Filtering Faces
Privacy Protection by Feature Selection

Artur Filipowicz
arturf@princeton.edu
Princeton University

January 21, 2016

Abstract

This paper examines the use of filters on feature vectors for privacy protection in facial recognition. Feature vectors are the results of Fast Fourier Transform and Wavelet Transform on the Yale and Olivetti datasets. Several filters are proposed. Filters based on the signal to noise ratio and t test select feature which prevent privacy compromising reconstruction without sacrificing accuracy. The use of phase removal for FFT and normalization are also shown to protect privacy.

1 Introduction

In modern times, the existence of both terrorist threats and advance digital technology created both the need and the possibility of mass surveillance. A critical competent of such system is biometric identification of people, mainly by facial recognition. As noted in [3], the events of 9/11, Edward Snowden incident and other events have resulted in both a demand for recognition systems and a concern for privacy violation by such systems. While a lot of research has gone into improving facial recognition systems - [1], [9], [2], [4], [8] among others - relatively little research has been done on incorporating privacy into such systems; some examples being [5], [7], and [6].

The primary approach to privacy in facial recognition is cryptography. In [5], Eigenfaces recogition system is used on homomorphically encrypted data. In this first cryptographic system for facial recognition [5], the client wants the server to identify a face without revealing the image to the server. The server also does not want to reveal the contents of it's database. In this approach, data is quantized for Pailler encryption and server and client share computations needed for matching faces. [5] Experimentally, 96% accuracy

was achieved in [5] on "ORL Database of Faces". [7] improved the algorithm presented in [5] by reducing the time and space complexity with the use of garbled circuits.

Along a different line of research, [6] used Helper Data Systems to provide privacy. The approach generates binary feature vector by determining reliable bits based on statistics from sample data. In the following paper, I investigate a similar approach to privacy. However, in my proposed approach no encryption or quantization is necessary. In my approach there are not additional processes. Privacy is built into the classification process.

This approach rests on the idea that privacy is compromised when an image can be visually inspected by an individual. Therefore, as long as the reconstruction of an image from a feature vector is not meaningful to a human, privacy has been maintained. Facial recognition systems often utilize Fast Fourier Transform (FFT) or Wavelet Transform (WT) as part of feature engineering. For example [9], [2], [4], and [8] use FFT or WT, their systems can be seen in Figure 1 and 2 and classification accuracies are provided in Figures 3 and 4. I investigate if in recognition systems like these, it is possible to alter the output of FFT or WT to reduce the quality of the reconstructed image without sacrificing accuracy of the classification.

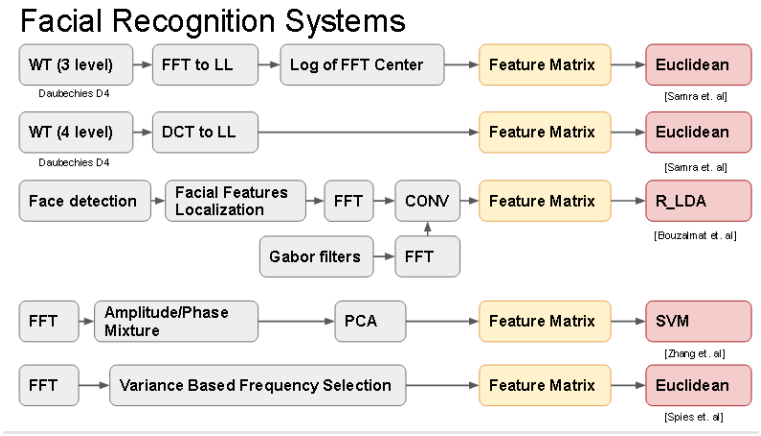


Figure 1

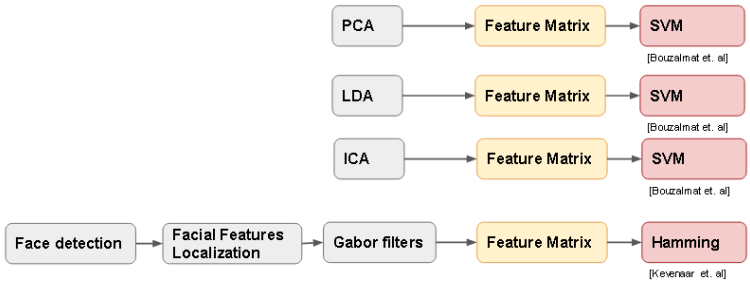


Figure 2

System	Recognition Rate (best)	Database
WT-FFT	96.36%	Yale
WT-FFT	66.25%	Olivetti
WT-DCT	78.18%	Yale
FFT-GABOR-CONV	na	XM2VTSDB
PCA-SVM	84.21%	Yale
FFT-PCA-SVM	93.42%	Yale
FFT-VAR	98%	Olivetti
Eigenface	74%	Yale
Eigenface	70.8%	Olivetti
Template Method	78%	Yale
Template Method	73.33%	Olivetti

Figure 3

SystemC	Recognition Rate (best)	Database
PCA-SVM	90.2%	ATT
LDA-SVM	93.9%	ATT
ICA-SVM	91%	ATT
PCA-SVM	66.8%	IFD
LDA-SVM	70%	IFD
ICA-SVM	62.9%	IFD
FaceDetection-Gabor	97.5%	FERET
FaceDetection-Gabor	99.75%	Ca;tech

Figure 4

2 My Classification Systems

As picture in Figure 5, my classification systems for this investigation, which are very similar, begin with an application of FFT or WT to an image. Then a filter is applied as part of feature selection. Classification is accomplished with an SVM. For all of the following experiments an SVM with a leaner kernel and C = 1 was found to produce the best results.

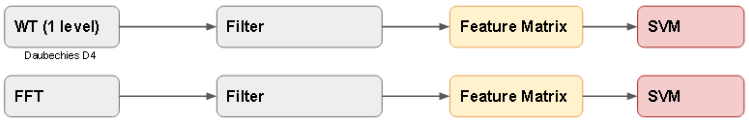


Figure 5

3 Filters

A filter is a binary matrix which selects features based on some criteria. Let $F_{x,d}$ be a filter $F_{x,d} \in \{0,1\}^{dxm}$ where m is the number of features in each example, x is one of the filter types, and d is the number of features selected by the filter.

$F_{x,d,i,j} = 1$ if $W_x(j)$ is with in the d largest W_x, otherwise $F_{d,i,j} = 0$.

Let X be an example: $X_{filtered} = F_{x,d}X$

The following Python code illustrates the mathematical definition above. In this code, 151 examples from X_nophase are used to compute the variance of each feature, with each feature being the amplitude of a frequency from FFT. Then the 100 features with the highest variance are selected to pass thought the filter.

```
def varFilter(train_X, numFeatures):
    F = np.zeros(train_X.shape[0])
    var = np.var(train_X, axis=1)
    varSorted = np.argsort(var)[::-1]

    F[varSorted[0:numFeatures]] = 1

    return F

(X,Y) = loadYaleData()
```

```
X_nophase = np.zeros([X.shape[0]/2, X.shape[1]])
for i in range(0,X.shape[1]):
   X_nophase[:,i] = removePhase(X[:,i])[:]

F = varFilter(X_nophase[:,0:151], 100);

X_filtered = X[np.append(F, np.ones(X.shape[0]/2), axis=0) != 1]
```

Variance is only one of the several filters investigated, and the only unsupervised filter. The other filters include rectangle and triangle filters, inspired by results in [9], which are independent of the training data, and 4 supervised filters based on the signal to noise ratio, Fisher discriminant ratio, symmetric divergence and t test. The supervised filters require a labels for positive and negative classes. During training, for each individual the training examples are divided into two classes. Positive class contains the pictures of that individual and the negative class contains all other pictures. The signal to noise ratio, Fisher discriminant ratio, symmetric divergence or t test are computed based on those two classes for each feature, and the final weight for that feature is the mean of weights cross all of the individual in the training set.

The filters are mathematically defined below, expect for the triangle and rectangle filters, for which Python code was a much more succinct form of definition. For the equations below, let μ_j^+, σ_j^+, and N_j^+ be the mean, standard deviation and number of examples for the target person and let μ_j^-, σ_j^-, and N_j^- be the mean, standard deviation and number of examples for all other people in the database.

3.1 Variance

$$W_{VAR}(j) = \sigma_j^2 \tag{1}$$

3.2 Signal to Noise Ration

$$W_{SNR}(j) = \frac{\overline{|\mu_j^+ - \mu_j^-|}}{\sigma_j^+ + \sigma_j^-} \tag{2}$$

3.3 Fisher Discriminant Ratio

$$W_{FDR}(j) = \frac{\overline{(\mu_j^+ - \mu_j^-)^2}}{(\sigma_j^+)^2 + (\sigma_j^-)^2} \tag{3}$$

3.4 Symmetric Divergence

$$W_{SD}(j) = \sqrt{\frac{1}{2}\frac{(\mu_j^+)^2}{(\mu_j^-)^2} + \frac{(\mu_j^-)^2}{(\mu_j^+)^2} + \frac{1}{2}\frac{(\mu_j^+ - \mu_j^-)^2}{(\sigma_j^+)^2 + (\sigma_j^-)^2} - 1} \tag{4}$$

3.5 T

$$W_T(j) = \sqrt{\frac{\mu_j^+ - \mu_j^-}{\sqrt{\frac{(\sigma_j^+)^2}{N_j^+} + \frac{(\sigma_j^-)^2}{N_j^-}}}} \tag{5}$$

3.6 Rectangle

```
def recFilter(w,h,fw,fh):
    h = h + 1
    F = np.zeros((h,w));
    for i in range((w-1)/2+1 - fw/2, (w-1)/2+1 + fw/2 - 1):
        for j in range(h - fh - 1, h):
            F[j,i] = 1
    return np.reshape(F,(w*h),order='F')
```

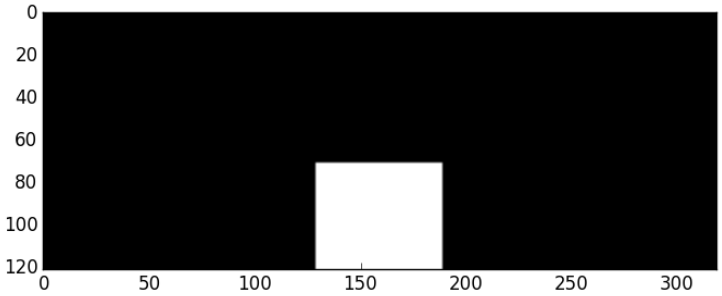


Figure 6: Picture of a rectangle filter. White corresponds to 1.

3.7 Triangle

```
def triFilter(w,h,fw,fh):
    h = h + 1
    F = np.zeros((h,w));
    for j in range(h - fh -1, h):
        span = (j - (h - fh)) * (fw/2)/fh;
        for i in range((w-1)/2+1 - span, (w-1)/2+1 + span - 1):
```

```
        F[j,i] = 1
return np.reshape(F,(w*h),order='F')
```

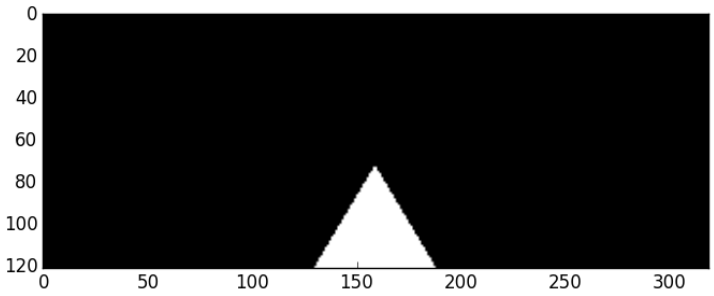


Figure 7: Picture of a triangle filter. White corresponds to 1.

4 Method

4.1 Classification

Ten fold cross validation was used to test classification accuracies. Grid Search was used to optimize the SVM parameters for each fold. For all of the following experiments an SVM with a leaner kernel and C = 1 was found to produce the best results. 100 cross validations were performed and the training of supervised and unsupervised filters was constrained to appropriate fold. Since the rectangle and triangle filters are parameterized they were used to determine d for all other filters. For FFT, the filters where only applied to the amplitudes of frequencies. For WT, the filters were applied to all four maps.

4.2 Reconstruction

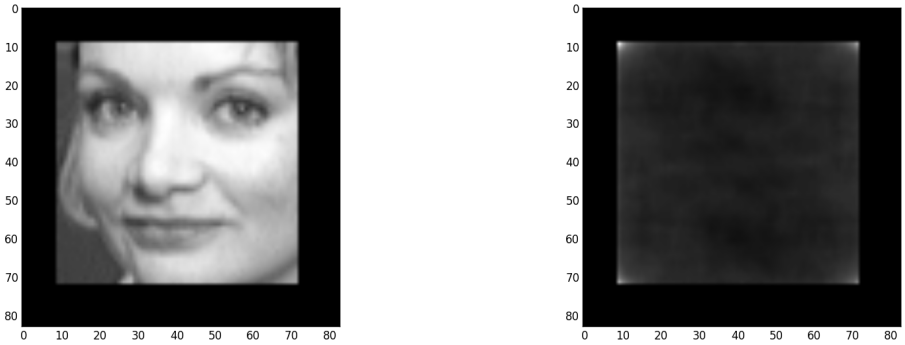
To reconstruct original images from filtered feature vectors I set the values of all parameters which were filtered out to be zero.

5 Experimental Results

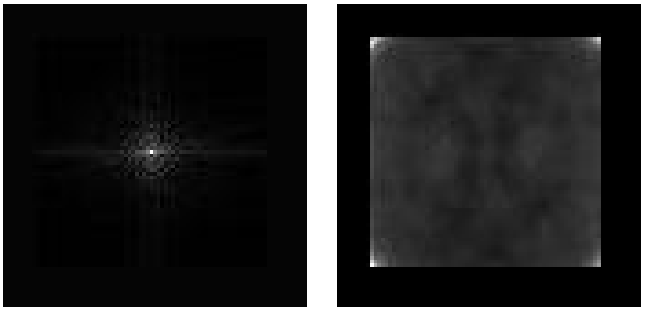
The baseline accuracy for Yale with FFT transform is 0.741 and 0.735 when the phase is removed. The baseline for Olivetti with FFT transform is 0.975 and 0.9737 with the phase removed. The baseline for Yale with WT transform is 0.807. The baseline for Olivetti with WT transform is 0.963.

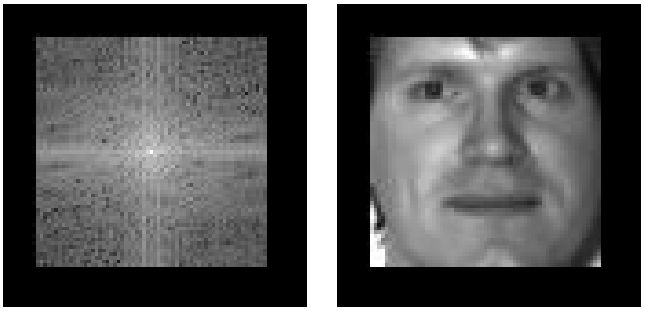
5.1 Basic Observations

FFT produces a amplitude and phase for the transformed image. If the phase is removed and the image is reconstructed based on the amplitude the result is a faceless image. The image on the left is of the original, while the image on the right is the reconstruction.



A similar phenomena happens when the result of FFT is normalize to zero mean and unit variance. Top images are the spectrum and reconstruction of a normalized FFT output. The bottom images show the spectrum and reconstruction for the same image with the original mean and variance restored.





5.2 Yale Database FFT Accuracy

Mean Accuracy for Filters on Yale with Phase

	Dimensions					
Filter	2115	2224	2415	2614	2915	3072
Rect	0.689	0.725	0.737	0.738	0.741	0.741
Var	**0.724**	**0.728**	0.739	0.741	0.743	0.743
SNR	0.681	0.720	0.739	0.747	0.748	0.746
FDR	0.687	**0.728**	0.735	0.744	0.748	0.745
SD	0.585	0.649	0.686	0.711	0.729	0.737
T	0.685	0.718	**0.741**	**0.749**	**0.750**	**0.747**

Table 1: Mean accuracy for filters on Yale with phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .07.

Mean Accuracy for Filters on Yale with Phase (Triangle Filter)

	Dimensions					
Filter	2048	2088	2178	2258	2408	2528
Tri	**0.676**	0.713	**0.732**	0.732	0.736	0.74
Var	0.658	**0.726**	0.727	**0.733**	0.739	0.740
SNR	0.485	0.657	0.706	0.719	0.739	0.743
FDR	0.512	0.657	0.722	0.730	0.734	0.740
SD	0.478	0.547	0.624	0.663	0.689	0.694
T	0.487	0.654	0.707	0.725	**0.740**	**0.744**

Table 2: Mean accuracy for filters on Yale with phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .07.

Mean Accuracy for Filters on Yale without Phase

	Dimensions					
Filter	99	208	399	598	899	1056
Rect	0.689	0.725	0.737	0.738	0.741	0.741
Var	**0.724**	**0.728**	0.739	0.741	0.743	0.743
SNR	0.681	0.720	0.739	0.746	0.748	0.746
FDR	0.687	**0.728**	0.735	0.744	0.748	0.745
SD	0.585	0.649	0.686	0.711	0.729	0.737
T	0.685	0.718	**0.741**	**0.749**	**0.750**	**0.747**

Table 3: Mean accuracy for filters on Yale without phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .07.

Mean Accuracy for Filters on Yale without Phase (Triangle Filter)

	Dimensions					
Filter	32	72	162	242	392	512
Tri	**0.676**	0.713	**0.732**	0.732	0.736	0.74
Var	0.658	**0.726**	0.727	**0.733**	0.739	0.740
SNR	0.485	0.657	0.706	0.719	0.739	0.743
FDR	0.512	0.657	0.722	0.731	0.734	0.740
SD	0.478	0.547	0.624	0.663	0.689	0.694
T	0.487	0.654	0.707	0.725	**0.740**	**0.744**

Table 4: Mean accuracy for filters on Yale without phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .07.

5.3 Olivetti Database FFT Accuracy

Mean Accuracy for Filters with Phase Olivetti

	Dimensions					
Filter	2115	2224	2415	2614	2915	3072
Rect	**0.971**	**0.967**	0.967	0.967	**0.968**	0.967
Var	0.968	**0.967**	**0.970**	0.967	**0.968**	**0.969**
SNR	0.968	**0.967**	0.968	**0.968**	**0.968**	0.968
FDR	0.967	0.966	0.965	0.965	**0.968**	0.968
SD	0.898	0.936	0.952	0.962	**0.968**	0.967
T	0.967	0.966	0.968	0.967	**0.968**	0.968

Table 5: Mean accuracy for filters on Olivetti with phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .03.

Mean Accuracy for Filters on Olivetti with Phase (Triangle Filter)

	Dimensions					
Filter	2048	2088	2178	2258	2408	2528
Tri	**0.964**	**0.969**	0.968	0.967	0.967	0.968
Var	0.959	0.968	**0.969**	**0.968**	**0.970**	**0.969**
SNR	0.936	0.964	0.966	0.967	0.968	0.968
FDR	0.943	0.967	0.967	0.966	0.965	0.965
SD	0.812	0.873	0.930	0.940	0.952	0.961
T	0.932	0.966	0.967	0.967	0.968	0.967

Table 6: Mean accuracy for filters on Olivetti with phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .03.

Mean Accuracy for Filters on Olivetti without Phase

	Dimensions					
Filter	99	208	399	598	899	1056
Rect	0.971	0.967	**0.967**	0.967	**0.968**	0.967
Var	**0.968**	**0.967**	**0.970**	0.967	**0.968**	**0.969**
SNR	**0.968**	**0.967**	0.968	**0.969**	**0.968**	0.968
FDR	0.967	0.966	0.965	0.965	**0.968**	0.968
SD	0.898	0.936	0.952	0.962	**0.968**	0.967
T	0.967	0.966	0.968	0.966	**0.968**	0.968

Table 7: Mean accuracy for filters on Olivetti without phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .03.

Mean Accuracy for Filters on Olivetti without Phase (Triangle Filter)

	Dimensions					
Filter	32	72	162	242	392	512
Tri	**0.964**	**0.969**	0.968	0.967	0.967	0.968
Var	0.959	0.968	**0.969**	**0.968**	**0.970**	**0.969**
SNR	0.936	0.964	0.966	0.967	0.968	0.968
FDR	0.943	0.967	0.967	0.966	0.965	0.965
SD	0.812	0.873	0.930	0.940	0.952	0.961
T	0.932	0.966	0.967	0.967	0.968	0.967

Table 8: Mean accuracy for filters on Olivetti without phase based on 100, 10 fold cross validations. The standard deviation for each measurement is about .03.

5.4 Yale Database WT Accuracy

Mean Accuracy for Filters on Yale with Wavelette

	Dimensions					
Filter	99	208	399	598	899	1056
Var	0.498	0.647	0.737	0.782	0.804	0.800
SNR	0.754	0.791	**0.813**	**0.812**	0.815	0.817
FDR	0.758	**0.792**	0.807	**0.812**	**0.823**	**0.820**
SD	0.420	0.524	0.672	0.725	0.776	0.783
T	**0.760**	0.783	0.812	**0.812**	0.816	0.816

Table 9: Mean accuracy for filters on Yale with wavelette transform based on 100, 10 fold cross validations. The standard deviation for each measurement is about .07.

5.5 Olivetti Database WT Accuracy

Mean Accuracy for Filters on Olivetti with Wavelette

	Dimensions					
Filter	99	208	399	598	899	1056
Var	0.752	0.853	0.941	0.971	0.978	0.980
SNR	0.946	0.957	0.968	0.974	0.984	0.985
FDR	0.934	0.959	0.969	**0.978**	**0.985**	**0.987**
SD	0.762	0.811	0.883	0.919	0.947	0.958
T	**0.950**	**0.960**	0.969	0.972	0.984	0.986

Table 10: Mean accuracy for filters on Olivetti with wavelette transform based on 100, 10 fold cross validations. The standard deviation for each measurement is about .03.

5.6 Yale Database FFT Accuracy Graphs

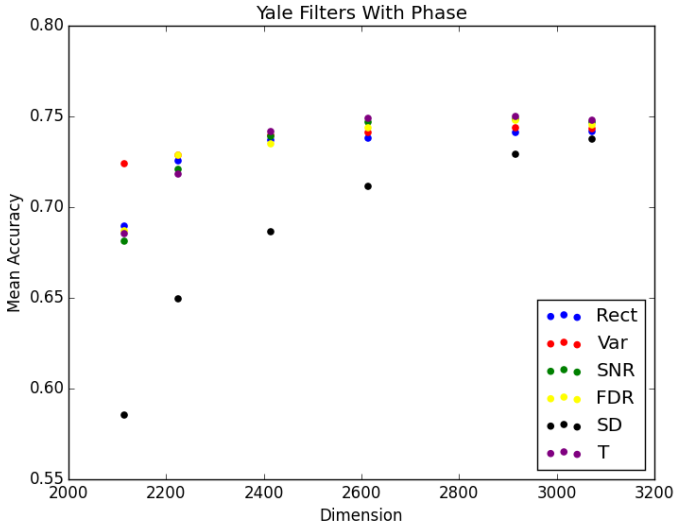


Figure 8

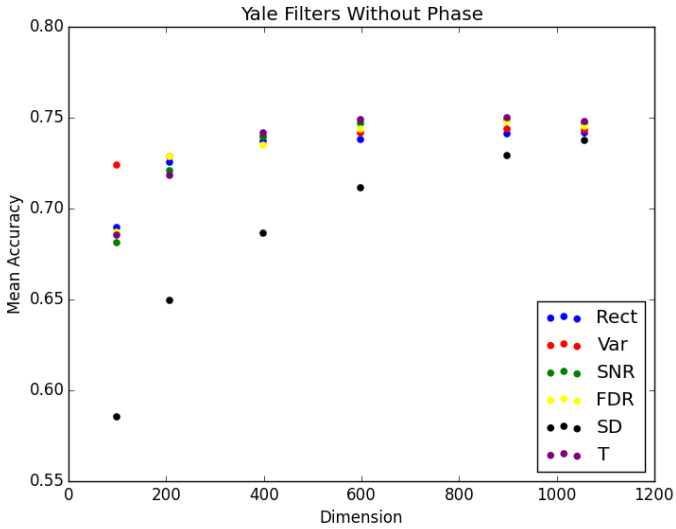


Figure 9

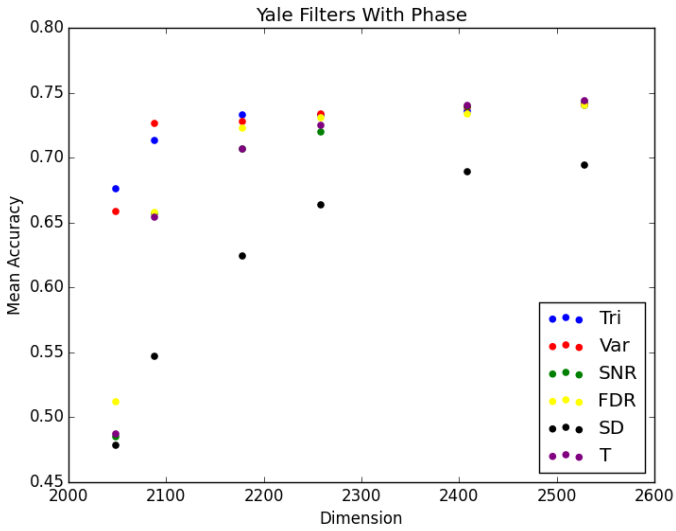


Figure 10

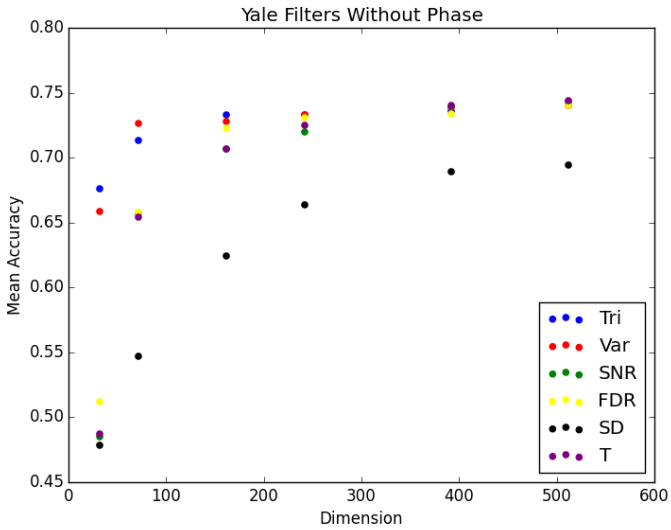


Figure 11

5.7 Olivetti Database FFT Accuracy Graphs

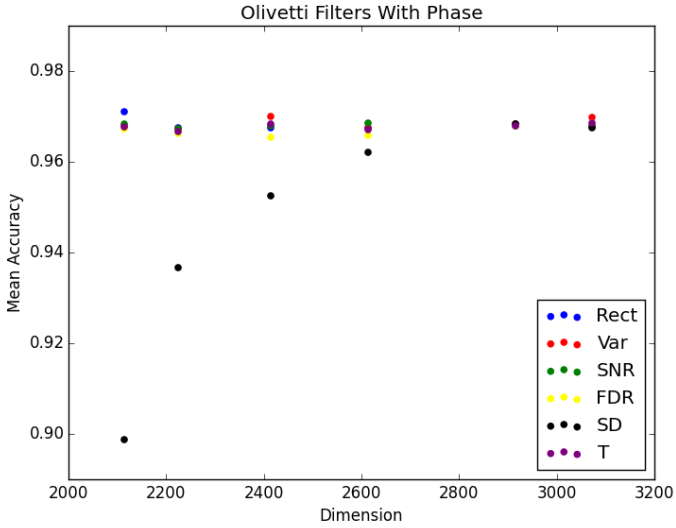


Figure 12

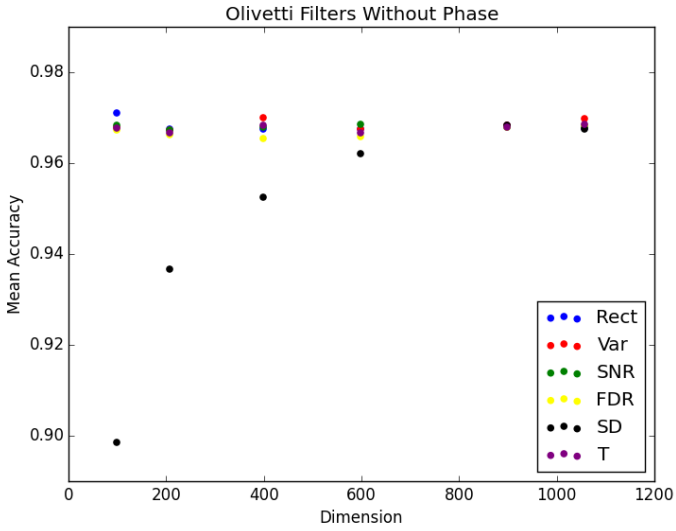


Figure 13

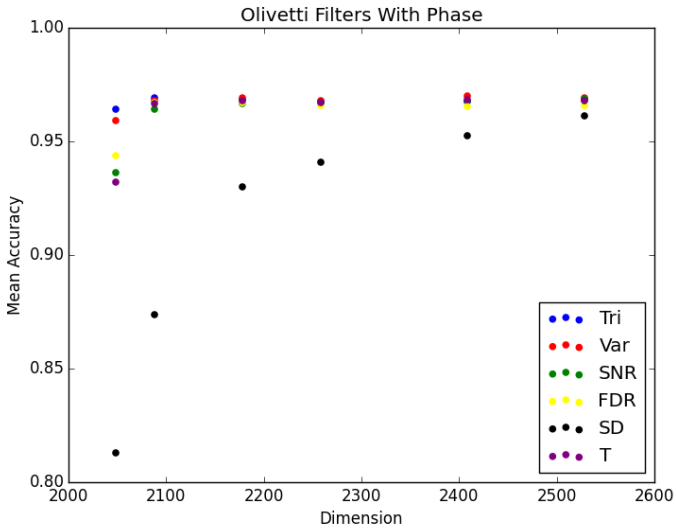


Figure 14

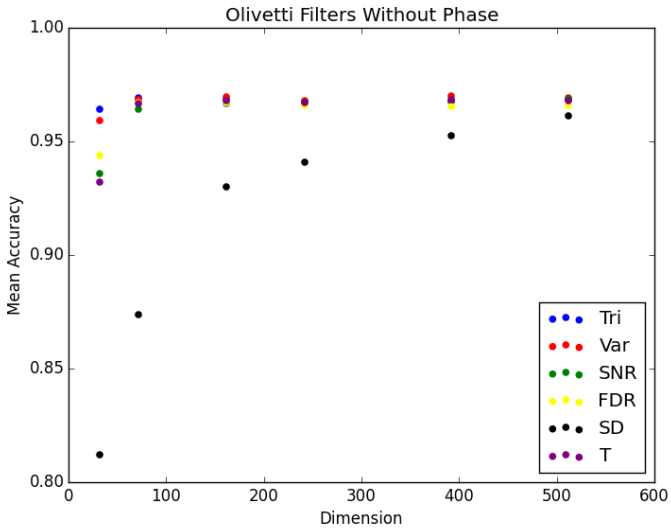


Figure 15

5.8 Olivetti Database Wavelet Accuracy Graphs

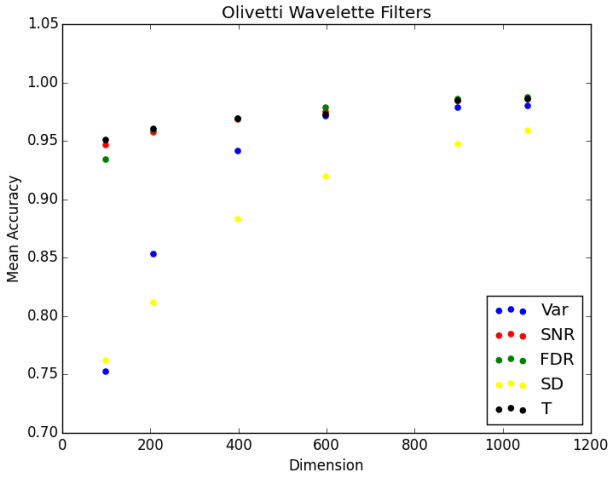


Figure 16

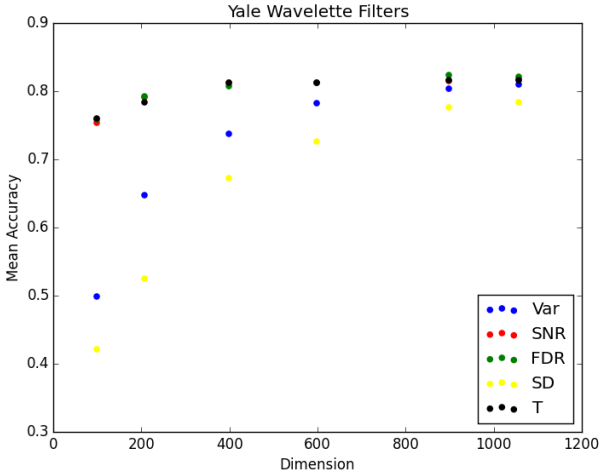
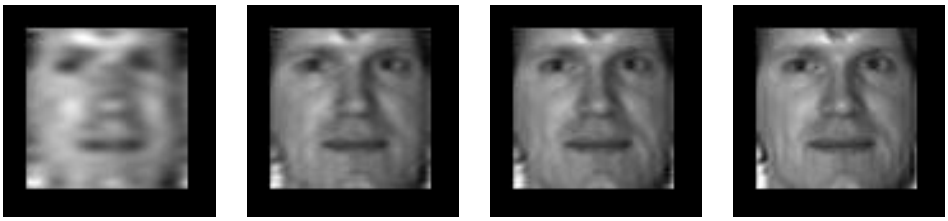
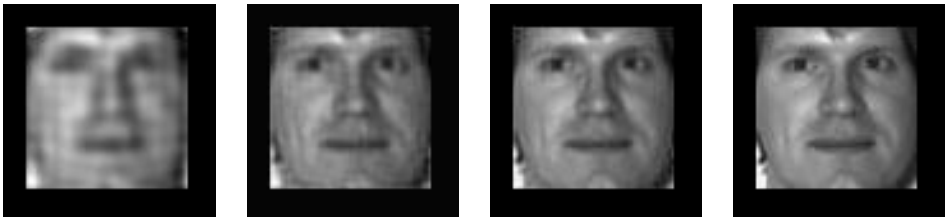


Figure 17

5.9 Yale FFT Reconstructions



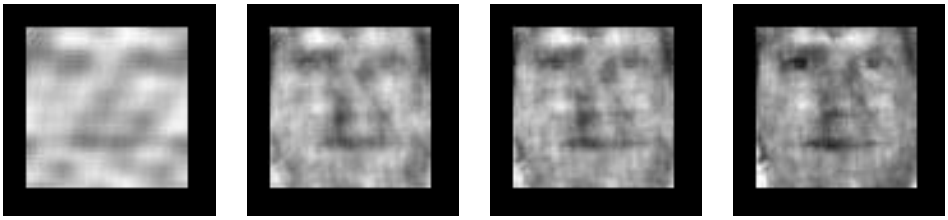
(a) Rect 99 0.68 (b) Rect 399 0.73 (c) Rect 598 0.78 (d) Rect 1056 0.74



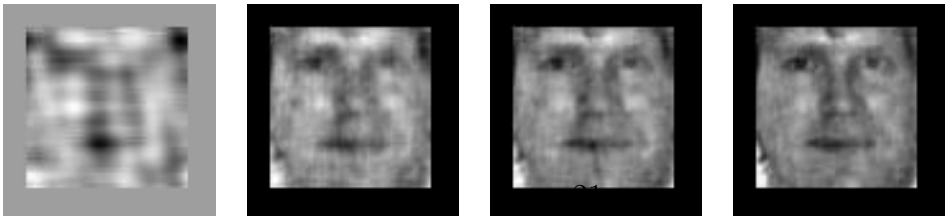
(e) Var 99 0.724 (f) Var 399 0.739 (g) Var 598 0.741 (h) Var 1056 0.743

(i) SNR 99 0.681 (j) SNR 399 0.739 (k) SNR 598 0.741 (l) SNR 1056 0.74

(m) FDR 99 0.68 (n) FDR 399 0.73 (o) FDR 598 0.74 (p) FDR 1056 0.74



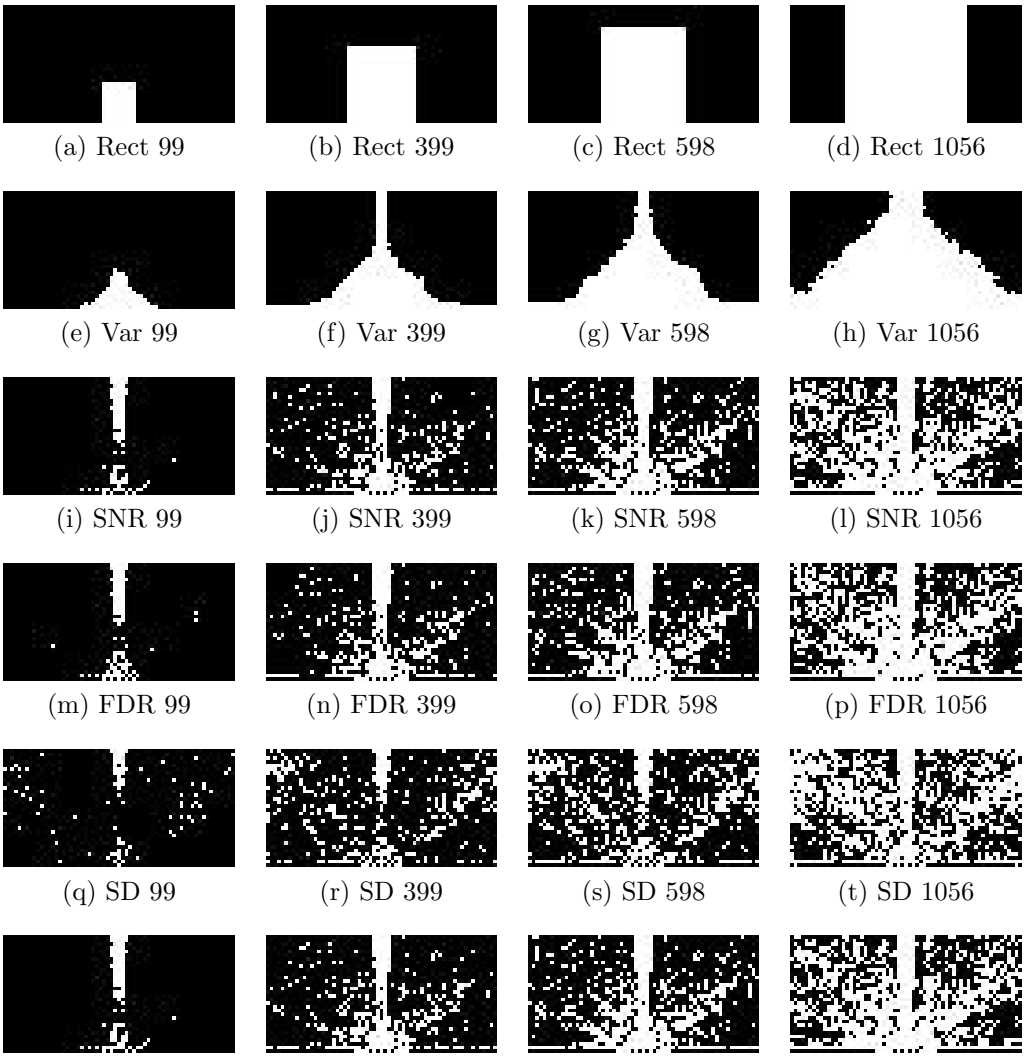
(q) SD 99 0.585 (r) SD 399 0.686 (s) SD 598 0.711 (t) SD 1056 0.737



(u) T 99 0.685 (v) T 399 0.741 (w) T 598 0.749 (x) T 1056 0.747

Table 11: Reconstruction of from the filtered result of a FFT transform on the first image in the Yale database. The caption specifies filter, d, and accuracy with phase.

5.10 Yale FFT Filters



(a) Rect 99 (b) Rect 399 (c) Rect 598 (d) Rect 1056

(e) Var 99 (f) Var 399 (g) Var 598 (h) Var 1056

(i) SNR 99 (j) SNR 399 (k) SNR 598 (l) SNR 1056

(m) FDR 99 (n) FDR 399 (o) FDR 598 (p) FDR 1056

(q) SD 99 (r) SD 399 (s) SD 598 (t) SD 1056

(u) T 99 (v) T 399 (w) T 598 (x) T 1056

Table 12: Filters trained on Yale database. White indicated a 1. Caption specifies filter type and d.

5.11 Yale WT Reconstructions

(a) Var 99 0.498 (b) Var 399 0.737 (c) Var 598 0.782 (d) Var 1056 0.800

(e) SNR 99 0.754 (f) SNR 399 0.813 (g) SNR 598 0.812 (h) SNR 1056 0.817

(i) FDR 99 0.758 (j) FDR 399 0.807 (k) FDR 598 0.812 (l) FDR 1056 0.820

(m) SD 99 0.42 (n) SD 399 0.672 (o) SD 598 0.725 (p) SD 1056 0.783

(q) T 99 0.760 (r) T 399 0.812 (s) T 598 0.812 (t) T 1056 0.816

Table 13: Reconstruction of from the filtered result of a wavelette transform on the first image in the Yale database. The caption specifies filter, d, and accuracy.

6 Conclusions

There is a lot of potential for privacy protection with proper alterations in the feature engineering process. Across all experiments the removal of the phase from the FFT result yields almost the same accuracy. However, the image becomes impossible reconstruct using simple means. Additionally, the normalization mean and variance can be hidden by the server, which also prevents reconstruction. Along the same lines, the feature vectors can be permuted by some hidden permutation matrix.

Regarding the filters, which should be applied even for performance improvements, SNR and T are the most useful for privacy protection. For both FFT and WT, these filters allow a large dimension reduction with almost no accuracy loss. This allows for reconstructions to be unidentifiable, see Table 11 cells j and v and Table 13 cells f and r. In general the supervised filter promote privacy in the reconstructed image. This is because they use high and low frequencies for reconstruction in FFT as seen in Table 12. The mix avoid details form being reconstructed fully. In WT, the filters prevent reconstruction of the eyes which is useful, although not fully understood. For high level of privacy, rectangle, triangle and variance filter performs best across experiments at the smallest tried d.

In future research consecutive search methods can be utilized to produce filters. Additionally, the reconstruction for WT should be explained in terms of the underlying bands. Lastly, a fully algorithm utilizing the filters and taking into the account the client server model should be developed.

This paper represents my own work in accordance with University regulations.

References

[1] A. Bouzalmat, J. Kharroubi, and A. Zarghili. Comparative study of pca, ica, lda using svm classifier. *Journal of Emerging Technologies in Web Intelligence*, 6(1):64–68, 2014.

[2] A. Bouzalmat, A. Zarghili, and J. Kharroubi. Facial face recognition method using fourier transform filters gabor and r_lda. *IJCA Special Issue on Intelligent Systems and Data Processing*, pages 18–24, 2011.

[3] K. W. Bowyer. Face recognition technology: security versus privacy. *Technology and Society Magazine, IEEE*, 23(1):9–19, 2004.

[4] Z. Dehai, D. Da, L. Jin, and L. Qing. A pca-based face recognition method by applying fast fourier transform in pre-processing. In *3rd International Conference on Multimedia Technology (ICMT-13)*. Atlantis Press, 2013.

[5] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*, pages 235–253. Springer, 2009.

[6] T. A. Kevenaar, G. J. Schrijen, M. van der Veen, A. H. Akkermans, and F. Zuo. Face recognition with renewable and privacy preserving binary templates. In *Automatic Identification Advanced Technologies, 2005. Fourth IEEE Workshop on*, pages 21–26. IEEE, 2005.

[7] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *Information, Security and Cryptology–ICISC 2009*, pages 229–244. Springer, 2010.

[8] A. S. Samra, S. G. El Taweel Gad Allah, and R. M. Ibrahim. Face recognition using wavelet transform, fast fourier transform and discrete cosine transform. In *Circuits and Systems, 2003 IEEE 46th Midwest Symposium on*, volume 1, pages 272–275. IEEE, 2003.

[9] H. Spies and I. Ricketts. Face recognition in fourier space. In *Vision Interface*, volume 2000, pages 38–44, 2000.