Face Verification by Using Sparse Representation Algorithm in Compressive Sensing

Arie Yang, Andreas Noviko Hadinata, Frisian Salim, Endra Oey and Rinda Hedwig

***Abstract*— Face verification, a part of security systems, is widely used in many applications. This biometric application is more hygienic comparing with other biometric systems since there is no direct contact between face and camera. Moreover, it is a low cost setup. A sparse representation algorithm as a part of compressive sensing was used in this paper with the accuracy achieved up to 88% during non-optimized sensing matrix and with the average time process of 4.37 seconds. The accuracy achieved was 94% during optimized sensing matrix but the average time process was slower at 8.73 seconds. Encryption process also happened during the image compression which not only reduced the size of the image but also increased the data security.**

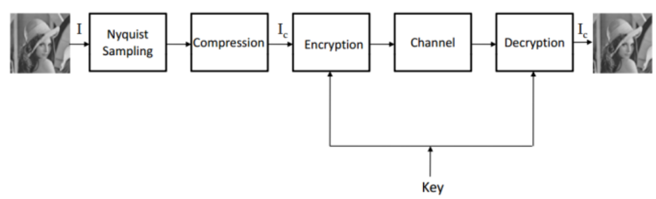
***Index Terms*— face verification, biometric, sparse representation algorithm, compressive sensing, encryption, security.**

I. Introduction

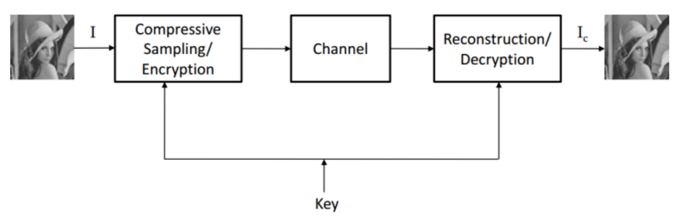
Security system by using biometric identification has been widely applied in many applications and a patent on its system was published in 2002. There are many choices of the system such as fingerprint identification, ear recognition, iris identification, palm recognition, palm vein identification and face verification. These biometric systems were chosen in order to increase the capability and efficiency of security system which is compared to the security system with card identification or personal identification number.

This paper mainly focuses on face verification since this is one of the biometric systems that are hygienic and low cost for the first installment. There are lots of algorithms to verify the face from individual to others with 1-to-1 matching. The face verification system developed in this paper uses sparse representation algorithm in compressive sensing. This technique has become one of the standard methods in face verification.

Data size and data security during its transmission from a local computer to the server are also in concern to discuss. Some biometrics images are too large for fast transmission so that they need first to be compressed in lossy or lossless media. In order to secure the data transmission, usually encryption is applied right after the compression process is carried out. These two steps processes sometimes need independent algorithm. Orsdemir et al. showed in his paper how image compression and encryption can be done together as one process. Moreover compressive sensing based encryption can overcome robustness against additive noise [1]. Figure 1 shows the difference of conventional compression and encryption steps, and the one that Orsdemir et al. did.



(a)



(b)

Figure 1: Block diagram of (a) conventional method of compression and encryption step, and (b) modified method of compression and encryption step.

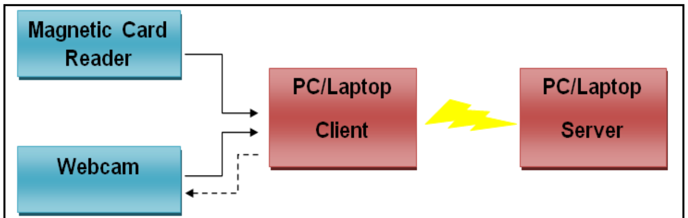
In figure 1, the modified method called compressive sampling or compressive sensing unifies sampling and compression to reduce data acquisition and computational load at sensor. Although computational load reduced at sensor, the computation increases at the receiver. It's because of compressive sampling lossier than conventional method and need to find the right 'key' so that can be reconstructed. Compressive sampling also allow a sensor to very efficiently capture the information in a sparse signal without trying to comprehend that signal.

In this paper, the same technique with some modifications applied previously by Orsdemir et al was used in order to improve the time process and its accuracy. Moreover, the security system was expected to be higher for wireless local area network (LAN) transmission.

A. Yang, A.N. Hadinata, F. Salim, E. Oey and R. Hedwig are with the Computer Engineering Department, Bina Nusantara University, Jakarta, Indonesia (phone: +6221-534-5830 ext. 2144; e-mail: arie.yank@gmail.com, andreasnoviko@yahoo.co.id, frisiansalim@yahoo.co.id, endraoey@binus.edu and rinda@binus.edu, respectively).

II. Hardware Design and Algorithm

Figure 2 shows the hardware design used in this paper. Two personal computers (PC) were used. One was functioned as a client and the other as database server where Windows 7® was installed as the operating systems. The specification for database server is Intel Core i5 Quad-Core 2.2 GHz with 4GB DDR3 amount of memory. Then for the client is Intel Core 2 Dual-Core processor 1.8 GHz with 2 GB DDR2 amount of memory. Both computers were connected via wireless media with ad-hoc configuration. In order to assure that there would be no interference from other hardware, the ad-hoc connection was protected with Wi-Fi Protected Access version 2/Pre-Shared Key (WPA2/PSK) encryption.



Notes: solid arrow indicates data line and the connectivity through universal serial bus (USB)
dash line indicates control line and the connectivity through USB
yellow lightning indicates connectivity via wi-fi adaptor

Figure2: Hardware design

Database server was used to store all data including biometric data, name of the user, account number, and magnetic card number. Biometric data was stored in dictionary form and sparse representation instead of original face image. Verification of face image was processed in database server and the verification result was sent directly to the client computer. On the other hand, client computer communicated with both magnetic card reader and webcam. Once the user swiped his/her card, the client would start to take his/her face image from webcam.

Checking the magnetic card number was done in order to simplify the data which was $\boldsymbol{\Psi}_{\boldsymbol{learned}}$ and 12 sparse coefficients of the user(s). After picture was taken, the image was encrypted and compressed in the same time through sparse representation algorithm which then was sent to the server later on. The client would wait for sometimes before getting the validation of verification process from the server.

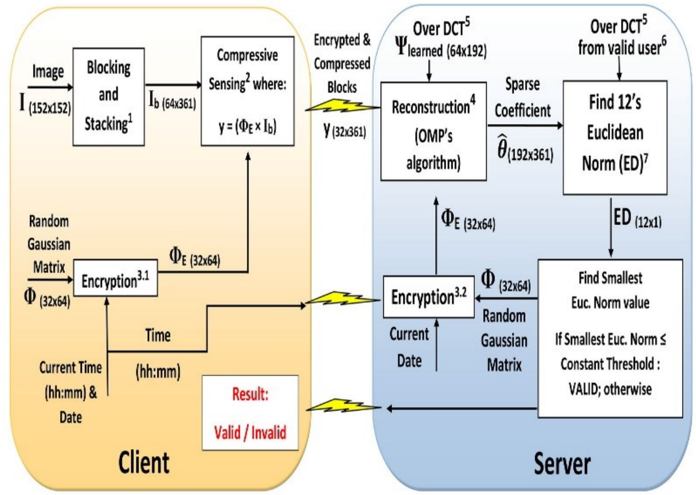


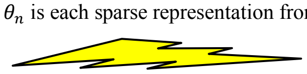
Figure 3: Block diagram of face verification through non-optimized sensing matrix. The communication which is shown here is between client and server.

Note for figure 3:

1. Blocking and stakcing: every 8 columns and 8 rows are grouped into 1 block whereas 1 block contains 8x8 matrix, for example (not related with this diagram):

$$I = \begin{bmatrix} \overset{\text{1st block}}{\boxed{\begin{matrix} I_{11} & I_{12} \\ I_{21} & I_{22} \end{matrix}}} & \overset{\text{2nd block}}{\boxed{\begin{matrix} I_{13} & I_{14} \\ I_{23} & I_{24} \end{matrix}}} & \cdots & \overset{\text{10th block}}{\boxed{\begin{matrix} I_{1(n-1)} & I_{1n} \\ I_{2(n-1)} & I_{2n} \end{matrix}}} \\ \vdots & & \ddots & \vdots \\ \underset{\text{91st block}}{\boxed{\begin{matrix} I_{(m-1)1} & I_{(m-1)2} \\ I_{m1} & I_{m2} \end{matrix}}} & & \cdots & \underset{\text{100th block}}{\boxed{\begin{matrix} I_{(m-1)(n-1)} & I_{(m-1)n} \\ I_{m(n-1)} & I_{mn} \end{matrix}}} \end{bmatrix}$$

   After be grouped into blocks, every block are stacked into 64x1 matrix, so it can produce 64 rows and 361 columns matrix.
2. Based on Candes, Emmanuel J., Wakin, Michael B.
3. For 3.1 and 3.2 are same. The encryption just to give a picture where the random Gaussian matrix need to be encrypted, by common encryption method like Advanced Encryption Standard (AES) or others method if the matrix need to be transmitted. In here, we avoid to transmit the matrix so the matrix reside in client and server side. We created basic encryption method based on time and date where they are calculated in basic mathematic operations. We just send the time, which used to encrypt random Gaussian matrix in client, to server. The result then to be added into random Gaussian matrix to blur the original value of the matrix. However in 3.2, we make the value of random Gaussian matrix as same as the 3.1 by the same method in 3.1 rather than decrypt them into original value. It's because the reconstruction need the exactly same value matrix as in the compressive sensing.
4. Algorithm by Eldar, Y.C., Kuppinger, P., Bolcskei, H.
5. Resulted from algorithm K-SVD by Aharon, Elad, Bruckstein (2006)
6. These are resulted from image training when user registered his/her face. The steps to find each sparse representation are: Find overDCT dictionary by using K-SVD and then find each sparse coefficient by using OMP where over DCT dictionary and each image training as input. The final result is 12's sparse representation and stored in database.
7. $\|\hat{\theta} - \theta_n\|_2$ where $\hat{\theta}$ is sparse representation from reconstruction and θ_n is each sparse representation from database.

 : Wireless transmission

During face verification, we tried two methods: non-optimized sensing matrix method as shown in figure 3, and optimized sensing matrix method as shown in figure 4. Here we try to figure out if compressive sensing can be used as application in security system such as face verification. We apply the compressive sensing method in face verification system where the biometric data must be transmitted via link. The main reason we used compressive sensing is it can compress biometric data into small size for bandwidth efficiency and can be reconstruct to resemble original data, then the reconstruct data can be used for authentication. It's not only compress the biometric data but also secure the data because the compressed data contains useless information if we don't reconstruct it with a key. The image used had the dimension of 152 X 152 pixels and it was changed to gray scale mode double data type. The image taken was processed in $\boldsymbol{I}$ matrix as shown in figure 3 followed by 8 X 8 pixels blocking. Blocking is a process to make a single picture into blocks with certain size of pixels which means that the first 8 X 8 blocks will be placed in first row to eighth row and first column to eighth column and so on. By doing so, we would get 19 blocks from first row to eighth row and another 19 blocks from first column to eighth column. In total, we would have 361 blocks for 152 X 152 pixels image. The whole 8 X8 blocks would then be stacked into 64 X 1 per block and each of it would be organized in 1 column of matrix so that for 152 X 152 pixels we could get matrix of 64 X 361which was called $\boldsymbol{I_b}$ matrix.

Compressive sensing was done afterwards before this $\boldsymbol{I_b}$ matrix was sent to the server to be verified. In order to run the compressive sensing, a $\boldsymbol{\Phi}$ sensing matrix was necessary to be defined at the beginning. This $\boldsymbol{\Phi}$ sensing matrix was determined by random matrix which was saved both in server and client. Before using it, a unique key such as date and time was added. This precaution was applied in order to make the intruder felt confused since the value of $\boldsymbol{\Phi}$ matrix would keep on changing from time to time.

The compressive sensing on $\boldsymbol{I_b}$ matrix was started right after sensing matrix was gained. $\boldsymbol{I_b}$ matrix was projected to $\boldsymbol{\Phi_U}$ which represented sensing matrix plus key. The compression rate allowed in this paper was 50% which meant that we reduced number of rows in $\boldsymbol{I_b}$ matrix into 32 X 361. By projecting $\boldsymbol{\Phi_U}$ to $\boldsymbol{I_b}$ matrix, it changed the value in $\boldsymbol{I_b}$ matrix into random value with lesser dimension and the matrix was named $\boldsymbol{y}$ matrix. This $\boldsymbol{y}$ matrix was then sent from client to server.

Server would reconstruct $\boldsymbol{y}$ matrix with $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi_{learned}}$ right after it received $\boldsymbol{y}$ matrix from client and in the same time the key was added to $\boldsymbol{\Phi}$ beforehand. Basically the reconstruction process was mean to find coefficient of sparse without regenerating $\boldsymbol{I_b}$ matrix from $\boldsymbol{y}$ matrix. This was due to the necessity of finding difference value between sparse coefficient reconstruction and sparse coefficient in database. The algorithm used to find sparse coefficient reconstruction was orthogonal matching pursuit (OMP)[2-3].

Sparse coefficient reconstruction was done block by block or column per column from $\boldsymbol{y}$ matrix where each given coefficient was kept in $\boldsymbol{\theta_R}$ matrix. The sparse coefficient found from reconstruction was compared with other 12 sparse coefficient from database and was determined the difference value of it. This process was done by applying Euclidean Norm [4-6] used as basic quantity when measuring vector difference.

The different value between 12 coefficient sparse and reconstruction coefficient sparse was stored in Euclidean Distance [7-8] matrix in which it contains 12 Euclidean norm in 12 rows (12x1). The smallest value determined from this 12 Euclidean Norm value which then was compared with threshold value. If the smallest value was 9.12, smaller than 10 the threshold value which, the server would tell the client to tell the user that 'access granted' and vice versa. If the user finished accessing his/her personal data or if access was denied, client would show the window for user to enter their identification number and delete all temporary data from previous transaction.

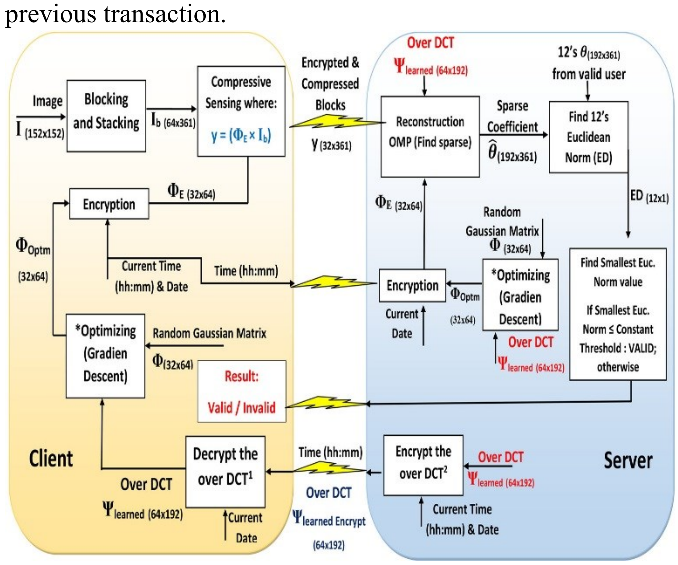


Fig. 4. Block diagram of face verification through optimized sensing matrix. The communication which is shown here is between client and server.

Note figure 4:

1 and 2: This encryption concept is same with encryption in random Gaussian matrix. But in here, the over DCT must be decrypted into original data before use for processing data.

Abolghasemi et al. introduced algorithm of optimized sensing matrix [9] used in this paper as seen in figure 4. The step of optimized sensing matrix was almost similar to non-optimized sensing matrix. However the $\boldsymbol{\Phi}$ sensing matrix should be optimized in the beginning before adding the key as shown in figure 4. Client would ask the server to send $\boldsymbol{\Psi_{learned}}$ of the user in order to optimize the $\boldsymbol{\Phi}$ sensing matrix and the server would send the encrypted $\boldsymbol{\Psi_{learned}}$ to client for decrypted. The encryption done in the server was a combination of day, date, and time as the keys to send to client directly. This encryption-description process defined that $\boldsymbol{\Psi_{learned}}$ should be secured first by conventional method. Dictionary would be used during optimizing $\boldsymbol{\Phi}$ sensing matrix when client got $\boldsymbol{\Psi_{learned}}$ so that an optimized $\boldsymbol{\Phi_{Optm}}$ sensing matrix could be achieved. If we used optimized sensing matrix to reconstruct $\boldsymbol{y}$ matrix, the server should use optimized $\boldsymbol{\Phi}$ sensing matrix as well and the key should be similar to the one in the client. By using this method, the reconstruction process would result the right reconstruction sparse coefficient of the image.

In this paper, validity of user was determined if the threshold value was higher than Euclidean Norm. The threshold value was determined by selecting the lowest 10 Euclidean Norm value from valid user and its outlier. Each person has 2 outliers so that there were 30 lowest Euclidean Norm value. We would find the highest value from 10 lowest Euclidean Norm value of valid user ($\boldsymbol{U_{max}}$), and also the lowest value from 10 lowest Euclidean Norm value of each outliers ($\boldsymbol{O_{min}}$). From the 1 value of $\boldsymbol{U_{max}}$ and 2 values of $\boldsymbol{O_{min}}$ we would determine the threshold value which was higher than $\boldsymbol{U_{max}}$ but lower than $\boldsymbol{O_{min}}$ (threshold > $\boldsymbol{U_{max}}$ & threshold < $\boldsymbol{O_{min}}$.

During the testing, we used 12 training image with 2 threshold value both in non-optimized sensing matrix and optimized sensing matrix. For information, there are restriction with some parameters when taking a photo either for training image or testing image. The parameters are:

-. Camera angle is between ± 15° of horizontal line in center of lens camera

-. Distance from face to camera is about 60 ± 5 cm.

-. Background is plain black-colored fabric.

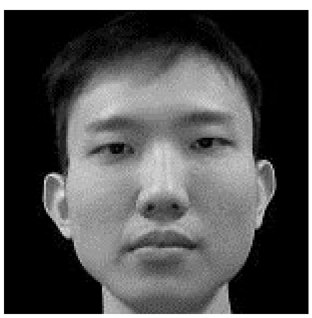


Fig. 5. Sample of training image with restriction of some parameters

Each threshold was tested as many as 540 data which consisted of 180 user data and 360 outliers' data. Each valid user took 60 data that consisted of 20 normal face expressions, 20 smiley face expression, and 20 grin face expression. Each outlier would take similar data like user so that for 2 outliers we could get 120 data. If A was valid user, B and C would be the outlier of A and vice versa. In total we got 2160 data.

Accuracy would be measured based on how the system performs during identifying between valid users and outliers. False Rejection Rate (FRR) and False Acceptance Rate (FAR) would also be measured. We also measured the time process from reading the magnetic card until the reply of the system to the user such as 'access denied' or 'access granted'.

III. Results and Discussion

Table 1. Accuracy, FRR and FAR data of system testing based on sensing matrix and threshold value.

Sensing Matrix	Threshold Value	Expression	Result (%)		
			Accuracy	FRR	FAR
Non Optimized	13	Normal	86.67	20.00	10.00
		Smiling	87.22	11.67	13.33
		Grinning	86.67	15.00	12.50
	14	Normal	79.44	3.33	29.17
		Smiling	82.78	3.33	24.17
		Grinning	82.22	8.33	22.5
Optimized	11	Normal	93.89	18.33	0.00
		Smiling	93.33	20	0
		Grinning	92.78	18.33	1.67
	11.5	Normal	92.78	18.33	1.67
		Smiling	93.33	18.33	0.83
		Grinning	93.89	18.33	0

Table 2. Time process data of system testing based on sensing matrix and threshold value.

Sensing Matrix	Time Process (seconds)		
	in total	validating identity number	verification
Non Optimized	19.88	3.17	4.37
Optimized	27.17	3.41	8.73

Table 1 show the accuracy is better in optimized sensing matrix compared to non-optimized sensing matrix. Nevertheless, the time process, as shown in table 2, was needed to do the optimized sensing matrix somewhat higher. The face expression itself would give no influence in verifying the face image. In non-optimized sensing matrix we found out that the modification of threshold value influenced both FRR and FAR. These changes gave less impact in optimized sensing matrix. From the above tables we can conclude that the best method which can be applied in face verification is by using optimized sensing matrix with threshold value of 11. This is due to the lower FAR percentage so that the higher security level can be obtained. On the other hand, the higher FRR can give influence on validity during user verification which can be reduced by re-verification.

Table 3. Accuracy, FRR and FAR data of optimized sensing matrix testing based on ratio of measurement number (RMN)

RMN	Thres-hold Value	Expres-sion	Result (%)		
			Accuracy	FRR	FAR
50%	11	Normal	93.89	18.33	0.00
		Smiling	93.33	20	0
		Grinning	92.78	18.33	1.67
	11.5	Normal	92.78	18.33	1.67
		Smiling	93.33	18.33	0.83
		Grinning	93.89	18.33	0
25%	11	Normal	82.22	53.33	0.00
		Smiling	82.22	53.33	0
		Grinning	80	60	0
	11.5	Normal	84.44	45	0.83
		Smiling	91.67	23.33	0.83
		Grinning	87.22	35	1.67

After gaining the result that the use of optimized sensing matrix was better than non-optimized sensing matrix, further investigation on ratio of measurement number (RMN) was taken by decreasing RMN from 50% to 25%. The calculation of RMN is based on equation of $RMN = M/N$, where M is number of rows and N is number of sensing matrix columns. Table 3 shows that the accuracy decreased about 10% but the FRR increased drastically. This is due to low RMN which reduces its accuracy and hence makes the validity low. Lower RMN also makes the whole system more sensitive in re-positioning in face recognition and increases the FRR value as well. The accuracy threshold is 11.5 when the RMN is 25% which is better than 11. However the FAR value is also higher in that threshold. If FAR value in 11.5 thresholds is compared between RMN of 25% and 50%, the FAR value for RMN of 25% will be higher which causes reducing the sensitivity of the system. Besides, the face recognition somewhat has lower accuracy which leads to misrecognizing people.

Table 4 shows that the time elapsed needed to validate the face recognition process between RMN of 25% and 50% is almost similar although the verification process is higher in RMN of 25%. For this experiment the number of data taken increases from 2160 data to 3240 data.

Table 4: Time process data of optimized sensing matrix testing based on ratio of measurement number (RMN)

RMN	Time Process (seconds)		
	in total	validating identity number	verification
50%	27.17	3.41	8.73
25%	25.48	3.32	9.48

IV. CONCLUSION

Face verification by using optimized sensing matrix is able to achieve an accuracy above 90%, which better than the non-optimized sensing matrix with accuracy below 90%, although the processing time on the optimized sensing matrix is higher. By decreasing RMN value of optimized sensing matrix from 50% to 25%, the accuracy decreased about 10%. For further improvement of this research, we recommend to use better dictionary learning such as Enhanced K-Singular Value Decomposition (EK-SVD) and better sparse coding such as Compressive Sampling Matching Pursuit (CoSaMP).

REFERENCES

[1] A. Orsdemir, H.O. Altun, G. Sharma, and M.F. Bocko, "On The Security and Robustness of Encryption Via Compressed Sensing", *IEEE Military Communications Conference*, 16-19 Nov 2008, p. 1-7.

[2] J.A. Tropp and A.C. Gilbert, "Signal Recovery from Random Measurement via Orthogonal Matching Pursuit", *IEEE Transactions on Information Theory*, 53, 12 (2007), p.4655-4666.

[3] R. Rebollo-Neira and D. Lowe, "Optimized Orthogonal Matching Pursuit Approach", *IEEE Signal Processing Letters*, 9, 4 (2002), p. 137-140.

[4] A.S. Householder, "Unitary Triangularization of a Nonsymmetric Matrix", *Journal of the ACM*, 5, 5 (1958), p. 339-342.

[5] F.L. Bauer and C.T. Fike, "Norms and Exclusion Theorems", *Numerische Mathematik*, 2, 1 (1960), p. 137-141.

[6] M. Barni, V. Cappellini, and A. Mecocci, "Fast Vector Median Filter Based on Euclidean Norm Approximation", *IEEE Signal Processing Letters*, 1, 6 (1994), p. 92-94.

[7] J.C. Gower, "Properties of Euclidean and non-Euclidean Distance Matrices", *Linear Algebra and Its Applications*, 67 (1985), p. 81-97.

[8] P.E. Danielsson, "Euclidean Distance Mapping", *Computer Graphics and Image Processing*, 14, 3 (1980), p. 227-248.

[9] V. Abolghasemi, S. Ferdowsi, B. Makiaabadi, and S. Sanie, "On Optimization of the Measurement Matrix for Compressive Sensing", *18th European Signal Processing Conference*, 23-27 August 2010, p. 427-431.