Computing Disparity on Demand: Disparity based Classification using Error-Tolerant Decision Tree Ensembles *

Arnab Dhua, Florin Cutzu
Computer Science Department
Indiana University
Bloomington, IN 47405, USA

John Bailey
Advanced Engineering Department
Delphi Corporation
Kokomo, IN 46904, USA

Abstract

Most range-based recognition systems require the calculation of a full disparity map at adequate resolutions prior to the recognition step. There also exist range-based systems that only require the computation of a sparse disparity map. We introduce a 3D shape classification method in which the disparity calculation is guided by the needs of the classification process. The method uses decision trees for shape classification and calculates the disparity only at certain locations in the image, as required by the tree structure. The calculation is very efficient as only the minimum number of disparity values are calculated. To render the classification robust, we use an ensemble of trees. The proposed ensemble method is different from the currently known ensemble methods and makes the classification system more robust to errors in the disparity calculation. The method was applied to a real world problem and good classification results were obtained.

1 Introduction and Background

Vehicle passenger-side airbags can sometimes injure a small or forward leaning passenger, or an infant placed in a front-facing or a rear-facing infant seat. We developed a vision-based system that classifies the passenger seat occupant into different size and position categories and accordingly modulates the deployment of the airbag.

The target object categories identified in our problem domain are (1) Empty Seat, (2) Rear facing infant seat (RFIS), (3) Front facing infant/child seat (FFIS, FFCS), (4) Adult in a normal or twisted position (Adult NT) and (5) Adult out of position (Adult OOP). There is substantial appearance variability (shape and texture) within each of these classes [20]. Additional sources of image variability are the various background objects (located inside and outside the vehicle) and, more importantly, the lighting variability induced by variable sunlight, street lamps, cast shadows, vehicle headlights etc. Illumination variability limits the utility of intensity-based features and requires the use of disparity-based features. We propose a disparity-based method that performs fast and accurate shape classification. Disparity has clear advantages over intensity features: (1) it is a measure of

*This work was supported by a grant from Delphi Corporation.

the true, 3D location of the object in the scene; (2) no particular image feature needs be detected; we only need match similar image regions in the the stereo pair; (3) it is robust to illumination variability. Our problem domain imposes certain constraints. First, the system has to run close to real time on inexpensive hardware. This precludes the computation of a dense disparity map. Second, the system needs to be highly accurate and reliable.

Most disparity-based object classification systems require the calculation of the full disparity map before steps such as registration [2], matching and pose estimation can be performed. Methods proposed in [12, 23] are based on matching piecewise-planar surfaces using interpolation trees. Later methods [18, 19] use a *semi-local* surface representation, the *spin image*, that simplifies the matching process. Other approaches generate features from disparity data which are then fed to classifiers like neural networks [20]. These methods also require calculation of all disparity information before the classification. Methods that use sparse disparity values to approximate full maps have also been proposed [7]. However, use of sparse features like the disparity of image corners [15] is not directly suitable for classification because a fixed number of disparity values at fixed locations are necessary for typical vector classification methods.

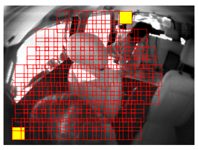
The first contribution of this paper is a method for disparity-based object classification based on decision trees. This approach does not require the calculation of all disparity values prior to the classification step. Instead, the structure of the decision tree indicates the image regions where the disparity needs to be estimated, based on the outcome of the disparities evaluated earlier.

The second contribution is a novel method for creating an ensemble of error-tolerant decision trees. The results obtained from multiple decision trees on the same input image pair are combined to achieve high classification accuracy. The use of ensembles of decision trees generated by different methods like bagging [6], random subspace selection [17], random forest creation [4], extremely random trees [14], have been evaluated in the literature. Our method is based on the intuition that decisions taken at the highest levels in the decision tree are the most critical. An ordered sequence of decision trees is generated by discarding the root attribute of the previous tree in the sequence and re-applying the CART (Classification and Regression Tree [5]) algorithm. The resulting trees are combined by a voting method, leading to an increase in the robustness of the classification.

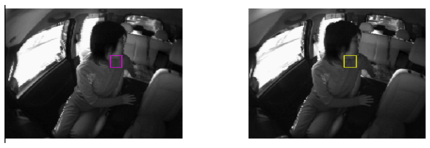
2 Method

Building the Decision Tree The first requirement for applying a CART classifier to a set of images is to extract a fixed number of disparity attributes at corresponding locations from each image. Full disparity maps meet this condition but are computationally expensive and feature based disparity methods do not yield a fixed number of disparity values at fixed locations. Therefore, our first step was to define a fixed set of small rectangular regions in the image at which disparity was calculated. Each rectangle in the set yields a single disparity value. This fixes the number and the positions of the disparity observations obtained from each training image pair.

The rectangles were chosen to be of intermediate size: too small and the matches across the images of the stereo pair will be ambiguous, too large and a good match will be difficult to find in the other image [26]. The rectangles were positioned manually such as to cover only the image locations where the objects of interest were likely to be found.



(a) The set of overlapping rectangles defined on the left image of the stereo pair. The first and last rectangles are colored to indicate size.



(b) The magenta rectangle in the left image was matched to the yellow rectangle in the right image to yield the disparity estimate.

Figure 1: Setting up the disparity calculations.

The rectangular regions were allowed to overlap, to account for the position and size variability of the objects of interest. Note that the experimenter merely specifies which portion of the image to look at and the spatial sampling frequency of the disparity profile of the scene. This step does not specify the relative importance of the various rectangles for classification. The resulting overlapping rectangles (188 in our case), defined on the left image of the stereo pair are shown in Figure 1(a).

After the rectangles were defined, disparity was calculated at each rectangle for all training images in all classes by finding the best matching rectangle in the right image for each rectangle in the left image. The match was estimated using normalized correlation. Rough knowledge of the parameters of the stereo rig and of the expected range of disparities was used to define the size of the search window in the right image. A simple optimization that performs the match in a multi-resolution manner [11, 16] was employed to increase the speed of the matching process. The basic idea behind this optimization scheme is to first look in a low resolution image pair for correspondences, and then refine the correspondences by local search in a high resolution pair. This method also ensures that the match values obtained are more accurate. We have deliberately chosen a simple and fast method for disparity calculation; obviously, the decision tree framework and the later ensemble of trees will work with any of the existing methods [25] for disparity calculation.

An example of a matched rectangle is shown in Figure 1(b). The match provides a single value for disparity–the difference in the position of the same rectangle in the left and right images. The disparity of a particular rectangle is a component of the disparity vector that is passed to CART. From each image pair in the training set an 188-dimensional vector of disparity values is thus derived.

The disparity vectors of all training images from all classes are passed to the CART (Classification and Regression Tree [5]) tree building algorithm along with their corresponding class labels. We used Matlab's Statistical Toolbox CART algorithm (the 'treefit' function), employing the Gini impurity measure [21]. The amount of pruning was decided experimentally by using a validation set. A node was eliminated when containing ≤ 3 rectangles, or whenever its purity exceeded 96.5%.

The advantage of using a tree is that it is not necessary to calculate the disparity values of all rectangles of an image pair. Instead, the structure of the decision tree dictates which sequence of disparity attributes, starting with the root, must be calculated. The location and number of points at which the disparity is calculated varies according to the input image. Thus, the algorithm computes the disparities in an "on-demand" manner. Since the average depth of the tree is much smaller (~ 15) than the total number of rectangles (~ 188), the classification is very efficient.

Building the Tree Ensemble Since the classification accuracy obtained (see Section 3) using a single decision tree was deemed insufficient, we used an ensemble of such trees. To motivate our novel ensemble approach we note the decisions taken at the very top of a tree are crucial. In our experiments, often one of the children of the root contained four classes (out of a total of five). However, the root rectangle may not be matched correctly. This could happen, for instance, in an overexposed image or in an image with a large shadowed region. Therefore, it would be beneficial to have alternate trees that do not use this mismatched rectangle. The published ensemble methods (*e.g.* [6, 17]) do not have a systematic way of making the successive trees immune to the mistakes made at higher levels of the preceding trees. Since we do not know whether the root rectangle is mismatched (the value of the correlation coefficient is not always a reliable indicator of match quality), we prepare alternate trees using different root rectangles. We focused on the root rectangle because the errors resulting from a mismatch at the root are the most serious. A sequence of trees was derived in which each tree is formed by discarding the feature present at the root of the previous tree. The sequence of trees was derived as follows:

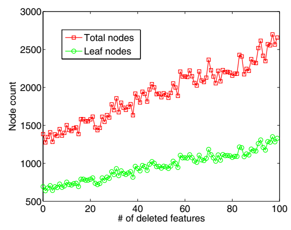
1. Build tree using current rectangle disparity data. Store tree.
2. Modify the disparity data by removing the disparity of the the root rectangle of the current tree.
3. Stop if enough trees have been produced or goto step one.

Note that the only free parameter in the proposed method is the number of trees in the ensemble. This number was chosen by optimizing the performance of the ensemble on a validation set. In our application this number was 60.

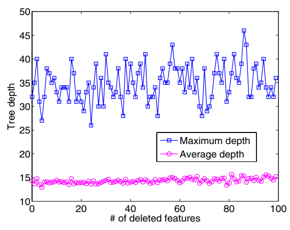
The trees of the ensemble were combined using majority voting: the class of the input image pair is the class predicted by the majority of the trees in the ensemble. We decided on majority voting after experimental comparison with two other methods, as described later.

3 Results

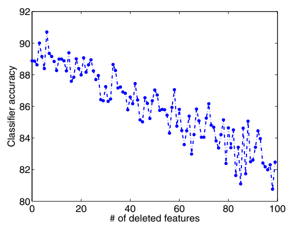
The algorithm was evaluated on a dataset of $18,862$ images taken in the same vehicle. In our experiments a low-resolution (320×240 pixels) rig mounted to the right of the rear-view mirror was used to obtain stereo images of the passenger seat area. The image set was designed to simulate the large variation in the images that a system installed in a similar real-world vehicle would encounter. The dataset included ~ 40 subjects both male and female, of varying sizes. The child classes were represented by realistic dummies. There was no restriction on the kind of clothing worn. Common objects (newspapers, blankets, purses, books, etc.) were placed along with the subjects to increase the variability. The images were taken both indoors and outdoors in varying weather conditions to simulate the variation in illumination conditions. There was no restriction on the seating position of the subject. This dataset was partitioned using different random partitions to generate different training and test sets. The training set contained 6947 images for class NT, 3118 for OOP, 1451 for FFIS, 1634 for RFIS and 998 for EMPTY for a total of 14148 images. The test set contained 2313 images for class NT, 1040 for OOP, 484 for FFIS, 545 for RFIS and 332 for EMPTY for a total of 4714 images. The results reported in this section are averaged over 5 different test and training sets of the same size.



(a) Number of total (red squares) and leaf (green circles) nodes.



(b) Maximum (blue square) and avg. (magenta circle) tree depth.



(c) Classification accuracy of each individual tree on the test set.

Figure 2: Statistics of individual trees in the ensemble. All the values are plotted against the number of deleted features.

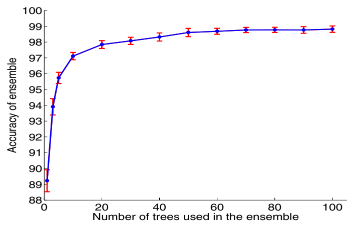
'Base' tree classification The average accuracy obtained using the first, 'base', tree in the sequence was 89.23% (standard deviation 0.6967%), with the best accuracy being 89.35%. This accuracy level ($\sim 90\%$) obtained using *only* a single decision tree is comparable to some of the previous final results published on the same problem using different features and combination of classifiers [20, 10][1]. The average depth of this tree is ~ 20, which means that on an average *only* 20 disparity values have to be calculated to reach a classification decision. This is the main source of increase in computation speed.

Comparison with the nearest-neighbor (k-NN) classifier The image pairs were encoded by the 188 disparity values of the rectangles, and Euclidean distance in this 188-dimensional space was used for classification. All training images were used as prototypes. The classification accuracy obtained with 1-NN is 80.23%. We obtained similar results of 80.19% with 3-NN, 79.02% with 5-NN and 78.28% with 11-NN. Thus, even the 'base' decision tree (89.23%) significantly outperforms nearest neighbor classifiers.

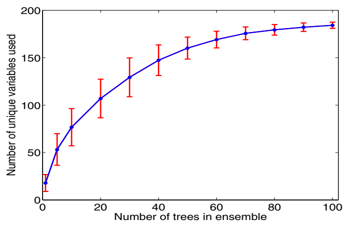
Statistics of individual trees Figure 2(a) shows the variation in total nodes and leaf nodes as we eliminate features. We see that the trees are becoming larger. This agrees with our notion that the later trees in the ensemble have to be more complex to be able to deal with the absence of the more discriminative attributes. Figure 2(b) shows the change in depth of the individual trees as we drop features. The average depth increases only slightly (varies between 13 and 17), thus the later trees in the ensemble are not significantly slower than the earlier trees. As expected, the depth of the trees increase only as the log of the number of nodes. The average depth *per class* for each tree (not plotted) is almost the same as the average depth of that tree. This indicates that on an average, classification into any of the classes should take approximately the same amount of time. One would also expect the classification accuracy of this ordered sequence of individual trees to decrease, as we are eliminating the most discriminant nodes one by one. This is indeed the case as can be seen in Figure 2(c). However, given that the likelihood of a false match at the root of earlier trees is not negligible, the use of multiple, if weaker, trees, is justified.

Performance of the tree ensemble To evaluate the accuracy of the ensemble of error-tolerant trees we trained the individual trees in the ensemble, combined them using majority voting and tested them against test sets. We performed these experiments on the same training and test set combinations as for the 'base' tree. The results are shown in Figure 3(a).

[1]The datasets used in [20, 10] while not the same as ours, are qualitatively very similar. They have also been designed to address the same sources of variability.



(a) The mean and std. dev. of the accuracy over 5 different sets of training and test data.



(b) The mean and std. dev. of the number of unique attributes used by the ensemble during classification.

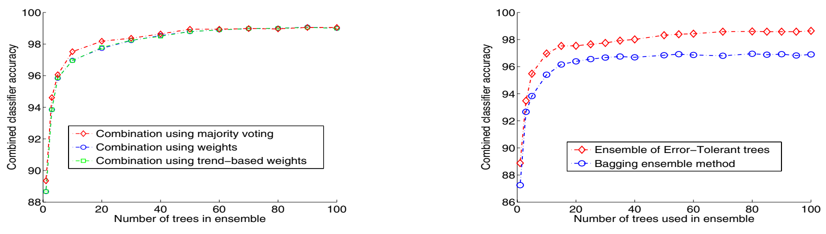
Figure 3: Plots of accuracy and number of unique attributes used by the proposed ensemble. The x-axis in both plots represents the number of trees contained in the ensemble.

Initially, there is an almost monotonic increase in classification accuracy on the test set with increase in the number of decision trees used in the ensemble. This increase levels out after a point. As explained before, this region in the graph is the optimum choice for the number of trees, because a further increase in number of trees would increase the required computations with no tangible benefits. We had decided that the optimum number of trees is 60 using a validation set. These results on the test set also validate our choice. With this choice for the optimum number of trees the classification rate is 98.68% on an average with a standard deviation 0.1904%. This is better than the accuracy reported in [20, 10] on the same problem. Also, as we can see from the small values of the standard deviation, the accuracy of the ensemble is very stable with respect to changes in the training set and test set.

Figure 3(b) shows the efficiency of the tree ensemble with regard to the average number of *unique* disparities that need to be computed for a general input image, plotted against the number of trees in the ensemble. Most of the trees in the ensemble share the same rectangle features, and once computed for one tree, these features do not need to be computed again. We can see that even with 30 trees in the ensemble only 60% of the available prediction attributes are being used on an average. This means that the system gives a classification accuracy of $\sim 98.07\%$ using only ~ 120 disparity calculations. We have not optimized our code for speed, but clearly a system that needs only around 120 disparity estimates will run very fast. (Consider that a dense disparity method will need around $320 \times 240 = 76800$ disparity estimates.)

Comparison of majority voting with other combination methods We compared majority voting with two other methods, which we call weighted voting and trend-based voting. In weighted voting the training set is split into two, one larger part is used to train the individual tress and the second (smaller) part is used as a validation set to find the weights (proportional to accuracy) of each individual tree. Trend-based weighted voting is similar to weighted voting, except we know that each successive tree should have slightly lower classification accuracy than the previous tree in the ensemble. While testing on the validation set, this is the overall trend noticed. So instead of directly using the classification accuracy (on the validation set) of each individual tree as its weight, and thus probably overfitting to the validation set, we fit a line to the gradually decreasing accuracy of the trees. Thus in this case every individual tree gets a strictly lower weight than the preceding tree.

To compare majority voting, weighted voting and trend-based voting the test set was exactly the same for the three methods. However, for majority voting the entire training



(a) Three voting methods (b) Comparison with bagging

Figure 4: Comparison of different combination methods and different ensemble methods. The x-axis in both plots represents the number of trees contained in the ensemble.

set was used to train the individual trees. For the other two methods the training set was split into two parts: actual tree training set and the validation set. Figure 4(a) shows the results of the three different combination methods. When there are > 60 trees in the ensemble all three combination methods give essentially identical results. For < 60 trees, the majority voting method fares better. Similar results were obtained with different selections for the training and test set. Since majority voting does not require the estimation of any additional parameters and also performs better we decided to use majority voting.

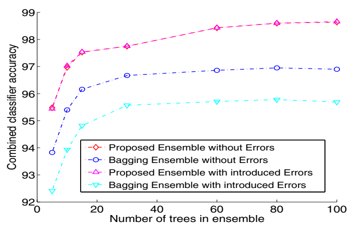
Comparison with bagging We compared our ensemble method with bagging [6]. We generated multiple bootstrap replicates by sampling with replacement [9] from the training set and used these replicates to build the individual trees which were then combined by majority voting. The two methods were compared using the same training and test sets and also using the same number of trees in the ensemble. Our method performed better by around 2% as seen in the Figure 4(b).

We performed another set of experiments to further validate our claim that our system is tolerant to a wrong decision taken higher up in the tree structure by trees present earlier in the ensemble. The experimental setup was as follows: given a bagging system and our error-tolerant ensemble trained on the same training set, we estimate which method is more tolerant to errors introduced into the test set. We choose an attribute that is highly discriminative and is thus placed high up in the tree structure. We introduce errors in the disparity values for that attribute, for the test images. The error value was taken from a uniform distribution between -5 and +5. Then this erroneous data is used for classification using both ensemble methods. The results are seen in Figure 5(a). There is almost no difference in the classification accuracy with and without the erroneous data while using our error-tolerant ensemble. In the case of bagging we notice a 2% increase in error. We repeated these experiments with various training and test sets and obtained similar results.

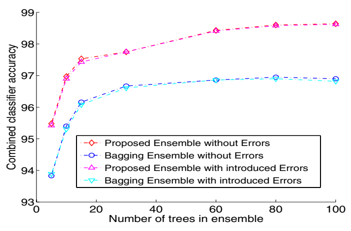
Although it is clear from Figure 5(a) that our ensemble is tolerant to errors occurring higher up in the tree, we also need to ensure that the ensemble method is resilient to errors in estimation made at lower levels in the tree structure. We repeated the above experiments for an attribute that occurs at a lower level in the tree and is less crucial for classification. This time both our method and bagging give almost the same accuracy with or without error (Figure 5(b)) as the attribute is not very crucial to the decision process.

4 Discussion

We introduced a shape classification method based on applying decision trees to local disparity features. Use of local disparity features is appealing as it does not require a time-



(a) Errors introduced in an attribute that is very crucial for classification.



(b) Errors introduced in an attribute that is *not* very crucial for classification.

Figure 5: Comparison of the accuracy of the proposed ensemble with bagging when errors are introduced in the test set. The x-axis in both plots represents the number of trees contained in the ensemble.

consuming calculation of a full range map. In addition, disparity is not affected by the unpredictable variability of the texture of the occupants clothes or by drastic illumination changes affecting vehicle interiors.

Tree classifiers have the unique advantage that their decision making can be explained very intuitively [24, 14]. In our application there is a possibility that a set of rectangles that give good classification on the training set will *not* yield good results on novel images. Poor generalization will occur if the rectangles happen to incorporate “accidental” features (for e.g., if presence of certain objects outside the vehicle is associated with certain image classes) irrelevant to the classification. Since we are using decision trees it is easy to check for accidental features by manually inspecting the rectangles (at least the most discriminative ones) used for classification. In fact, in our experiments we saw that the the most important rectangles for classification were chosen in the region where the seat back meets the seat base. This region clearly contains the most discriminative information about the different occupants.

In addition, decision trees are computationally efficient. The tree structure requires the computation of only very few disparity values in an “on-demand” manner, based on the outcome of the disparities evaluated earlier. We have tested our method only with the CART [5] implementation of decision trees, but we expect to obtain similar results with other decision tree methods (*e.g.* [22]).

One reason that our method works so well is that we do not need to perform a full search in the image space to locate the objects of interest. In our problem domain the passenger will appear only within a certain area in-and-around the seat. While the location is not fixed, it is not as difficult as finding an object of interest in a general scene. Although the use of disparity for classification in a decision tree framework is tailored to our application, the “on-demand” architecture proposed is general and can be used for classification based on features that are expensive to compute.

A number of ensemble methods which satisfy the perturb and combine paradigm [3, 4] in the context of decision trees already exist. These methods give rise to a set of trees by varying different parameters of the decision tree creation process. Some of these ensemble methods can also be applied to classifiers other than decision trees. In bagging [6] the different models are produced by drawing a bootstrap replicate of the learning sample before each induction step. Another ensemble method proposed by Ho [17] is based on selecting a random subset of the input attributes before building each model of the ensemble. Breiman combines this later idea with bagging to form his “random forests” [4] algorithm. Besides the general ensemble methods described above, there exist ensemble methods specifically designed for decision trees. These methods work by altering the tree creation algorithm. Ali and Pazzani [1] perturb the algorithm by replacing the choice of the best test by the choice of a test at random among the best ones. Dietterich [8] proposes a very similar method that randomly selects a test among the 20 best splits.

Our method for creating a classifier ensemble is different from the other methods explored in the literature. Bagging [6] involves changing the training data set by removing training examples. Our method involves changing the data set by removing features that the classifier could have used. In this sense it is similar to random subspace selection method of [17]. However, unlike in [17] our criteria for selecting the attributes to be left out is not random, rather it is based on the idea of preventing the classification results from being affected by errors in the most crucial decisions. The proposed error-tolerant ensemble method is not specific to our application and can be applied in general with decision trees.

Efforts have been made to compare the performance and working of an ensemble of trees to the AdaBoost [13] approach. In [4] the author conjectures that AdaBoost is in fact equivalent to "random forests". The intuition for our ensemble method, *i.e.*, reinforcing against previous errors made by a good classifier, differs from that of AdaBoost which uses a combination of weak learners to reach an accurate classification. An interesting avenue for future work would be to compare in detail the working of our proposed method with the AdaBoost method.

We experimented with using the quality metric of the disparity estimate to improve the accuracy of the error-tolerant ensemble. On comparing this method we see that if we use few such trees (≤ 10) in the ensemble the results were slightly better than when using error-tolerant ensembles of ordinary CART trees. When using more (> 10) trees both methods start giving comparable results. Since the optimal number of trees is around 60 the error-tolerant ensemble of *ordinary* CART trees suffices. The power of using a collection of trees outweighs the benefits of using individual trees that are slightly better.

5 Summary and Conclusions

We introduce a decision tree methodology for the classification of disparity data. Decision trees have the benefit that the disparity does not have to be calculated at every point before the classification can start; rather, an "on-demand" disparity calculation is performed. This leads to a significant speedup as the disparity values have to be calculated only at a small subset of the image locations. We show that this 'basic' method is much superior to nearest-neighbor approaches even if the whole training set is stored. We further propose and evaluate the use of an ensemble of such decision trees to achieve very high classification accuracies. The ensemble is based on the idea of protecting the most crucial decisions taken by the earlier decision trees. This ensemble method compares favorably to bagging and we also demonstrate the error-tolerance property of this method. We apply our system to a difficult real world problem and achieve good classification results.

References

[1] Kamal M. Ali and Michael J. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202, 1996.

[2] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *PAMI*, 14(2):239–256, February 1992.

[3] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[6] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] Arnab Dhua, Florin Cutzu, Riad Hammoud, and Stephen Kiselewich. Triangulation based technique for efficient stereo computation in infrared images. In *Intelligent Vehicles Symposium*, June 2003.

[8] Thomas G. Dietterich. Ensemble methods in machine learning. *LNCS*, 1857:1–15, 2000.

[9] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[10] Michael E. Farmer and Anil K. Jain. Occupant classification system for automotive airbag suppression. In *CVPR*, pages II: 756–761, 2003.

[11] O. Faugeras, B. Hotz, H. Mathieu, T. Viville, Z. Zhang, P. Fua, E. Thron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation-based stereo: Algorithm, implementations and application. Technical report, 1993.

[12] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *IJRR*, 5(3):27–52, 1986.

[13] Y. Freund and R. Schapire. A Short Introduction to Boosting. *Journal of Japanese Society for AI*, 14(5):771–780, 1999.

[14] P. Geurts. Extremely randomized trees. Technical report, University of Liege, 2003.

[15] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proc. 4th Alvey Vision Conf.*, pages 147– 151, 1988.

[16] H. Hirschmuller, P.R. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *IJCV*, 47(1-3):229–246, April 2002.

[17] Tin Kam Ho. The random subspace method for constructing decision forests. *PAMI*, 20(8):832–844, 1998.

[18] A.E. Johnson and M. Hebert. Surface matching for object recognition in complex 3-dimensional scenes. *IVC*, 16(9-10):635–651, July 1998.

[19] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *PAMI*, 21(5):433–449, May 1999.

[20] Henry Kong, Qin Sun, William Bauson, Stephen Kiselewich, Paul Ainslie, and Riad Hammoud. Disparity based image segmentation for occupant classication. In *CVPRW*, 2004.

[21] Richard J. Light and Barry H. Margolin. An analysis of variance for categorical data. *Journal of the American Statistical Association*, 66(335):534–544, 1971.

[22] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[23] I. D. Reid and Michael Brady. Recognition of object classes from range data. *AI*, 78(1-2):289–326, 1995.

[24] S. Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE Trans on SMC*, 21(3):660–674, May/June 1991.

[25] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *IJCV*, 47:7–42, 2002.

[26] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.