Semi-supervised learning for adaptive object recognition in RGBz images

Andrew Duchi
Stanford University
aduchi@stanford.edu

Abstract

In this project I focus on the problem of using depth data to improve performance of object classification on RGB and depth images on-the-fly, with no depth data in our training set. I used both retraining of models with high confidence images as well as outlier detection using both distance-based and probabilistic metrics in hopes of increasing performance. Outlier detection over the depth data offered no improvement in performance, but training a new model to incorporate depth data using only high-confidence classifications as ground truth provided modest improvement over the RGB-only model. The results also seem to indicate that improvement can be achieved for classification systems with F1-scores on a given dataset down to .75 and possibly below, giving between a .02 improvement and .1 improvement in F1-score when the model was retrained. It also seems that lower scores leave room for more improvement and can benefit more from inclusion of depth data than classifiers that are already functioning well. Next steps should include creation of an additive model rather having to discard the entire trained RGB model when we retrain.

Future Distribution Permission

1. Introduction

Since the release of the Microsoft Kinect in November of 2010, there has been an explosion in the use of depth data in visual object recognition. This change is because Kinect provided the first low-cost, high-quality depth sensor that made acquisition of reasonable quality depth data feasible in a wide range of settings. These sensors are especially useful in computer vision because, even when two images may have different colorations that cause confusion in a standard RGB image, the depth image will be unaffected and shape correspondences can be easily seen, as in Figure 1. Despite the utility of depth data, there is an unfortunately small amount of it easily available as most capture devices are still standard visual cameras and cheap depth sensors were only recently made available.

While collection of depth data is starting to ramp up, it still pales in comparison to the sheer volume of standard RGB image data. For almost any object type, one can recover a large corpus of weakly supervised images via a standard image search on any major internet search engine. RGB image detection schemes can leverage this data and, thus, be trained without a need to directly collect and capture new data for each new object type, but RGBz object detection schemes do not currently share this convenience.

It is the goal of this paper to investigate ways that we can leverage the large amount of labeled RGB image data and the depth sensing ability of technologies such as Kinect to improve object recognition without a need to directly collect new RGBz data. This could have applications in a variety of tasks, such as robotics. For example, imagine a robot with a depth sensor encountering a novel object type: the robot can use a search engine to generate weak images for that type and build an RGB classifier,[4] but cannot do supervised RGBz learning. The method I propose will allow it to use an RGB model to do initial classification of RGBz images and improve its

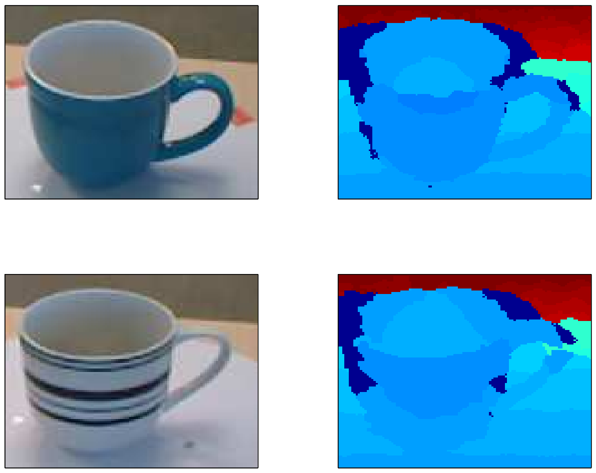
Figure 1. Examples of RGB and depth mug images

detection capabilities gradually over time with no need for supervision.

The general framework I provide is to learn an RGB object classifier. Use it to classify objects until such time as we have a set of high-confidence RGBz positive examples. Then train a new classifier using our collected data.

2. Background/Related Work

One of the major components required in this work is a visual descriptor for the objects. There are a number of good candidates that have seen success in the object classification task including HAAR wavelets [10], SIFT keypoints [8], simple Bag-of-Words models [2], and Histograms of Oriented Gradients (HOG) [3]. In this project, I use HOG features because they seem to offer superior performance over these other image features and are relatively straightforward to implement[3]. These features are well suited to object recognition because of how they capture object shape well due to their gradient information and are invariant to changes in brightness due to the normalization of local descriptors.

Another important component of this project was the use of depth features to include the depth information available on non-training data. Again, we decided to use HOG features as they have been shown to work on a range of data modalities aside from only RGB images. For example, HOG features have been used with both depth [7, 9] and infrared [11] image data to improve detection. Thus, HOG features also seem to fit as a good feature for extraction on the depth images.

For classification of images after feature extraction, we rely on the standard formulation of a Support Vector Machine (SVM). SVMs are commonly used as the machine learning classifier for object recognition tasks with HOG features and exhibited good performance on these tasks[3, 7, 12]. These classifiers assign objects a margin distance which, though not a normalized probability of correctness, gives a rough estimate of correctness likelihood with smaller margin distance implying lower classification confidence[5].

Given a classification based on the RGB information, our goal is to remove false positives using the test set depth information. This problem can be seen as one of outlier detection, where we want to remove “abnormal” instances from our set of positive classifications. There are two main (related) metrics for outlier detection and we investigate both: distance metrics and probabilistic detection.

In distance detection we generally assume that the true instances will be grouped “close together” in some sense, while false detections will be dispersed. One natural and common way to define this distance is the L2 distance between feature vectors. Given this, the metric used to define an “outlier” can take on a number of forms. Two standard approaches are either to look at which instance has the largest nearest neighbor distance, leveraging the fact that normal instance will likely be close to other normal instance while outliers will be dispersed. Another approach is to use the average distance between each point and all others, though this is more computationally intensive[1].

Another form of outlier detection relies on probabilistic modeling. In this case, it is common to model the data as coming from some multivariate distribution (such as a multivariate gaussian which, though not known to be correct for our data, is often used in practice) and use this to generate the probability density at each data point. The probability density can then be used to determine whether or not a point is an outliers[1]. Such a model can easily be learned in the Gaussian case using closed form maximum likelihood estimation [6].

3. Approach

3.1. Visual and Depth Descriptors

As stated above, I use HOG features both as visual descriptors and depth descriptors. I use the standard implementation of bins over 9 rectangular blocks of pixels, each overlapping with adjacent blocks by 1/2 its width or height, with 9 orientation bins per block. The one deviation I make from the standard implementation is using directional gradients rather than absolute magnitude as this has been noted to improve accuracy on non-human objects [3]. Rather than rescaling images to a fixed size and then applying the descriptor, I apply the descriptor at varied scales based on the image to avoid aliasing effects. This will not cause issues due to normalization of the descriptor blocks. Additionally, I experimented with grids of HOG descriptors over the image from a single HOG descriptor to a 2 by 2 grid and so forth.

In addition to the HOG descriptor, I experimented with using a raw depth mask that are simple rescalings of the image to a smaller resolution (ie. 5 by 5) as these will have shorter descriptors than the HOG feature and I was concerned about overfitting when using features on test data. To generate the descriptor, I first rescale the original depth image D^* down to a fixed size image D. Then, to avoid problems of objects at varied distance not matching I normalize by subtracting the central point from all others to generate D' as follows:

$$D'(x,y) = D(x,y) - D(\frac{W}{2}, \frac{H}{2})$$

I also experimented with using a weighted depth mask such that pixels far from the center (which are likely less relevant) were downweighted and would thus influence distance less.

$$\begin{aligned} D'(x,y) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-W/2)^2+(y-H/2)^2}{2\sigma^2}} \\ &\quad (D(x,y) - D(\frac{W}{2}, \frac{H}{2})) \end{aligned}$$

However, this did not impact performance so I treat it in unison with the other depth descriptor in my evaluation.

3.2. Outlier detection

In this document, I experimented with both distance-based and probabilistic outlier detection. For distance-based outlier detection I used a distance thresholding approach as follows:

1. Calculate the L2 distance $d(x, y)$ between the feature vectors of every pair of instances x and y.
2. Calculate a score $s(x)$ for each node x in the set of instances current classified as positives, P.
   - If we are using average thresholding, $s(x) = \frac{1}{|P|-1} \sum_{y \neq x} d(x, y)$
   - If we are using minimum thresholding, $s(x) = \min_{y \neq x \in P} d(x, y)$.
3. For all x such that $s(x) > thresh$, reclassify x as a negative.

In addition to this distance-based method, I tried a simple multivariate Gaussian approach. It is certainly true that the distribution of my features need not be a Gaussian, but this is a reasonable starting point and the Gaussian offers more flexibility (automatically downweighting high-variance elements) than a simple distance comparison. I attempted the following procedures for outlier identification with a multivariate Gaussian:

1. Fitting a Gaussian to the positive instances and removing all those below some threshold.

2. Fitting distributions to both positive and negative examples (one each) and removing all positive instances that had higher probability under the negative example distribution.

3. Iteratively fitting, removing examples below a threshold, and refitting as follows:

   (a) **M-step:** Fit our mean and covariance matrix for each class C_i

   - $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$
   - $\Sigma_i(j,k) = \frac{1}{|C_i|} \sum_{x \in C_i} (x^{(j)} - \mu_i^{(j)})(x^{(k)} - \mu_i^{(k)})$

   (b) **E-step:** determine class assignments C_i for each instance feature vector x. This is done based on probability thresholding (one class) or which class has a greater probability density at that point.

3.3. SVM learning and re-learning

In this paper, we used a simple out-of-the-box implementation of an SVM that was compatible with MATLAB (a derivative of SVM-Lite). I used the models suggested parameters of L2 regularization with an RBF kernel. The parameters could certainly have been optimized further, but there did not seem to be any need in this case as it functioned relatively well and was not he core of our investigation.

One important aspect of this project was the process of relearning an image classifier to include depth data as another option separate from outlier detection for performance improvement. This procedure continues as follows:

1. Train an SVM on our RGB training set

2. Classify the Test set based on its RGB values getting labels l for the Test set.

3. Discard all instances where the instance was classified as positive, but does not meet some threshold t. (ie. if instance i has $0 < l_i < t$, discard it)

4. Retrain our SVM on the RGB and depth data (RGB HOG features concatenated with depth HOG features) of Test instances that were not discarded.

5. Reclassify the entire Test set (including previously discarded instances) using the new RGBz model to get margin values (labels) l'.

6. Consider any instance j that has $l'_j > s$ for some threshold s.

We choose s and t via validation on a validation set before using them for the test set. The reason we choose a threshold t to determine which instances to discard is that the SVM margin gives us a rough indicator of classification confidence, so elements close to the margin have lower confidence than those farther away. The basic idea is that we can discard items that are close in the RGB space to prevent corruption of the model learned over the depth features and the depth features will be robust enough to let us correct previous mistakes. Now we note that we need the s threshold because we have removed the positive instances that created support vectors in the original RGB model, so the dividing plane should encroach on our higher confidence positive instances and we will have to "push it back" to recover the instances that were previously "harder" and thus had lower margin. This process will not just take us back to the original state because the inclusion of our depth features will disambiguate incorrect examples that were not obvious visually but were obvious based on depth and push them away from the margin and give them a negative labeling.

4. Experiments

4.1. Dataset and Performance Metric

In this project, I used the "Large-Scale Multi-View RGB-D Object Dataset" from Washington

University[7]. Unfortunately, the caching files required plus the size of the images required a great deal of size, so I was restricted to using only approx. 15000 of the images from this dataset and a subset of the object classes. I used the following object classes as example images: {apple, ball, banana, bell pepper, binder, bowl, calculator, camera, cap, cell phone, cereal box, coffee mug, comb, dry battery, flashlight, food bag, and food box}. Of these object types, I used the coffee mug as my classification objective due to its relative invariance to rotation (sans its handle and designs on the mug). Once this dataset was constructed, I split it into train, validation, and testing datasets with approximately 40%, 30% and 30% of the images in each. In addition to randomly splitting object instances across all 3, I held out 25% of coffee mug instances (ie. an instance being all pictures of the same real mug) for each of the validation and test set. I also held out 25% of object types for each set so that, for example, banana never appeared in the training set, but was in the validation set. This decision was made to better simulate the occurrence of novel objects in real-world data where the learning algorithm will not have access to examples of all object types as labeled data.

Another important starting point is that throughout this paper I use F1-score as my performance metric. The F1-score gives a succinct description of performance based on the precision and recall of our classification. This approach avoid the problem of data set bias (having an uneven positive/negative split – as is the case with this data, we have more negative images) influencing the accuracy score. Additionally, F1-score seems to be the standard score for use in retrieval tasks, as we are doing in this instance of trying to correctly retreive all coffee mugs from the data set.

4.2. Initial Performance and Feature Validation

As discussed above, I use HOG features over varied granularities as the visual descriptors for classification. I validated the granularity of my HOG feature grid in order to determine what feature setup to use for all subsequent classification. In doing this validation I both demonstrated that HOG features on visual and depth images provide good features for classification and identified what granularity was best. A plot of the results can be seen in Fig. 2.

At first glance we note that these features allowed us to achieve F1-scores over .9 for both training and validation sets, which is relatively good performance and demonstrates that our classifier is working well. These results show that the training performance on both RGB and depth images increases monotonically with feature complexity (a denser grid/larger feature vector) as does performance on depth images. This is expected as higher dimensionality allows our SVM more flexibility in fitting a separating hyper-plane in the training data. On the other hand, we see that performance of a model trained on training data for classifying the validation RGB images degrades after a grid density of 2. This indicates that beyond a grid size of 2, we are overfitting and adding noise to our model. Thus, we choose to use a HOG grid of 2 features by 2 features for visual features in the remainder of this project. Note that depth seems to increase up to a grid of 4, but we will not have ground truth depth training data, so we may not want to use such a dense grid for fear of overfitting on noisy data.

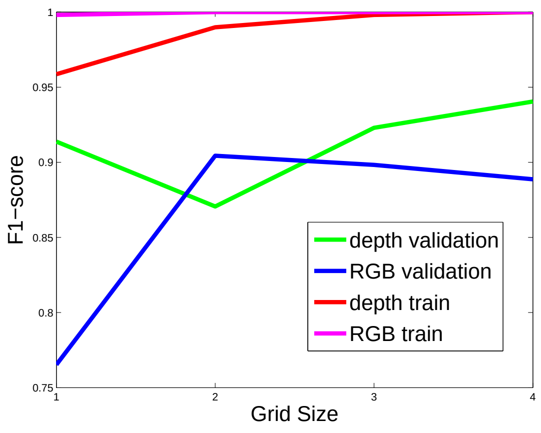

Figure 2. F1-score compared to feature granularity on validation and training data

4.3. Outlier Detection

In performing detection of the outliers, I attempted to use both distance-based and probabilistic methods for detecting outliers. In using the distance methods on the validation set, I found that across all instances, the F1-score was approximately .01 higher using average distance than it was using closest neighbor distance. This is not surprising as if we have two instances of the same false positive object, they may be very close though they are far from the majority of our positive objects. The minimum distance will not differentiate this from the distances for true positives near other true positives, but the average distance will see a high distance to the plurality of points.

In terms of improvement, the only improvement was using the smallest possible threshold that actually eliminated one instance that was labeled as positives, causing a bump in F1-score from .9044 to .9047. This is not a large enough difference to be considered anything more significant than a lucky change and indicates that these distance based methods do not seem to work. We can see in Figure 3 that the F1 score does not recover as we decrease the threshold to some distance, but in fact monotonically decreases as we decrease our distance threshold for removal (this specific instance is over depth HOG with a 4 by 4 grid). Thus, it appears that using this distance metric does not improve classification.

I also attempted to use fitting of a multivariate Gaussian distribution to the positively classified instances and use probabilistic thresholding. I thought this would likely provide better performance than the raw distance comparison as it would essentially weight each element of the feature vector based on its variance, so deviation in small variance elements would be penalized more than deviation in high variance elements – potentially giving a better measure of how "atypical" the values were. Unfortunately, as with the raw distance metric, I could not find a threshold that improved the F1-score. Additionally, attempting to fit a distribution to the negative data as well then comparing probability densities and removing all those that had higher density under the negative example distribution did not cause any changes. This was because the variance over the negative examples was dramatically higher, causing the probability density to be much wider than the positive distribution, so no previously positive examples had lower probability under the positive distribution than the negative distribution.

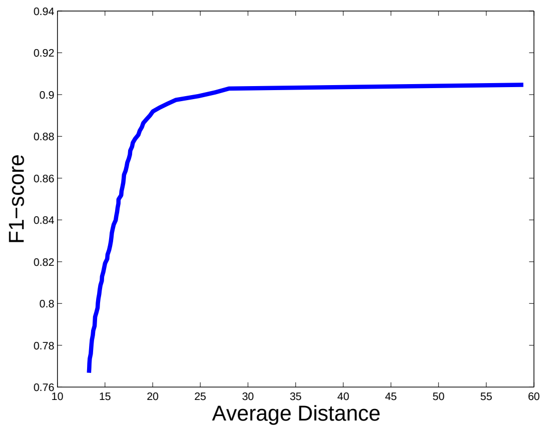

Figure 3. F1-score across different average distance thresholds for removal

Thus, these simple outlier detection methods over the depth information do not improve detection.

4.4. SVM Relearning

In doing relearning of a model over the RGB and depth data, I analyzed performance on the validation set to see if I could identify thresholding techniques that consistently improved performance. I analyzed results using a 2 by 2 HOG grid on the visual features and using grid sizes from one to four for the depth features. The best F1-scores I received for the various grid sizes can be seen in Figure 4.

We note that though our initial validation indicated a four by four grid gave the best performance on training data, we see that a two by

HOG grid	1 by 1	2 by 2	3 by 3	4 by 4
best F1-score	.9237	.9309	.8922	.8937

Figure 4. F1-score of the relearned for varied HOG grid sizes used for depth feature extraction

two grid performs best and then performance drops off. This was true across all threshold settings, not just the optimal setting for each. Such a drop off seems to indicate that taking a HOG grid of size larger than two by two causes overfitting that does not generalize well back to the low-confidence validation images.

While this does show an improved F1-score of .9309 as compared to .9044 for the original RGB model, we are just picking the best thresholds for which data to exclude in retraining and what margin boundary to use in final classification, so it may seem that this should not generalize well. In that vein, we note that the thresholds were quite consistent across all four grid sizes, with the optimal confidence window being the top 20% or top 30% of the margin distance above our margin. Additionally, all grids types achieved maximal F1-score when we shifted the margin toward the negative examples (decreased our threshold for positive classification) by 30% of the margin range (highest SVM score minus lowest SVM score). Finally, we note that the distribution over F1-scores across the threshold values grid (of both the confidence level for retraining and margin for final classification) was relatively single peaked and exhibited a strong global maximum for each grid size as seen in figure 5. This indicates that the parameters were displaying consistent behaviors as their values changed, so our improved performance is likely not just a "chance" good parameter setting.

4.5. Final Test Results

Given the performance on my validation data, I decided to use the SVM retraining approach with the two by two grid of visual HOG features concatenated with the two by two grid of depth HOG features as the feature vector for each picture. For

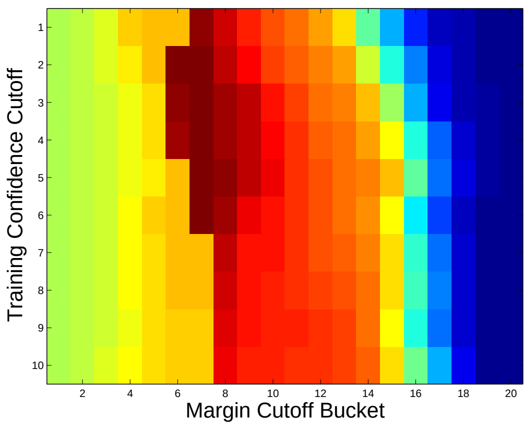

Figure 5. F1-score across different thresholds for retraining. Warmer colors indicate higher values

settings, I used a threshold of the top 30% of our positive confidence range (ie. if the maximum confidence was c, all values with margin distance over $.7c$ were used) as per the validation results. For final classification, I used a margin that was shifted 30% closer to the minimum margin value (as per the validation dat performance).

These settings gave a test F1-score of .7822 which was .0266 or 3.5% higher than the F1-score of .7556 given by the standard RGB model trained on our training data. This demonstrates that the proposed method can be used to improve classification (as measured by F1-score). Also note that .7822 was actually one of the lower F1-scores using depth data, and many of the other settings (different thresholds) achieved F1-scores of over .8, with the maximum achieving .8494 – but we cannot consider that as a true result in this experiment as it relies on non cross-validated parameters that were picked after testing.

The difference in F1-score between our validation and test sets is due to the fact that I used different unseen object classes and different mug instances (different real world coffee mugs) in the test, validation, and testing sets to keep the test set as decoupled as possible from our optimizations on the validation set. Thus, there just happened to be more difficult objects in the test dataset, causing this decrease in performance. Also, note that the improvement on the test dataset was equivalent to the validation set, but that other parameters would have given even greater improvement – perhaps indicating that when we have worse RGB performance the depth data can be even more valuable.

5. Conclusions and Future Work

This paper demonstrated that a classifier over RGB and depth data can be learned on-the-fly in test data for improved performance without a need for depth data in the training set. The performance increase is modest, but is certainly a step in the right direction and indicates that there may be large benefits to be had when we have only mediocre classification accuracy with the RGB classifier (F-score of .75 as in the test data as opposed to a solid .9 as in the validation data). I also noted that retraining of an SVM model based on high confidence images is an effective way to improve classification, while doing simple outlier detection is not terribly effective.

A great deal of future work is left for fully taking advantage of depth data in the training. One low hanging fruit is likely optimization of the threshold selection for retraining of the SVM and deciding a classification margin. I suspect that these margins are a function of the F1-score, which cannot be directly inferred, but by looking at the range of margin scores it may be possible to do better selection. This is especially true as the raw value of the confidence cutoff that gave optimal score for the test data was closer to the raw values of our thresholds than the scaled threshold value (by raw value, I mean the actually numerical margin as opposed to "30% of the observed maximum margin"). Doing more analysis of these thresholds would potentially allow better performance on test data.

Another change that should be made to the model is using a weighted combination model (similar to boosting) rather than simply throwing out the original RGB model. In this way, we could scale the weighting of the original RGB model and a learned RGBz model as we acquired more test data so that we could quickly start seeing improvements for any gross depth outliers with only a small number of test images, while still leveraging the RGB model for the vast majority of cases.

Similarly, I would like to analyze of the learned RGBz model with different numbers of test instances. It seems clear that performance should improve as we acquire more test data, but I did not how much data is needed before it starts providing improvement. In this study improvement was seen with down to approximately 100 high confidence classifications–many of which were the same object from different orientations, which does not seem like an unreasonable level for real world applications. However, it would be interesting to see the specific impact of data set size.

Also, despite my inability to get outlier detection working, I still feel that there should be some way to perform outlier detection in order to remove bad examples, though this still may not be better than the SVM model.

6. Notes on outside resources

The only pieces of code I used in this project that I did not develop personally were built in MATLAB functionality and a MATLAB MEX interface of Thorsten Joachims' SVM-Lite software developed by Tom Briggs of Shippensburg University.

This was a solitary project and I generated the idea for the project, testing methodologies, and analysis myself.

References

[1] I. Ben-Gal. Outlier detection, 2005.

[2] G. Csurka, C. Dance, L. Fan, J. Williamowksi, and C. Bray. Visual categorization with bags of keypoints, 2004.

[3] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection, 2005.

[4] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71:273–303, 2007.

[5] B. Jian, X. Zhang, and T. Cai. Estimating the confidence interval for prediction errors of support vector machine classifiers. *Journal of Machine Learning Research*, 9:521–540, 2008.

[6] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[7] K. Lai, L. Bo, X. Ren, and D. Fox. A large scale hierarchical multi-view rgb-d object dataset, 2011.

[8] D. G. Lowe. Object recognition from local scale-invariant features, 1999.

[9] M. Luber, L. Spinello, and K. O. Arras. Learning to detect and track people in rgbd data, 2011.

[10] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38:15–33, 2000. 10.1023/A:1008162616689.

[11] F. Suard, A. Rakotomamonjy, A. Bensrhair, and A. Broggi. Pedestrian detection using infrared images and histograms of oriented gradients, 2006.

[12] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491 – 1498, 2006.