3D Point Representation For Pose Estimation: Accelerated SIFT vs ORB

K.K.S. Bhat(✉), Juho Kannala, and Janne Heikkilä

Center for Machine Vision Research, University of Oulu, Oulu, Finland
{sbhatkid,jkannala,jth}@ee.oulu.fi

Abstract. Many novel *local image descriptors* (Random Ferns, ORB etc) are being proposed each year with claims of being as good as or superior to SIFT for representing point features. In this context we design a simple experimental framework to compare the performances of different descriptors for realtime recognition of 3D points in a given environment. We use this framework to show that robust descriptors like SIFT perform far better when compared to fast binary descriptors like ORB if matching process uses *approximate nearest-neighbor search* (ANNS) for acceleration. Such an analysis can be very useful for making appropriate choice from vast number of descriptors available in the literature. We further apply machine learning techniques to obtain better approximation of SIFT descriptor matching than ANNS. Though we could not improve its performance, our in-depth analysis of its root cause provides useful insights for guiding future exploration in this topic.

Keywords: 3D point recognition · Augmented Reality · Interest Points

1 Introduction and Background

Estimating pose (position and orientation) of a camera in a given image is at the heart of many applications in Augmented Reality and Visual Robot Navigation. Over the past few years *interest point* or *keypoint* based 3D point recognition has facilitated substantial progress in vision based camera pose estimation. 3D point recognition provides the necessary input for well known PnP (Perspective n-Point) framework [13,15,29] for computing pose. Many high dimensional robust keypoint descriptors [3,18] have been developed in order to identify 3D points in images under the challenge of variation in camera viewpoint. SIFT descriptors are consistently proven to be one of the best candidates for point matching [8,19]. While they have been successfully applied on problems involving computation of camera pose [4,25], their cost of computation and matching prevents their use in applications which require real-time speed on video frames.

For achieving real-time speed on video frames, many keypoint recognition approaches which employ simple binary decisions based on raw pixel comparisons are being proposed. In [14] sequence of binary decisions based on comparing random pair of pixels around interest points (extremas of Laplacian of Gaussian

R.R. Paulsen and K.S. Pedersen (Eds.): SCIA 2015, LNCS 9127, pp. 79–91, 2015.
DOI: 10.1007/978-3-319-19665-7_7

filter) are used to identify the keypoints. Random Ferns framework [22] uses similar pixel comparisons, but the sequence of decisions are converted to binary coding which is indexed directly to attribute values for faster recognition. In [27] boosting is used to arrive at the optimal binary coding scheme for an image patch around a keypoint. In [24] depth values from Kinect sensors are also used along with the color information while learning the sequence of binary decisions.

Despite claims of significant success in each new approach, the effort to produce new methods to match the performance of SIFT seems to be still continuing. The reason may lie in the failure of those approaches to stand up to their claims in independent evaluations. For example, in [17] , it is shown that SIFT descriptors perform better than random ferns in terms of ability and accuracy of matching. Evaluation in [11] claims that SIFT outperforms many popular binary descriptors like BRIEF [6], BRISK [16], ORB [23].In this context we feel that it is immensely necessary to perform careful evaluation for a newly designed descriptor before claiming improvement over SIFT. Moreover, it is preferable if the evaluation framework can be applied on images of any environment and not limited to datasets which either have dense depth information [30] or put limitations on the geometric variations [1].

1.1 Contribution and Overview

In order to make a fair evaluation, we feel that it is necessary to match SIFT descriptors with operations having similar computational complexity as that of the type of descriptor with which it is being compared. We use approximate nearest neighbor search (ANNS) to accelerate SIFT based matching and compare its accuracy with that of ORB binary descriptor [23]. The choice of ORB is justified by the fact that ORB, like SIFT, has invariant property w.r.t. rotation and it is shown to be better compared to other binary descriptors in the presence of scale and rotation change [12]. We design an evaluation framework which needs only a sparse set of 3D points and camera positions in a set of images of the environment. This information can be obtained for any environment by simply running Structure from Motion (SfM) [28] on a set of images. The results from this experiment motivates us to explore further in the direction of improving techniques for matching SIFT descriptors rather than trying to design new binary descriptors. We train axis-parallel decision trees (APDT, decision trees which use only one attribute of the feature vector in a node to take decision) to learn SIFT based matching. We exploit the fact that SIFT descriptors are integer valued and perform exhaustive search to obtain optimal decision threshold at each node of the tree. We find that ANNS still performs slightly better than APDTs. Next we employ Canonical Correlation Analysis (CCA) [26] and train oblique decision trees whose decision boundaries need not strictly align to the axes of the SIFT descriptor space. Though we do not succeed in obtaining improvement over ANNS by using oblique decision trees for SIFT based matching, the insight obtained through these experiments which are detailed in this paper are useful for future exploration in this topic.

This paper is organized as follows. In section 2, we present the decision tree framework we used for accelerating SIFT based matching. The three types of different decision trees we use in our experiments are described in subsections 2.2, 2.3 and 2.4 respectively. Our evaluation framework and the results we obtained through experiments are presented in section 3. Finally, we mention concluding remarks and future work in section 4.

2 Decision Trees for Fast SIFT Descriptor Matching

Let $\mathbb{Q} = \{Q_1, Q_2, ..., Q_N\}$ be a sparse set of 3D points in the target environment. Let $S = \{f_i\}$ bet a set of SIFT vectors extracted from images of the environment. We use decision tree to identify 3D points using SIFT vectors extracted from images. For training decision tree we need class labels $C = \{c_i\}$ where each label c_i is associated with $f_i \in S$. For the task of 3D point recognition, labels are such that $c_i = k$ if f_i corresponds to 3D point Q_k, otherwise $c_i = 0$. For a vector f, we use the superscript f^j to denote the value of j^{th} attribute. We can obtain $\mathbb{Q}, S, C$ by performing Structure from Motion (SfM) [28] on a set of images of the environment (as described with experimental setup in section 3.1). Rest of this section is as follows. First, we provide a brief description of general framework for training axis-parallel decision tree (APDT) in section 2.1. For APDT, the decision at each node is based on one of the attribute values of the descriptor vector. Two subsequent subsections present the methods we use for random and exhaustive search for optimal decision boundaries for APDT. In section 2.4 we present the method we used to learn *oblique* decision trees through *Canonical Correlation Analysis* [26]. Oblique decision trees use linear combination of values of multiple attributes of descriptor vector for decision at each node.

2.1 Training APDT

Decision trees are built recursively during training. Each tree node (starting with root node which receives all the training samples) decides whether the given set of training samples (S,C) should be split or not based on an entropy measure. The tree building procedure we employ at every tree node is described in Algorithm 1. If the samples need to be split, then an optimal combination of attribute t_S and threshold θ_S is computed. The training samples f with $f^{t_S} \leq \theta_S$ are directed towards left node. The rest are directed towards right node. If the samples need not be split, then a leaf node is created which stores the dominant label c_S and pointers to all the samples provided to it. We use *Information Gain* (IG) based measure [7] to choose optimal values θ_S and t_S. Optimal decision parameters maximize the IG. When a set S is split into S_L and S_R, IG of this division (denoted by $G(S)$) is defined as

$$G(S) = H(S) - \sum_{i \in \{LR\}} \frac{|S_i|}{|S|} H(S_i) \tag{1}$$

Algorithm 1. Tree(S,C) : Basic tree growing algorithm
Description: This algorithm is recursively applied to build APDT from a set $S = \{f_i\}$ of SIFT descriptor training vectors with corresponding labels $C = \{c_i\}$. In each recursion, an information entropy value based on the class distribution of S is computed to decide whether the samples in S need to be split or not. If S need not be split, then a leaf node is created containing all the samples in S. Otherwise a decision node is created which splits S into two subsets by applying a threshold θ_S on one of the attributes t_S. The optimal value for the pair (t_S, θ_S) is selected so as to increase the IG (as defined in equation 1).

1: **if** SamplesNeedToBeSplit(S,C) **then**
2: Create a decision node with attribute t_S and threshold θ_S which optimally splits S into two groups S_l and S_r with labels C_l and C_r respectively. (For details see section 2.1)
3: Add the nodes Tree(S_l,C_l) and Tree(S_r,C_r) respectively as the left and right children of the decision node. Next level of recursion is applied on these two child-nodes.
4: **else**
5: Create a leaf node storing the dominant label c_S and pointers to elements of S.
6: **end if**

where $H(S)$ is the information entropy of a set S defined as

$$H(S) = -\sum_{c \in C} p(c) log\, p(c) \tag{2}$$

We stop growing a tree node (and declare it as a leaf node) when the number of training samples available to it is less than N_{min} or when the entropy of the samples given to the node is less than H_{min}. As in [20], we set $N_{min} = 2$ and $H_{min} = 0$, that is, the training algorithm keeps on splitting the samples until all the samples in a subset belong to the same 3D point.

2.2 Random Search for Optimal Decision Parameters

In general, it is difficult to compute the best value for the decision parameters since SIFT vectors have 128 attributes and the number of different values that a descriptor attribute can have is very large. Hence, at each node, the IG measure is evaluated only on a small set of random pair of values (t_S, θ_S) and the best value among them is chosen. Multiple decision trees are learned with such strategy. During testing the outcome of those multiple trees are aggregated to obtain the final class label for a test sample. Aggregation is performed during testing through weighted voting scheme (More details in section 3.3). In our implementation we try 500 random decision values for each node. First, we randomly select 500 attributes between 1 to 128. For each random attribute, we choose a random threshold value between minimum and maximum value attained for the attribute by the samples used to train the node. We train 10 such trees during training.

Algorithm 2. FindBestThreshForAttribute(S,C,t)
Description: Finds threshold value for attribute t which optimally splits S for a given entropy measure

1: Find unique set of values (in the increasing order) $A = \{a_1, a_2, ..., a_k\}$ the samples in S take for attribute t.
2: Compute mid-values $B = \{b_i = \frac{a_i + a_{i+1}}{2}\}$.
3: Compute gap-values $g_i = a_{i+1} - a_i$
4: **for all** $i \in \{1, 2, ..., k-1\}$ **do**
5: Compute $S_l = \{f \in S : f^t \leq b_i\}$ and $S_r = \{f \in S : f^t > b_i\}$
6: $G_i \leftarrow$ IG (equation 1) due to splitting S in to S_l and S_r using threshold b_i
7: **end for**
8: Return the b_i and g_i corresponding to the highest G_i.

2.3 Exhaustive Search for Optimal Decision Parameters on SIFT Training Vectors

The attributes of SIFT descriptors take integer values between 0 to 255 (provided they are not normalized to unit length). Hence, each attribute can partition S in at most 254 different ways. This enables us to perform exhaustive search for optimal attribute-threshold values. The procedure for performing this exhaustive search for a given attribute t on the set of training samples S with labels C is explained in Algorithm 2. We perform this search on each attribute of S to compute the optimal decision value.

It is easy to notice in Algorithm 2 that any threshold value in an interval $[a_i, a_{i+1})$ will split S in the same way and hence lead to same value for IG (equation 1). Hence, we need to try only one threshold value for each interval in order to search for best IG. For each interval we use mid-value for threshold $b_i = \frac{a_i + a_{i+1}}{2}$ in order to maximize the margin between left and right samples. For threshold value b_i, the IG (G_i) is computed and the best threshold value b_j with highest IG is chosen. The algorithm also returns gap between left and right samples which is $g_j = a_{j+1} - a_j$. This algorithm is executed on all the attributes and the attribute threshold pair leading to highest IG is chosen. If there are multiple such decision parameters, then, we choose the one which gives maximum gap g_j.

2.4 Training Oblique Decision Tree Through Canonical Cross Correlation (CCA)

Given two matrices $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{n \times d}$ and $Y = [y_1, y_2, ..., y_n] \in \mathbb{R}^{n \times k}$, CCA [26] computes two projection vectors $w_x \in \mathbb{R}^d$ and $w_y \in \mathbb{R}^k$ such that the correlation coefficient

$$\rho = \frac{w_x^T X Y^T w_y}{\sqrt{(w_x^T X X^T w_x)(w_y^T Y Y^T w_y)}} \tag{3}$$

is maximized. If X contains training SIFT vectors as its columns and Y contains corresponding class labels as its columns in *1-of-k* binary coding scheme, (i.e., in i^{th} column of Y the j^{th} element is 1 if the SIFT vector x_i is associated with 3D point Q_j, otherwise it is zero) then the projection vector w_x can be used to obtain oblique decision at each node as follows. If X is the set of sample vectors provided to a tree node during training, we compute its w_x (which maximizes equation 3). For a threshold value θ, the training vectors satisfying $w_x^T x_i \leq \theta$ are directed towards left node and the rest are directed towards right node. Optimal value for θ is chosen based on the IG of the split. In our experiments we have used the regularized version of the CCA (rCCA) which solves the generalized eigenvalue problem

$$XY^T(YY^T + \lambda_y I)^{-1} YX^T w_x = \eta(XX^T + \lambda_x I)w_x \tag{4}$$

in order to compute w_x [26]. We set $\lambda_y = 0, \lambda_x = 0.1$.

In [9] CCA is used to reduce the dimension of the input data and then Random Ferns [22] are trained to perform classification instead of decision trees. In such a framework, each class label needs $F \times 2^{d'}$ number of double values to be stored during the classification process, where F is the number of ferns and d' is the reduced dimension of the vector space. For the parameter values used in the experiments of [9] this amounts to a memory storage requirement of more than 10MB per 3D point. This is very costly since SfM on a small target environment (eg. a single office room) may produce thousands of 3D points. In contrast the decision trees need only 128 double values (128 is the dimension of the SIFT vector) at each node. In our experiments the tree contained nearly $100k$ nodes which requires less than 100MB memory in total.

3 Experiments

In this section, we present the experimental setup for evaluating 3D point recognition accuracy for pose estimation (Sec. 3.1), list the software libraries used in our experiments (Sec. 3.2) and, then, present experimental results and discussion based on those results (Sec. 3.3).

3.1 Evaluating Accuracy

For evaluating the accuracy of keypoint recognition methods for camera pose estimation we need the following information:

1. Camera positions corresponding to the images
2. 3D coordinates of at least some of the keypoints in the images

This information can be obtained by performing SfM [28] on the set of given images as follows. Keypoint descriptors extracted from the images can be matched to obtain point-to-point correspondences between each pair of images. Using these 2D matches, SfM computes the camera position in each image and the 3D coordinates of the points corresponding to matched image locations. The

keypoint descriptors associated with the 2D matches of a 3D point can be labeled with the ID of the 3D point. The descriptor vectors not associated with any 3D point can be labeled as 0. Thus we obtain information (1) and (2) mentioned above.

Sometimes camera positions are given along with images (as in the case of 7Scenes dataset[24] we use). For such cases we can establish correspondence between the keypoint descriptors and 3D points through triangulation as follows:

1. Extract keypoint descriptors from each image
2. Obtain point-to-point matches between each pair of images by matching the descriptors extracted from those images
3. Discard those point-to-point matches which do not comply with epipolar constraints based on the fundamental matrix between a pair of images
4. The matched descriptors are tracked across images to obtain clusters
5. For each cluster obtain the 3D coordinates by performing triangulation [2] using the given camera positions
6. Label each descriptor vector associated with a 3D point using the ID of the 3D point. Otherwise the label is 0

Once we have the camera positions and labeled keypoint descriptors for a given set of images we can evaluate the accuracy of a 3D point identification method through reprojection error. We divide the set of images into Train and Test sets. The keypoint descriptors from the Train image set are used as samples for training. During testing the keypoint descriptors from test images are assigned one of the class labels. If a test descriptor is associated with a 3D point, then, we can measure its reprojection error in pixels using camera pose of the test image. In our experiments the matches which have less than 8 pixel error of reprojection are treated as *good* matches and those having higher error than that are considered as *bad* matches.

3.2 Libraries for ANNS, CCA and ORB

For performing ANNS on SIFT we use Balanced Box Decomposition (BBD) structure based library [21]. BBD is also a tree structure built using training set of vectors where each node is associated with a region in the descriptor vector space. It has many interesting properties like (i) the cardinality and geometric size of the region associated with a node reduce exponentially as one descends the tree, (ii) reasonable bound based on the tolerance ϵ on the distance ratio of the retrieved nearest neighbor (NN) to the actual NN while performing approximate search. For CCA, we use the code provided along with [26]. For computing ORB keypoints and descriptors we use OpenCV [5] library.

3.3 Results

In our experiments we use sequence 1 and 2 of the office sequences in 7Scenes dataset [24]. Both sequences contain 1000 images each. We subsample them by selecting 1 in every 8 images. We use the descriptors (SIFT and ORB) from 125 images of seq 1 for training and those from the other 125 images in seq 2 for

Table 1. APDT based classification on SIFT descriptors and hamming distance based classification on ORB descriptors. For descripton of columns please see section 3.3. Average time per image for Exh-Tree T-250 is 2 milliseconds, Random Tree with Maj vote takes 20 milliseconds, Random Tree with threshold and inv-dist vote takes 25 milliseconds. ORB descriptors take nearly 3 seconds per image (BruteForce matching in OpenCV [5]). Exh-Tree T-250 clearly outshines ORB in accuracy and speed.

	Exh-Tree	Exh-Tree T-200	Exh-Tree T-250	Rand-10 Maj	Rand-10 T-250	Rand-10 Inv-Dist Vote	ORB
Good	12.59	10.7	11.79	6.70	27.99	10.73	10.35
Bad	35.52	1.4	3.50	1.39	5.80	1.71	15.86

testing. There are around 90k and 80k SIFT descriptors in Train and Test set respectively. There are around 110k ORB descriptors each in Test and Train set.

Results with APDT and ORB. Results are shown in table 1, 2 and 3. Each column in these tables corresponds to a particular type of classification. There are two rows in each table showing the accuracy values. The 'Good' and 'Bad' rows indicate the % of matches that are good and bad respectively based on the reprojection error. The rest of the test vectors either matched with those training vectors which are not associated with any 3D point or did not match any training vector at all.

Table 1 shows the accuracy for APDT and ORB. Columns 2 to 7 show results of using different decision trees trained using SIFT training vectors. The last column shows the result of using hamming distance threshold based classification of ORB descriptors (we use the threshold 30). Exh-Tree in column 2, 3 and 4 indicates single decision tree trained using exhaustive search for optimal decision parameters. Column 2 (Exh-Tree) shows the result of assigning the class label of leaf node training samples to each test vector to which it reaches during classification. It contains only 12.59% good matches and 35.52% bad matches which is very large. In order to reduce bad matches we computed the distance between the test vector and the training samples at the leaf node and discarded those having distance higher than a particular threshold. Column 3 and 4 shows the result of Exh-Tree for distance threshold 200 and 250 respectively. We can see that thresholding at leaf reduces the bad matches significantly. But in order to perform thresholding at leaf we have to store all the training vectors even at run time. The time required for performing computation is 2 milliseconds which indicates that this method can be applied even in a larger environment to obtain real-time performance.

Next three columns (5, 6 and 7) show the results of classification by aggregating the outcome of 10 trained trees using random search for optimal decision parameters. Column 4 uses majority vote, column 5 applies a threshold on the distance to the NN among all the leaf samples of 10 trees and column 6 uses a voting mechanism in which each leaf sample's vote is weighed by inverse of its distance from the test vector. We can see that the only case having significantly

Table 2. ENN Classification using SIFT descriptors. Average time needed to compute the first neighbor is 7 seconds/image. Please refer to subsection "Results with ANNS" below for details.

	NN	R-of-NN				T-on-NN				
		0.6	0.7	0.8	0.9	100	150	200	250	300
Good	37.07	1.54	3.58	7.73	16.48	8.73	19.72	28.76	34.44	36.76
Bad	21.35	0.03	0.08	0.24	1.35	0.40	2.07	5.65	11.47	18.24

Table 3. ANN Classification using SIFT descriptors. Average time needed to compute the first neighbor is 2 milliseconds/image when approximation tolerance is set to 40. Please refer to subsection "Results with ANNS" below for details.

	NN	R-of-NN				T-on-NN				
		0.6	0.7	0.8	0.9	100	150	200	250	300
Good	19.35	5.49	7.73	10.53	14.22	5.76	11.79	15.79	17.96	18.97
Bad	36.70	0.24	0.85	2.98	10.10	0.28	1.50	4.58	9.76	17.72

higher good matches from Exh-Tree T-200 is column 6 (Rand-10 T-250). But the computational cost increases for it by a factor of 10 (20 milliseconds). Hence, using a single trained tree with exhaustive search by exploiting the integral value property of SIFT descriptors helps in achieving faster computation of matches.

The last column corresponding to ORB has 10.35% good matches and 15.86% bad matches. Such high % of incorrect matches is not suitable for pose estimation. We computed pose for each test image using the matches provided by Exh-Tree T-200. We use the algorithm [15] to compute pose with 1000 RANSAC [10] trials in order to reject outliers. After obtaining the pose, those images whose position is within 5cm and orientation is within 5° from the ground truth are declared to be correct (same as in [24]). We found that 92% of the camera positions obtained by our method are correct. This is better than that reported in [24] (86.8% when frame-to-frame tracking is used and 79.1% otherwise) which also uses depth information provided by the Kinect sensor along with the color image. This clearly indicates that it is better to use robust descriptors like SIFT with fast matching method rather than using binary descriptors for identifying 3D points for pose estimation.

Another thing we observe is that matching SIFT vectors through decision trees is much faster to matching ORB without any acceleration strategies. If approximate matching approaches are used for ORB for acceleration, it will only deteriorate the accuracy further from what is already considered very poor.

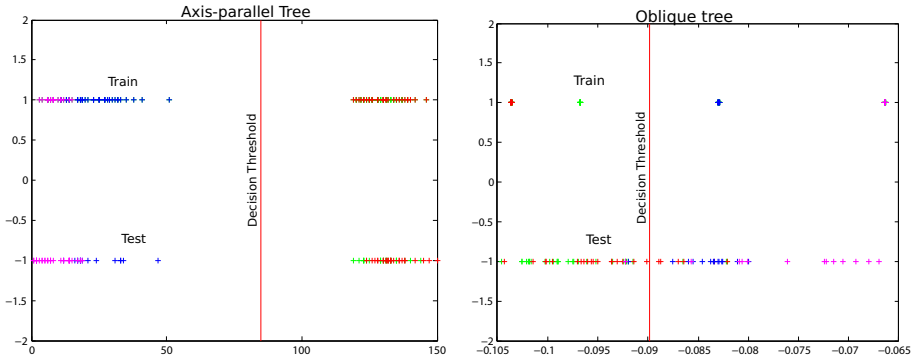
Results with ANNS. Table 2 and 3 show results of Exact Nearest Neighbor (ENN) and Approximate Nearest Neighbor (ANN) based classification respectively on SIFT descriptor vectors. The three column titles are described below:

- NN corresponds to NN classification (test vector is associated with the label of the closest training vector).
- R-of-NN corresponds to classification based on threshold on ratio of distances to two closest NNs belonging to different class labels. We compute 5 closest NNs for a test vector. If all the 5 training vectors belong to the same class, then the test vector is assigned to that class. Otherwise, we compute the ratio of distance of the NN and to the second NN belonging to a different class than the NN. If this ratio is less than a particular threshold, then, we assign the test vector to the class label of the NN. Otherwise, we discard it.
- T-on-NN classifies a test vector by applying a threshold on the distance.

The different threshold values applied on each case are indicated in the second row of these tables. From table 2, it is clear that ENN based classification provides better results than decision trees. For example T-on-NN with threshold 150 provides $19.72\%, 2.07\%$ good and bad matches which can be considered better than the results provided by Exh-Tree in columns 2, 3, 4 of table 1. But, ENN classification requires 7 seconds to finish computation where as Exh-Tree needs only 2 milliseconds. In order to accelerate NN matching we increased the tolerance value ϵ of ANNS to 40 at which the computation finished in 2 milliseconds per image. The results are shown in table 3. Though the approximation has reduced the accuracy of nearest neighbor matching it is still slightly better than decision trees. For example, T-on-NN with threshold 150 in table 3 gives $11.79\%, 1.50\%$ good and bad matches. Despite reduced accuracy it is still better than columns 2, 3, 4 of table 1.

Results with Oblique Decision Tree Based on CCA: We also experimented with oblique decision tree in order to improve the accuracy further. Due to the increased flexibility in decision boundaries we hoped to obtain better results. But, with oblique decision tree trained using CCA the results on test data only got much worse compared to APDT. In order to analyze the results we scaled down the problem to 4-class classification by selecting samples from only 4 clusters. We performed training and testing for one node on this reduced dataset using APDT and oblique decision trees. Fig. 1 shows typical decision boundaries and, the values of training and test samples on which decision threshold is applied. There are two subfigures each corresponding to axis-parallel (Fig. 1.a) and oblique tree (Fig. 1.b) respectively. The scalar values corresponding to the training and test samples used at the node of the respective decision trees are plotted along the x-axis. Each '+' mark indicates this scalar value. Its color (green, red, blue and magenta) indicates its class label. The training and test samples are shown at y=1 and y=-1 respectively. The vertical red line indicates the threshold value.

First, let us consider the distribution of training samples in Fig. 1.a and 1.b. APDT divides the training samples magenta and blue to left. The red and green training samples are spread at the right of the vertical line. Oblique decision tree divides red, green training samples to left and the rest to the right. In addition, we can see that Oblique decision tree packs the training samples tightly based



(a) Decision of APDT **(b)** Decision of oblique tree

Fig. 1. Along x-axis these two figures show the attribute values of training and test samples used for decision making. Each '+' mark represents the attribute value of a sample. The samples to the left of the red vertical line are directed towards left and vice-versa. Its color (green, red, blue and magenta) indicates its class label. The training and test samples are displayed at y=1 and y=-1 respectively. By comparing the decision boundaries of test and train samples we can say (b) is a clear case of overfitting.

on the class label. All the training samples belonging to a class have almost same projected value and well separated from others. As one would expect, this indicates that oblique decision tree fits better to the provided training samples.

When we observe the distribution of test samples it reveals the reason for poor accuracy of oblique decision tree. APDT is more consistent with its behavior during training when compared to Oblique decision tree. For Oblique decision tree the test samples are scattered close to the decision boundary and some of them are even misclassified. It seems to be the case of overfitting. This problem cannot be mitigated by using more training samples because even if we increase the training samples the tree nodes at the lower level will receive only a small portion of it. Hence the same problem will appear at the lower levels. Perhaps a combination of using more flexible oblique decision tree nodes at the top level and APDT at lower level may improve the results.

4 Conclusion and Future Work

Our experiments show that we can obtain real-time performance for 3D point recognition with SIFT descriptors if we use ANNS based matching. The accuracy we obtain with such a method is better than binary descriptor based matching even when color and depth information are used. However, we could not improve this performance with SIFT even by using oblique decision trees. This was due to overfitting at lower levels which obtain very few training samples. We would like to mitigate this problem by using oblique decision trees at the top level nodes

(which get large number of training samples) and APDT at lower level. It may also turn out that only NN based matching suits SIFT descriptors and hence using oblique decision trees at top level may also produce misclassification. In that case we would like to use BBD at the top level. BBD structure exponentially reduces the size of the region in the feature space associated with a node as we descend down. This will be similar to applying ANN classification at the top level. Then, APDTs can be used within the bounded regions at the lower levels.

References

1. Affine Covariant Regions data from Visual Geometry Group, Oxford University. http://www.robots.ox.ac.uk/~vgg/research/affine/
2. Matlab functions for multiple view geometry. http://www.robots.ox.ac.uk/~vgg/hzbook/code/
3. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding **110**(3), 346–359 (2008)
4. Bhat. S., Berger, M.O., Sur, F.: Visual words for 3D reconstruction and pose computation. 3DIMPVT 2011 (2011)
5. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
6. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: Computing a Local Binary Descriptor Very Fast. PAMI 2012 (2012)
7. Criminisi, A., Shotton, J.: Decision Forests for Computer Vision and Medical Image Analysis. Springer Publishing Company, Incorporated (2013)
8. Dahl, A.L., Aanæs, H., Pedersen, K.S.: Finding the best feature detector-descriptor combination. In: 3DIMPVT 2011 (2011)
9. Donoser, M., Schmalstieg, D.: Discriminative feature-to-point matching in image-based localization. In: CVPR 2014 (2014)
10. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
11. Hartmann, J., Klussendorff, J., Maehle, E.: A comparison of feature descriptors for visual slam. In: European Conference on Mobile Robots 2013 (2013)
12. Heinly, J., Dunn, E., Frahm, J.-M.: Comparative evaluation of binary features. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) Computer Vision-ECCV 2012. LNCS, vol. 2012, pp. 759–773. Springer, Heidelberg (2012)
13. Hesch, J., Roumeliotis, S.: A direct least-squares (dls) method for pnp. In: ICCV 2011 (2011)
14. Lepetit, V., Fua, P.: Keypoint Recognition using Randomized Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) (2006)
15. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: An Accurate $O(n)$ Solution to the PnP Problem. IJCV 2009 (2009). http://cvlab.epfl.ch/software/EPnP/
16. Leutenegger, S., Chli, M., Siegwart, R.: Brisk: Binary robust invariant scalable keypoints. In: ICCV 2011 (2011)
17. Lieberknecht, S., Benhimane, S., Meier, P., Navab, N.: A dataset and evaluation methodology for template-based tracking algorithms. In: ISMAR (2009)
18. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 2004 (2004)

19. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(10), 1615–1630 (2005)
20. Moosmann, F., Nowak, E., Jurie, F.: Randomized clustering forests for image classification. PAMI 2008 (2008)
21. Mount, D.M., Arya, S.: ANN: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/~mount/ANN
22. Ozuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast Keypoint Recognition using Random Ferns. PAMI 2012 (2012)
23. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. ICCV 2011 (2011)
24. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in rgb-d images. CVPR 2013 (2013)
25. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. ACM Trans. Graph. 2006 (2006)
26. Sun, L., Ji, S., Ye, J.: Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. PAMI 2011 (2011). http://www.public.asu.edu/~jye02/Software/CCA/index.html
27. Trzcinski, V.L.T., Christoudias, M., Fua, P.: Boosting Binary Keypoint Descriptors. Computer Vision and Pattern Recognition 2013 (2013)
28. Wu, C.: Towards linear-time incremental structure from motion. In: 3DV 2013 (2013)
29. Zheng, Y., Kuang, Y., Sugimoto, S., Astrom, K., Okutomi, M.: Revisiting the pnp problem: a fast, general and optimal solution. In: ICCV 2013 (2013)
30. Zhou, Q.-Y., Koltun, V.: Dense scene reconstruction with points of interest. ACM Trans. Graph. 2013 (2013). http://www.stanford.edu/~qianyizh/projects/scenedata.html