240 examples being classified correctly. The number of support vectors on the 360 strong training set was 205 on average.

For various other distributions of the cross validation percentages, the accuracy varied between 93% and 96%.

To make more evident the effect of the sample training size, we ran it on several different sample sizes, which we increased progressively.

Here are the results for 70-30 cross validation:

Training Size	SV	Test Accuracy
12	7	75%
24	13	75%
36	24	81.81%
48	31	80%
60	35	87.5%
120	61	91.67%
240	90	91.67%
360	139	93.33%
480	201	93.06%

Table 1: SVM testing results

The number SV represents the number of support vectors that were used in the final classified. Here is the corresponding learning curve.

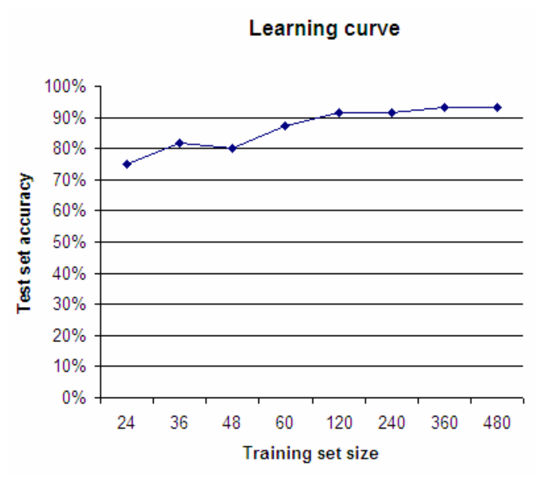


Figure 7: SVM learning curve

VI. Conclusions and Future Work:

The results given by SVM were quite accurate (up to 95%), and we are convinced that given more time and computing power, the results given by boosting weak classifiers would also have been far more accurate.

There were some problems with the training that we'd like to address in the future. The first one was obviously that the bottle samples are quite different from the non bottle samples, which made the problem "easy" to classify, even though we tried having some similar shapes, such as cylinders in the non-bottles.

However, because everything was run of the simulator we did not have an infinite amount of data, and the inputted negative samples would probably not work if run on an actual data. An actual device would have given far more representative results. Either way, the methods are solid and fast at runtime at this theoretical level.

References:

[1] Paul Viola, Michael Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features" (2004)
[2] Viola, Jones, "Robust Real-time Object Detection"
[3] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines—a kernel approach. In *Proc. of the 8th IWFHR*, pages 49–54, 2002.
[4] Sabri Boughorbel; Jean-Philippe Tarel **"**Non-Mercer Kernels for SVM Object Recognition**"**
[5] Chikahito Nakajima Norihiko Itoh "Object Recognition and Detection by a Combination of Support Vector Machine and Rotation Invariant Phase Only Correlation"
[6] Farshid Arman, "Model-Based Object Recognition in Dense-Range Images — A Review"
[7] Paul J. Besl and Ramesh C. Jain, "Three-Dimensional Object Recognition**"**
[8] Andreas Nuchter, Hartmut Surmann, Joachim Hertzberg **"**Automatic Classification of Objects in 3D Laser Range Scans**"**
[9] J. Mutch and D. Lowe. "Multiclass Object Recognition with Sparse, Localized Features" In: Proc. IEEE Conf. Comp. Vision Patt. Recog. 2006
[10] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in Proc. CVPR, 2006.
[11] Kristen Grauman, Trevor Darrell, Unsupervised Learning of Categories from Sets of Partially Matching Image Features, Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, 2006.