ECEN 765 Machine Learning with Networks
Final Report

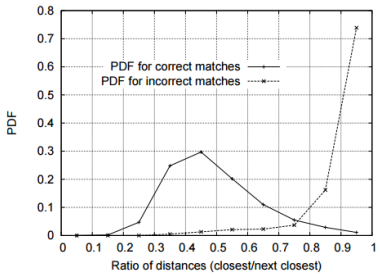
Jicheng Gong

December 13, 2017

1 Sift matching approach

1.1 David Lowe's original method

The approach Sift method used for matching is finding each key point's nearest neighbor from training images. The nearest neighbor is defined as the keypoint with minimum Euclidean distance of the 128 dimension descriptor.
And instead of using a global threshold, a more effective measure is by comparing the distance of the closest neighbor to that of the second-closest neighbor. According to Lowe's experiment, he chose the thresh hold to be 0.8, which has the best overall performance.



1.2 Inner Product method

To eliminate the affection produced by illumination, all the SIFT descriptors are normalized to size of 512. This feature brings us much convenience to calculate its inner product value to get the Euclidean distance.

$$x_1 \cdot x_2 = |x_1| \cdot |x_2| \cdot cos\theta = 512 \cdot 512 \cdot cos\theta = 262144 \cdot cos\theta$$

So the modified matching algorithm will become like:

$$P = x_1 * x_1' + x_2 * x_2' + x_3 * x_3' + ... + x_{128} * x_{128}'$$

x is a descriptor vector from first image and x' is a descriptor vector from second image. P is the inner product. This helps us to convert the original no-linear minimization problem (minimize Euclidean distance) to a maximization problem.

1.3 Dimension Reduction

As has been mentioned in the mid-term report, I want to reduce the dimension of SIFT descriptors. First we create a new vector which is like:

$$[a_1, a_2, ..., a_{128}]^T = [x_1 * x_1', x_2 * x_2', ..., x_{128} * x_{128}']^T$$

Then we have a binary assignment vector $\beta^T = [\beta_1, \beta_2, ..., \beta_{128}]$. If $\beta_i = 1$ means we choose i^{th} dimension while $\beta_i = 0$ means we will discard it. As for dimension reduction,

we will constrain $||\beta||_1$ to the number of dimension we want to keep.
So the reduced inner product value will be

$$P_{reduced} = a^T * \beta, |\beta| = \#dimension$$

2 Feature Selection Algorithms

The feature selection is the process that choose a reduced number of explanatory variable to describe a response variable. The main reasons why feature selection is used are:

1. Make the model easier to interpret, removing variables that are redundant and do not add any information.
2. Reduce the size of the problem to enable algorithms to work faster, making it possible to handle with high-dimensional data.
3. Reduce over-fitting.

In this project feature selection are used mainly for reason 1 and 2. Firstly we want to see whether we can reduce the problems size so that we are able to accelerate the matching process. Then we want to see whether we can gain some meaningful insights from the chosen features.

2.1 Inner product and data preprocessing

The image dataset I used is the KITTI Vision Benchmark Suite. It is a group of consecutive images shot while a car was driving. I modified SIFT algorithm to be able to export descriptors and coordinates for each keypoint in one image. Also export match pairs between two consecutive images as a ground truth to be compared with. The following figures show two groups of descriptors from two consecutive image, they respectively contains m and k descriptors in total.

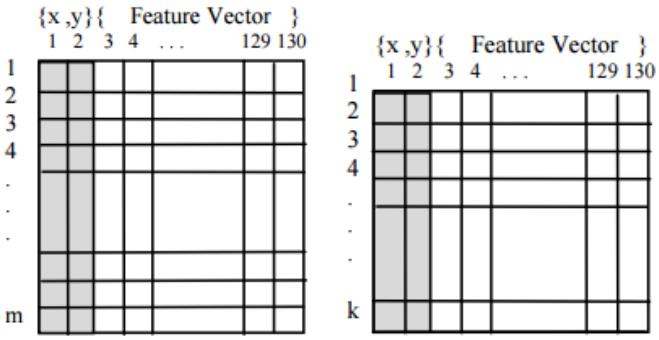


Figure 1: Decriptors extracted by SIFT

The first and second column above represent the coordinate of the keypoint while the remaining 128 columns are the descriptors. Suppose we set matrix M to represent descriptors for first image, matrix K for second image. The inner product matrix P will be:

$$P = M \cdot K^T$$

$P[i,:]$ represents the dot product value of i^{th} keypoint in first image with all the keypoints in second image. Similarly, $P[:,j]$ represents the dot product value of j^{th} keypoint in second image with all the keypoints in first image. The largest dot product in each row will be extracted and compared with the thresh hold. If it is larger than thresh hold then we will claim there is a match.

2.2 Unsupervised feature selection

2.2.1 Variance Threshold

In Geoffry's paper he grouped descriptors into 3 different priority parts according to their deviation, which, in fact is a variance threshold process. In feature selection we calculated each feature's variance of the training data and discard features that has a variance below threshold.

In sklearn it provides such a function sklearn.feature_selection.VarianceThreshold helps remove features with low variance. We can extract its selected features' index by calling get_support. Or just do the transform on the original training data.

2.2.2 PCA based Inner product

In this part we want to use PCA to see the performance if some of the dimensions are reduced. We combine every two consecutive descriptor matrix into one matrix and perform principle component analysis for each combined matrix. Then do the PCA transform for each matrix.

2.3 Supervised feature selection

2.3.1 Labeled training set

Before performing supervised feature selection, we first need to create a set of labeled training samples. Since we have descriptors of each image and match pair's coordinates of every two images, we need to first do the matching and then check whether they are in the list of matching pair.

The matching strategy we use is inner product with threshold 245000 which has been proved has less than 5% error rate comparing with the original matching method. So consider we have two lists of descriptors T_1 and T_2, for each 128 dimension descriptor, we create a diagonal matrix

$$D = \begin{bmatrix} x_1 & 0 & 0 & \dots & 0 \\ 0 & x_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & x_{128} \end{bmatrix}$$

Then we calculate the dot product of T_2 and D then for each row of the matrix will be:

$$\begin{bmatrix} x_1 x_1' & x_2 x_2' & x_3 x_3' & \dots & x_{128} x_{128'} \end{bmatrix}$$

And we check whether the sum of each row is larger than 245000, if yes we push it into the labeled training set with label 1, and if not, we give it a 1/3000 possibility to be pushed to the training set with label 0. The reason why I do so is because there are much more unpaired vectors than paired vectors, the low possibility can make label 1 and label 2 almost equal to each other.

2.3.2 LASSO

LASSO (Least Absolute Shrinkage and Selection Operator) is a supervised feature selection method which puts a constraint on the sum of the absolute values of the model parameters, the sum has to be less than a fixed value (upper bound). In order to do so the method apply a shrinking (regularization) process where it penalizes the coefficients of the regression variables shrinking some of them to zero. During features selection process the variables that still have a non-zero coefficient after the shrinking process are selected to be part of the model. The goal of this process is to minimize the prediction

error.

The cost function of LASSO is the sum of squared errors and we add a constraint that the model parameter should smaller than certain value.

$Minimize(\frac{||Y-X\beta||_2^2}{n})$, which is subject to $\sum_{j=1}^{k} ||\beta||_1 < t$

And this is equivalent to

$$\beta(\lambda) = argmin_\beta(\frac{||Y - X\beta||_2^2}{n} + \lambda||\beta||_1)$$

Where λ is the penalizing parameter, the larger λ is, the smaller β will be.

As what was mentioned in the class, the difference between Ridge Regression and LASSO method is that Ridge Regression use l2 norm. For the LASSO method the constraint region is a diamond, therefore it has corners. This means that if the first point is in proximity of the corner, then it has one coefficient β_j equal to zero. While for the Ridge Regression method the constraint region is a disk, thus it has no corners and the coefficients can not be equal to zero. So LASSO can be used to do feature selection while Ridge Regression cannot be used.

In this project, I implemented LASSO by calling sklearn.linear_model.Lasso(). The feature importance can be extracted by calling lasso.coef_. We train Lasso by giving the training samples generated as described in last part.

2.3.3 Tree based Feature Selection

Since Random Forest algorithm hasn't been introduced in details during the class, I can only give a non-mathematical introduction. Decision trees are one of the most popular methods used for inductive inference. They are robust for noisy data and capable of learning disjunctive expressions. A decision tree is a k-ary tree where each of the internal nodes specifies a test on some attributes from the input feature set used to represent the data. Each branch descending from a node corresponds to one of the possible values of the feature specified at that node. And each test results in branches, which represent different outcomes of the test. The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner.

In this project I have used Random Forest algorithm to do feature selection by calling sklearn.ensemble.RandomForestClassifier. Again, the training samples are from last part.

2.4 Evaluation Metrics

To evaluate the matching,we use recall vs. 1-precision. Recall will measure the ratio between the number of correct matches retrieved over the total of commit matches. As we can achieve a 100% of recall by returning a set with all possible matches, we notice that the recall measure is not enough; therefore, we also calculate the imprecision(1-precision). The precision measures the ratio between the quantity of correct retrieved matches over the number of retrieved matches, and the imprecision measures the ratio between the number of false retrieved matches over the total number of retrieved matches. Consequently we will realize that the Recall vs. 1-precision curve shows adequately the trade off to obtain.

$$Recall = \frac{Correctmatchesretrieved}{Totalnumberofcorrectmatches}$$

$$1 - Precision = \frac{Incorrectmatchesretrieved}{Totalofmatchesretrieved}$$

Besides, I also calculate their E/R ratio value so that it will shows the accuracy of the recalled matches. It is the key evaluation because what we really care is their accuracy.

3 Experiment result

Table 1: Comparison of Inner Product, PCA and VarianceThreshold

Method	Dimension	Error Rate	Recall Rate	E/R rate
Inner Product		4.2%	91.5%	0.046
Variance Threshold	80	5.7%	22.9%	0.25
Variance Threshold	64	6.1%	20.4%	0.29
Variance Threshold	32	10.5%	22.1%	0.48
Variance Threshold	16	16.8%	29.5%	0.57
PCA	80	4.3%	23.4%	0.18
PCA	64	11.1%	51.7%	0.21
PCA	32	8.6%	24.1%	0.35
PCA	16	15.1%	29.2%	0.51

3.1 Inner product

In the experiment I used 114 consecutive images to do SIFT matching. Figure 2 3 14 shows Error rate and Recall rate of the original inner product method with different threshold. We can see that 245000 threshold has almost the same accuracy as 250000 while it has a higher Recall rate.

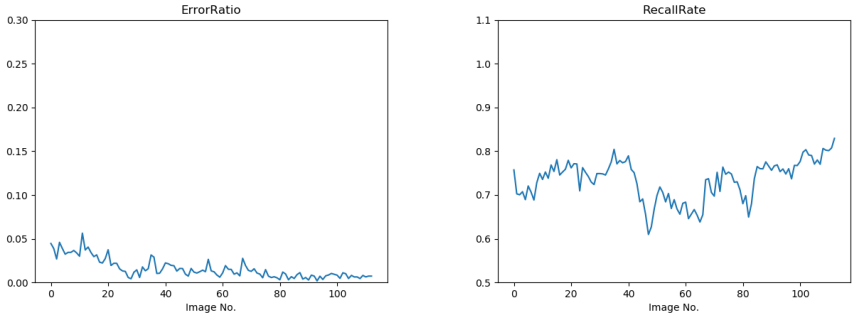


Figure 2: Error rate and Recall with threshold 250000

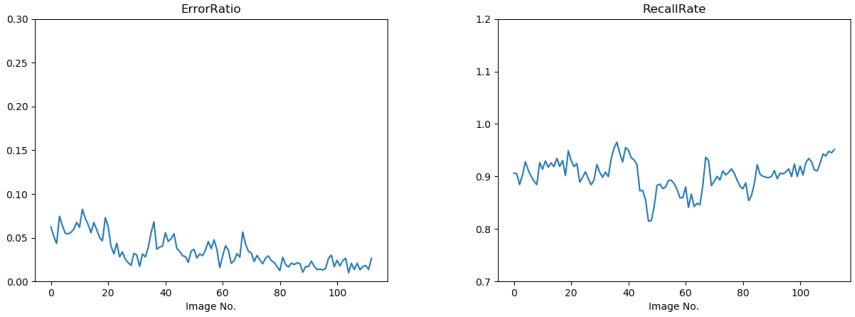


Figure 3: Error rate and Recall with threshold 245000

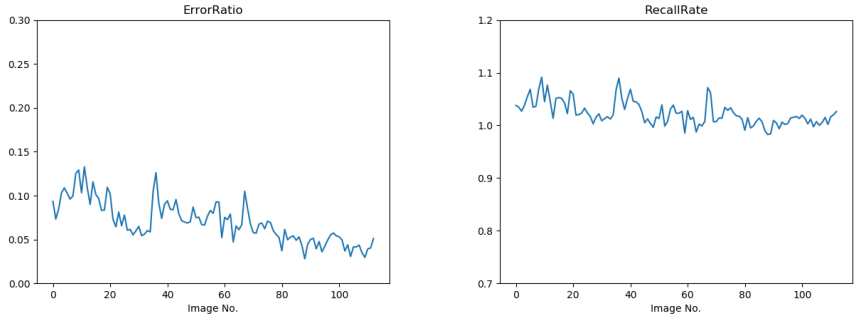


Figure 4: Error rate and Recall with threshold 240000

3.2 PCA

In this part I try to use PCA to reduce the dimension first. However, with the reduction of dimension, we should also decrease the threshold, too.

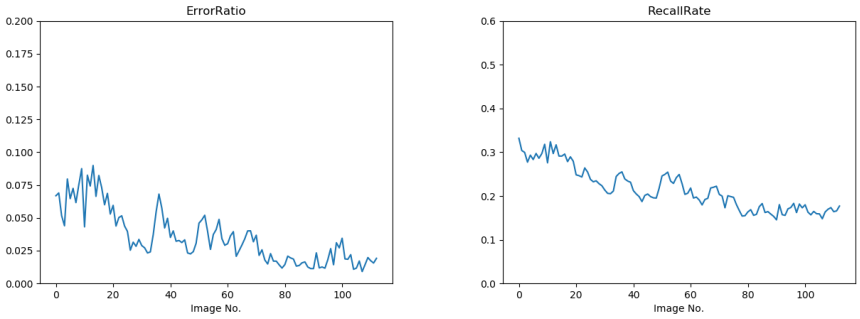


Figure 5: 80 dimension Error rate and Recall with threshold 160000

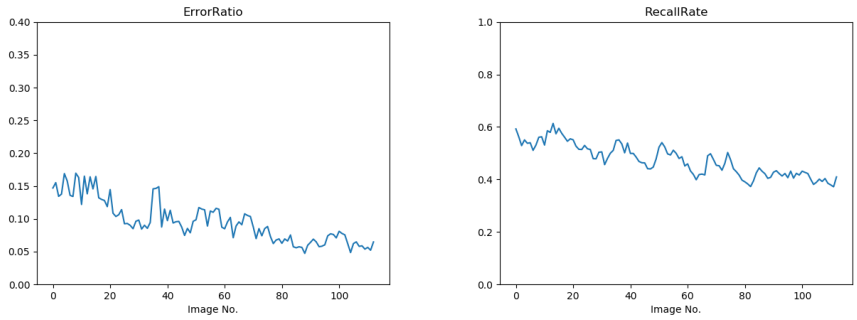


Figure 6: 64 dimension Error rate and Recall with threshold 140000

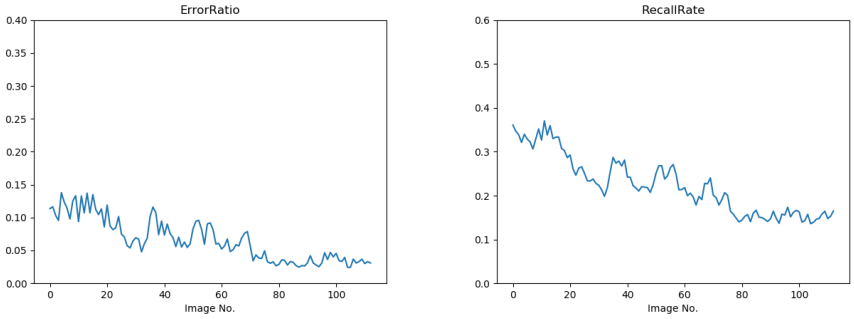


Figure 7: 32 dimension Error rate and Recall with threshold 140000

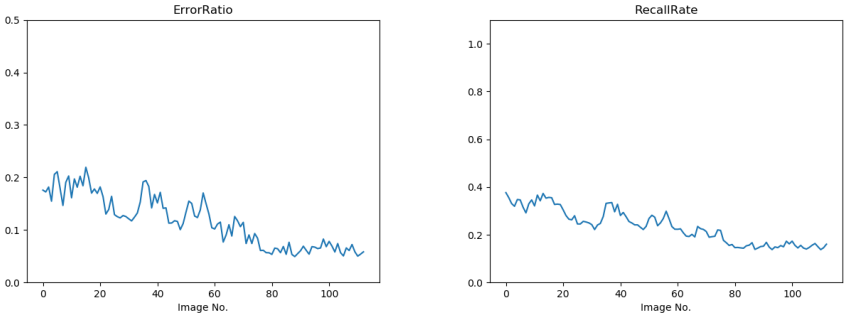


Figure 8: 16 dimension Error rate and Recall with threshold 120000

We may find that with fewer dimensions, the error rate will increase, otherwise the recall rate will drop vastly. What's more, the data set seems to have some bias because both error rate and recall rate drops in last a few images. Which indicates that a universal threshold may not work for this problem.

3.3 LASSO and RandomForest

Table 2: Comparison of LASSO and RandomForest

Method	Dimension	Error Rate	Recall Rate	E/R rate
Inner Product		4.2%	91.5%	0.046
LASSO	80	6.1%	38.1%	0.16
LASSO	64	4.2%	20.3%	0.20
LASSO	32	4.8%	13.7%	0.35
Random Forest	80	7.8%	18.6%	0.42
Random Forest	64	9.2%	21.3%	0.43
Random Forest	32	12.1%	18.5%	0.65

3.4 LASSO

Due to huge amount of matching pairs with reduce the algorithm efficiency a lot, I only trained first 3 images and extract important dimensions from these images. The number of training samples is about 10,000. And the experiment shows that the number of training images won't affect the result too much.

High ranking features: 19, 18 ,42, 15, 90, 83, 68, 54, 94, 65, 13, 38, 14, 45, 39, 106, 111, 57, 51, 56, 44, 43, 20, 53, 113, 127, 77, 7, 32, 124, 72 ,118.

The λ value for LASSO is 30. According to my observation, the larger λ is, the more accurate the result will be, but feature ranking value will be smaller, a lot features might have 0 contribution.

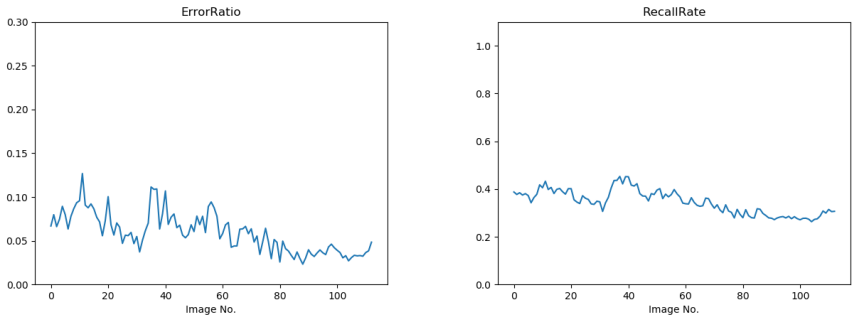


Figure 9: 80 dimension error rate and recall rate with threshold 195000

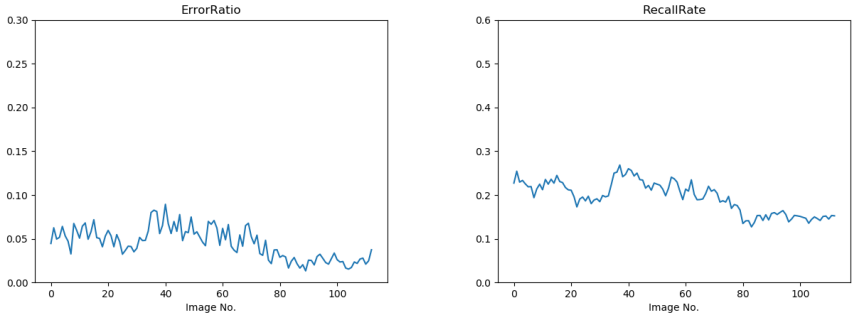


Figure 10: 64 dimension error rate and recall rate with threshold 165000

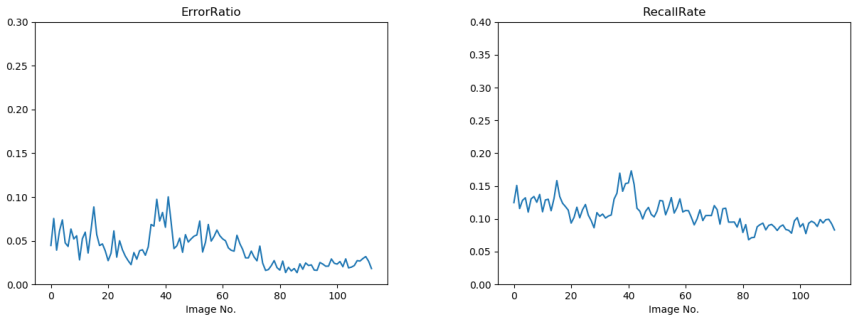


Figure 11: 32 dimension error rate and recall rate with threshold 93000

3.5 Random Forest

High ranking features: 72, 48, 92, 76, 87, 32, 56, 44, 47, 79, 55, 81, 73, 40, 95, 65, 46, 80, 86, 45, 75, 49, 85, 127, 41, 68, 39, 52, 50, 57, 84, 42.

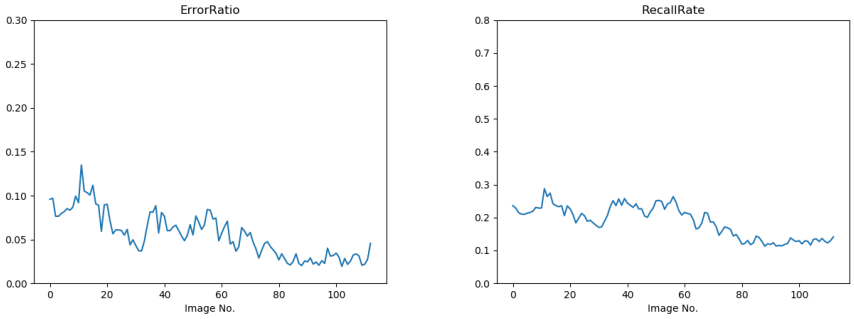


Figure 12: 80 dimension error rate and recall rate with threshold 220000

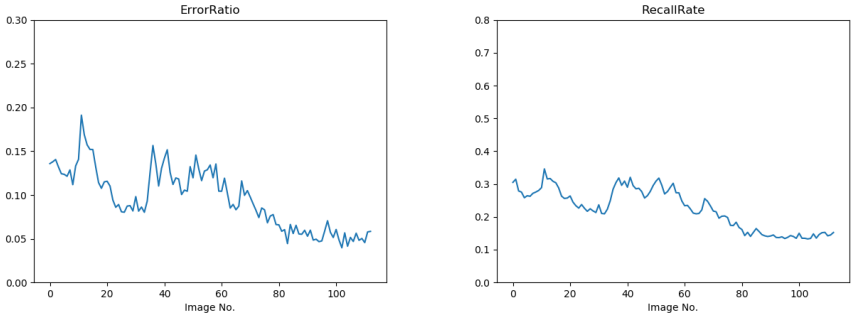


Figure 13: 64 dimension error rate and recall rate with threshold 200000

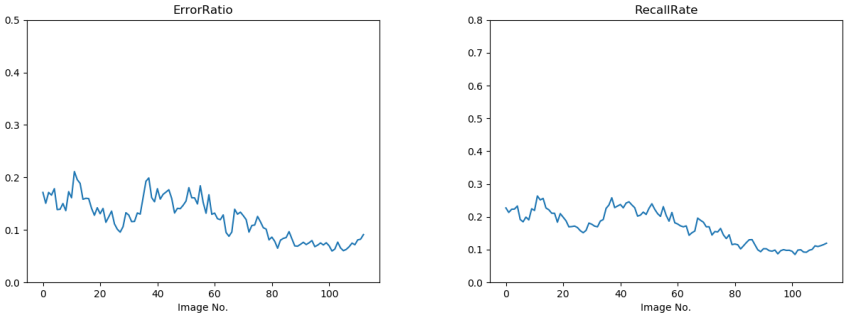


Figure 14: 32 dimension error rate and recall rate with threshold 170000

4 Result Analysis

According to the result above we first know that Inner product can be a good replacement for Euclidean distance because it will have 91.5% recall rate while only 4.2% of them are different. Considering SIFT will also make mistakes, it is possible that Inner Product can be a even better method.

The two unsupervised methods yielded similar results while PCA is slightly better than Variance Threshold if we are able to find a good threshold value.

As to the supervised method, LASSO and Random Forest share some similar hight ranking features however the E/R value tells that LASSO performs much better than Random Forest. Although the training set only include first 4 images, LASSO still has the highest E/R rate even comparing with PCA. If perform LASSO on more general training set, we might have an even better result.

5 Conclusion

The result shows LASSO is so far the best result. This method again can be used as a putative method to first eliminate most wrong matches. Then the left descriptors will be calculated by Euclidean Distance to achieve a higher accuracy and calculation efficiency.

References

[1] Geoffrey Treen *Methods for improved SIFT matching.* Ottawa, Ontario Novemenber, 2009

[2] Yan Ke, Rahul Sukthankar *PCA-SIFT: A more distinctive representation for local image descriptors*

[3] David G. Lowe *Distinctive image features from scale-invariant keypoints*

[4] Valeria Fonti *Feature selection using LASSO*