Chapter 5

SIMPCA and FLPCA: Feature Partitioning Approaches to PCA for Image Data

5.1 Introduction

Note: The work in this chapter has been published in *Pattern Recognition Letters Journal (Elsevier Science)*\(^1\) and in *proceedings of IAPR Conference on Machine Vision and Applications (IAPR-MVA 2007)*\(^2\).

In the last chapter, the SubXPCA method was proposed which is a novel Fea-


ture Partitioning based PCA (FP-PCA) method. SubXPCA method addressed some of the important feature partitioning issues-(i) Loss of inter-sub-pattern covariances or correlations and (ii) feature order dependency. During the study and motivation of the SubXPCA it was found that, it is not always correct to rely only on locally extracted features from feature blocks as done by methods like modPCA [53] and SubPCA [21] (Section 2.2 of Chapter 2). A more sophisticated combination approach with global features using inter-block feature correlations is required. It was shown that SubXPCA is relatively more robust against feature orders and less sensitive to overlapping sub-patterns. However, the existing FP-PCA methods including SubXPCA (Section 2.2 of Chapter 2) use classical PCA as their preferred method for local feature extraction. When feature extraction methods like classical PCA are applied on images, we see that classical PCA treats an image pattern of size $m \times n$ as a vector of $m.n$ feature values. In other words, classical PCA does not make use of inherent matrix structure of an image, thus recognition performance and computational performance ($O(N.m^2.n^2)$) of classical PCA may not be encouraging. The same problem lies with existing FP-PCA methods (such as modPCA, EigenRegions method, SubPCA, etc) although better than classical PCA, they do not use matrix structure of images. Please note that matrix structure of image is more appropriate and captures crucial spatial relationships in the image and may improve recognition or classification. Improving the performance on image recognition problems was confirmed through the explicit use of the matrix structure of image data as proposed in the IMPCA (also known as 2DPCA) approach [187][189]. However, IMPCA (2DPCA) approach is not based on a feature-partitioning framework, hence it does not exploit
the strengths of feature-partitioning framework.

Thus, in this chapter, our goal is to apply the feature partitioning framework to PCA computation upon matrix structure of image data. This gives rise to the first of our novel approaches, Sub-IMage based Principal Component Analysis (SIMPCA). A more sophisticated way to combine local features extracted from sub-image blocks (using global inter-block correlations) allows further improvement over SIMPCA and we propose the second novel method, FLexible Image Principal Component Analysis (FLPCA). We prove the computational superiority of our methods by algorithm analysis and confirm practically through comprehensive experimentation on face data sets and palmprint data. The experimentation methodology is the very systematic approach as proposed [135]. In our experimentation we find False Rejection Ratio (FRR), False Acceptance Ratio (FAR) in addition to Total Error Rate (TER). Another important experimentation parameter is the number of sub-images (blocks) used for partitioning, and we analyze the recognition rate stability of the approaches with respect to number of sub-image blocks. This is a unique aspect of our experimentation showing the variation in performance due to partitioning. We categorize the performance of the various approaches studied in this Chapter by plotting their performance in terms of the parameters of computation time and recognition. Our study shows that FLPCA performance is the best according to both of the performance parameters.

The rest of the chapter is organized as follows. In section 5.2 we present formally our proposed approaches upon image data. A detailed study of the time complexity of these approaches is performed in section 5.3. We present the details of experimental
application of these approaches on the standard face databases and palmprint data in section 5.4.

5.2 Feature Partitioning based PCA (FP-PCA) Approaches for Image Data

In this section, we propose the methods for image data, SIMPCA and FLPCA, which take advantage of the appropriate matrix arrangement of images and also improve performance by using feature partitioning framework. SIMPCA extracts features locally from sub-image blocks using 2DPCA (IMPCA) method [189], but performs a simple combination of these local features. FLPCA proposed here improves upon SIMPCA by a more informed combination approach to remove redundant local features using global correlations.

5.2.1 Sub-Image Principal Component Analysis (SIMPCA)

In this subsection, we formally present the first FP-PCA approach, SIMPCA, exclusively for image data. For a better understanding of the method Fig. 5.1 should be studied. Consider \( \mathbf{A} = \{ \mathbf{A}_1, \mathbf{A}_2, \ldots , \mathbf{A}_N \} \), the set of \( N \) mean-subtracted images of size \( m \times n \), where \( m, n \) are the number of rows and columns respectively. Each image, \( \mathbf{A}_i \), is treated as a \( m \times n \) matrix of image features (pixels).

**Step 1: Partitioning of images (Step-1 in Fig. 5.1)**

Divide the \( i^{\text{th}} \) image, \( (\mathbf{A}_i)_{m \times n} \), into \( k \), \( (2 \leq k \leq \frac{n}{2}) \), sub-images, \( \{ \mathbf{A}_j^i \; j = 1, 2, \ldots , k \} \), each of size \( m \times u \), where \( u = \left\lfloor \frac{n}{k} \right\rfloor \) and it is clear that \( 2 \leq u \leq \frac{n}{2} \). If \( k \) is not an exact
divisor of \( n \), (i.e. \( n \neq k.u \)), (i) one option is to truncate last \( (n - k.u) \) columns (we used this option for our experimentation), (ii) other option may be to extract features from the last sub-image similar to other sub-images, but with a different sub-image size and so on. We divide each image by taking features of \( u \) contiguous columns.

The \( j^{th} \) sub-image of an image \( A_i \) is given by

\[
(A^j_i)_{m \times u} = \begin{bmatrix}
  a_{1,(l+1)} & a_{1,(l+2)} & \cdots & a_{1,(l+u)} \\
  a_{2,(l+1)} & a_{2,(l+2)} & \cdots & a_{2,(l+u)} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m,(l+1)} & a_{m,(l+2)} & \cdots & a_{m,(l+u)}
\end{bmatrix}
\]

(5.1)

where \( l = (j - 1).u \) and \( a_{s_1,s_2} \) represents an image feature at matrix location \((s_1, s_2)\).

Here we used non-overlapping option of sub-images, that is no two sub-images have common features. Although we have shown here to divide an image with respect to columns, \( n \), in principle, we can also divide with respect to rows, \( m \), or with respect to both columns and rows.

**Step 2: Grouping of sub-images (Step-2 in Fig. 5.1)**

Form \( P^j \) as the set of \( j^{th} \) sub-images of images, \( \{A_i, i = 1, 2, \ldots, N\} \), given by

\[
P^j = \{A^j_1, A^j_2, \ldots, A^j_N\}
\]

(5.2)

Here we use the option of grouping of homogeneous sub-images (sub-patterns) (Defn. 2 of Chapter 3).

**Step 3: Local feature extraction from sub-images (Step-3 in Fig. 5.1)**

For each sub-image set, \( P^j, j \in \{1, 2, \ldots, k\} \) perform the following steps (a)-(c).

(a) Compute the local covariance matrix, \((M^j)_{u \times u}\) for the sub-image set as given by

\[
(M^j)_{u \times u} = \frac{1}{N} \sum_{i=1}^{N} [A^j_i]_{u \times m}^T [A^j_i]_{m \times u}
\]

(5.3)
(b) Find \( r (\leq u) \), eigenvectors of \( M^j \) corresponding to first \( r \) largest eigenvalues using eigenvalue decomposition (EVD) as follows.

\[
M^j e_p^j = e_p^j \lambda_p^j \tag{5.4}
\]

where \( e_p^j \) is the \( p^{th} \) eigenvector with eigenvalue \( \lambda_p^j \). Let \((E^j)_{u \times r}\) be the matrix of \( r \) column eigenvectors chosen in this step as given by

\[
(E^j)_{u \times r} = [e_1^j \ e_2^j \ldots e_r^j] = \\
\begin{bmatrix}
  e_1^j(1) & e_2^j(1) & \ldots & e_r^j(1) \\
  e_1^j(2) & e_2^j(2) & \ldots & e_r^j(2) \\
  \vdots & \vdots & \ddots & \vdots \\
  e_1^j(u) & e_2^j(u) & \ldots & e_r^j(u)
\end{bmatrix} \tag{5.5}
\]

(c) Project \( P^j = \{A_1^j, A_2^j, \ldots, A_N^j\} \) onto \( E^j \) to get a set of locally-reduced sub-images, \( \{B_i^j ; i = 1, 2, \ldots, N\} \) and is given by

\[
(B_i^j)_{m \times r} = (A_i^j)_{m \times u} \cdot (E^j)_{u \times r} \tag{5.6}
\]

\[
(B_i^j)_{m \times r} = \\
\begin{bmatrix}
  b_{1,(t+1)}^j & b_{1,(t+2)}^j & \ldots & b_{1,(t+r)}^j \\
  b_{2,(t+1)}^j & b_{2,(t+2)}^j & \ldots & b_{2,(t+r)}^j \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{m,(t+1)}^j & b_{m,(t+2)}^j & \ldots & b_{m,(t+r)}^j
\end{bmatrix} \tag{5.7}
\]

where \( t = (j - 1).r \) and \( b_{i_1,t_2}^j \) is a local feature at matrix location \((t_1, t_2)\).

**Step 4: Combining local features of sub-images (Step-4 in Fig. 5.1)**

Collate all reduced sub-images corresponding to the image, \( A_i \), to give locally-reduced image matrix, \( B_i \), and is given by

\[
(B_i)_{m \times k,r} = [B_i^1 \ B_i^2 \ldots B_i^k] \tag{5.8}
\]
Each \((B_i)_{m \times k, r}\) obtained in Step-4 is viewed as a \((m \times k \times r)\)-dimensional vector, and is used for subsequent pattern recognition tasks.

In SIMPCA, each image is partitioned into sub-images, then local features are extracted from each sub-image and these local features are used for subsequent tasks. We expect that SIMPCA works better when local structure plays a dominant role and may not perform well when features vary globally (i.e. when global structure is present). To perform well in either case, we propose a flexible method, FLPCA, under the feature partitioning framework in the next sub-section. The flexibility of FLPCA is due to its comprehensive local as well as global capacity for feature extraction.

### 5.2.2 FLexible Image Principal Component Analysis (FLPCA)

Here, we formally present the second FP-PCA approach, FLPCA. To appreciate and understand our method it is good to study Fig. 5.2.

**Step 1, Step 2 and Step 3:** Same as SIMPCA method (Steps 1-3 in Fig. 5.1).

**Step 4:** Combining local features of sub-images (Fig. 5.2):

(A) Collate locally-reduced sub-images: Same as Step-4 of SIMPCA. This step produces locally-reduced images, \(\{B_i; i = 1, 2, \ldots, N\}\) (Step-4(A) in Fig. 5.2).
(B) Global feature extraction from locally-reduced image matrices, $\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_N\}$ using inter-block correlations (Step-4(B) in Fig. 5.2):

(i) Compute image covariance matrix, $(\mathbf{M}^g)_{k.r \times k.r}$ from $\mathbf{B}$ as given by

$$\mathbf{M}^g = \frac{1}{N} \sum_{i=1}^{N} [\mathbf{B}_i]_{k.r \times m}^T [\mathbf{B}_i]_{m \times k.r}$$  \hspace{1cm} (5.10)

(ii) Compute eigenvectors of $\mathbf{M}^g$ using eigenvalue decomposition (EVD) as given by

$$\mathbf{M}^g.e_s = e_s \lambda_s$$  \hspace{1cm} (5.11)

where $e_s$ is the $s^{th}$ eigenvector with eigenvalue $\lambda_s$.

(iii) Select $w (\leq k.r)$, eigenvectors of $\mathbf{M}^g$ corresponding to first $w$ largest eigenvalues. Let $(\mathbf{E}^g)_{k.r \times w}$ be the matrix of $w$ column eigenvectors chosen in this step and is given by

$$(\mathbf{E}^g)_{k.r \times w} = [e_1 \ e_2 \ldots e_w] = \begin{bmatrix}
  e_1(1) & e_2(1) & \ldots & e_w(1) \\
  e_1(2) & e_2(2) & \ldots & e_w(2) \\
  \vdots & \vdots & \ldots & \vdots \\
  e_1(k.r) & e_2(k.r) & \ldots & e_w(k.r)
\end{bmatrix}$$  \hspace{1cm} (5.12)

(iv) Finally project $\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_N\}$ onto $\mathbf{E}^g$ to get a set of reduced image matrices, $\mathbf{D} = \{\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_N\}$. $\mathbf{D}_i$ is the reduced image matrix corresponding to the original image, $(\mathbf{A}_i)_{m \times n}$ and is given by

$$(\mathbf{D}_i)_{m \times w} = (\mathbf{B}_i)_{m \times k.r} \cdot (\mathbf{E}^g)_{k.r \times w}$$  \hspace{1cm} (5.13)

Each $(\mathbf{D}_i)_{m \times w}$ thus obtained is viewed as a $(m.w)$-dimensional vector, and is used for subsequent pattern recognition tasks.

In the above step, features are extracted based on global variation among the extracted local features (i.e. using inter-block dependencies or correlations as discussed
in section 3.3.8 of Chapter 3), which may further aid in dimensionality reduction and may increase recognition rate as well.

**Theorem 5** IMPCA (2DPCA) is a special case of FLPCA.

**Proof 5** IMPCA finds correlations between every pair of $n$ column-features. Here in steps 3(b)-3(c) of FLPCA we need to set $r = u$ (i.e. all features of every sub-image are chosen and $u = \frac{n}{k}$). Therefore in Step-4, FLPCA finds correlations between $m \times (k.r) = m \times (k.u) = m \times (k.\left(\frac{n}{k}\right)) = m \times n$ features, that is FLPCA finds correlations between all column-features. Hence the theorem is proved.

In the next section, we discuss the time complexities of the various variations of PCA and consequently of the proposed FP-PCA approaches, SIMPCA and FLPCA.

### 5.3 Time Complexity Analysis

Here we focus mainly on time complexity of computation of covariance matrix since it is the most dominant factor, computationally, in PCA variations. Consider $A = \{A_1, A_2, \ldots, A_N\}$, the set of $N$ images of size $m \times n$. Each image, $A_i$, is treated as a $m \times n$ matrix of image features (pixels).

Time complexities of calculating covariance matrix(ces), $T_C$, $T_M$, $T_o$, etc, by various PCA methods are shown in Table 5.1.

**Theorem 6** $T_M = \frac{1}{k.m} T_C$, where $2 \leq k \leq \frac{n}{2}$ is the number of sub-images per image and $m, n$ are the number of rows and the number of columns of an image matrix respectively.
Chapter 5: SIMPCA and FLPCA: Feature Partitioning Approaches to PCA for Image Data

Figure 5.1: Visualizing SIMPCA method
Chapter 5: SIMPCA and FLPCA: Feature Partitioning Approaches to PCA for Image Data

Figure 5.2: Visualizing FLPCA method
Table 5.1: Time complexities of various PCA methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Time Complexity</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficient Classical PCA</td>
<td>$T_E = O(N^2.m.n)$</td>
<td>–</td>
</tr>
<tr>
<td>Classical PCA</td>
<td>$T_C = O(N.m^2.n^2)$</td>
<td>–</td>
</tr>
<tr>
<td>modPCA</td>
<td>$T_o = O(k.N.u_1^2,u_2^2)$</td>
<td>$u_1 \times u_2$: sub-image size $u_1 = \left\lfloor \frac{m}{k_1} \right\rfloor$; $u_2 = \left\lfloor \frac{n}{k_2} \right\rfloor$; $k = k_1.k_2$: no. of sub-images</td>
</tr>
<tr>
<td>IMPCA (2DPCA)</td>
<td>$T_I = O(N.m.n^2)$</td>
<td>–</td>
</tr>
<tr>
<td>SIMPCA</td>
<td>$T_M = O(k.N.m.u^2)$</td>
<td>–</td>
</tr>
<tr>
<td>FLPCA</td>
<td>$T_L = O(k.N.m.u^2 + N.m.(k.r)^2)$</td>
<td>$O(k.N.m.u^2)$: to compute $k$ sub-image cov. matrices, $O(N.m.(k.r)^2)$: to compute image cov. matrix, $M^9$</td>
</tr>
</tbody>
</table>

**Proof 6** From eq. $T_M$ of Table 5.1, $T_M = k.N.m.u^2$

$\Rightarrow T_M = \frac{1}{k}.N.m.n^2$ (because $u = \frac{n}{k}$)

$\Rightarrow T_M = \frac{1}{k} \cdot \frac{1}{m}.N.m^2.n^2$

$\Rightarrow T_M = \frac{1}{k.m}.T_C$ (from eq. $T_C$ of Table 5.1)

Hence the theorem follows.

**Theorem 7** $T_L < T_C$, where $2 \leq k \leq \frac{n}{2}$ is the number of sub-images per image, $u$ is the number of columns in sub-image and $m$, $n$ are the number of rows and the number of columns of an image matrix respectively.

**Proof 7** From eq. $T_L$ of Table 5.1, $T_L = O(k.N.m.u^2 + N.m.(k.r)^2)$

$\Rightarrow T_L = T_M + N.m.\frac{n^2}{u^2}.r^2$ (because $k = \frac{n}{u}$ and from eq. $T_M$ of Table 5.1)

$\Rightarrow T_L = \frac{1}{k.m}.T_C + \frac{r^2}{u^2}.\frac{1}{m}.N.m^2.n^2$ (from Theorem 6, $T_M = \frac{1}{k.m}.T_C$)

$\Rightarrow T_L = \frac{1}{k.m}.T_C + \frac{r^2}{u^2}.\frac{1}{m}.T_C$ (from eq. $T_C$ of Table 5.1)

$\Rightarrow T_L = T_C.(\frac{1}{k.m} + \frac{r^2}{u^2}.\frac{1}{m})$
We know that \((\frac{1}{k.m} + \frac{r^2}{u^2} \frac{1}{m}) < 1\), because \(k \geq 2, m \geq 2\) and \(r \leq u\).

\[ \Rightarrow T_L < T_C \]

Hence the theorem follows.

**Theorem 8** \(T_M < T_E\), \(\forall u < N\), where \(u\) is the number of columns in a sub-image and \(N\) is the number of training image patterns.

**Proof 8** From eq. \(T_M\) of Table 5.1, \(T_M = k.N.m.u^2\)

\[ \Rightarrow T_M = \frac{1}{k}.N.m.n^2 \quad \text{(because } u = \frac{n}{k}) \]

\[ \Rightarrow T_M = N.\frac{n}{k}.m.n \]

\[ \Rightarrow T_M < N.N.m.n \quad \text{if } \frac{n}{k} < N \]

\[ \Rightarrow T_M < T_E, \forall u < N. \quad \text{(because } u = \frac{n}{k} \text{ and from eq. } T_E \text{ of Table 5.1)} \]

Hence the theorem follows.

**Theorem 9** \(T_L < T_E\), \(\forall r < \sqrt{(N-u)\frac{n}{k}}\) or \(\forall k < (N-u)\frac{n}{r^2}\) and \(u < N\), where \(u\) is the number of columns in a sub-image; \(N\) is the number of training image patterns; \(2 \leq k \leq \frac{n}{2}\) is the number of sub-images per image; \(m, n\) are the number of rows and the number of columns of an image matrix respectively; \(r\) is the number of projection (eigen)vectors per sub-image set.

**Proof 9** From eq. \(T_L\) of Table 5.1, \(T_L = O(k.N.m.u^2 + N.m.(k.r)^2)\)

\[ \Rightarrow T_L = O(\frac{1}{k}.N.m.n^2 + N.m.k.\frac{n}{u}.r^2) \quad \text{(because } k = \frac{n}{u}) \]

\[ \Rightarrow T_L = N.m.n.(\frac{n}{k} + \frac{k}{u}.r^2) \]

\[ \Rightarrow T_L < N.N.m.n \quad \text{if } (u + \frac{k}{u}.r^2) < N \quad \text{(because } u = \frac{n}{k}) \]

\[ \Rightarrow T_L < N.N.m.n \quad \text{if } r < \sqrt{(N-u)\frac{n}{k}} \]

\[ \Rightarrow T_L < T_E, \forall r < \sqrt{(N-u)\frac{n}{k}}. \quad \text{(from eq. } T_E \text{ of Table 5.1)} \]
\[ T_L < T_E, \forall k < (N - u) \frac{r^2}{u^2} \]

It is clear that \( u < N \), otherwise we get an invalid result.

Hence the theorem follows.

**Theorem 10** \( T_M = \frac{1}{k}.T_I \), where \( 2 \leq k \leq \frac{n}{2} \) is the number of sub-images per image; \( r \) is the number of projection (eigen)vectors per sub-image set; \( m \times u \) is the sub-image size.

**Proof 10** From eq. \( T_M \) of Table 5.1,
\begin{align*}
T_M &= O(k.N.m.u^2) \\
\Rightarrow T_M &= k.N.m.u^2 \\
\Rightarrow T_M &= \frac{1}{k}.N.m.n^2 \\
\Rightarrow T_M &= \frac{1}{k}.T_I \text{ (from eq. } T_I \text{ of Table 5.1)}
\end{align*}

Hence the theorem follows.

**Theorem 11** \( T_L < T_I, \forall r < u.\sqrt{\frac{k-1}{k}}, \) where \( 2 \leq k \leq \frac{n}{2} \), is the number of sub-images per image; \( r \) is the number of projection (eigen)vectors per sub-image set; \( m \times u \) is the sub-image size.

**Proof 11** From eq. \( T_L \) of Table 5.1, \( T_L = O(k.N.m.u^2 + N.m.(k.r)^2) \)
\begin{align*}
\Rightarrow T_L &= \frac{1}{k}.N.m.n^2 + \frac{r^2}{u^2}.N.m.k^2.r^2 \\
\Rightarrow T_L &= \frac{1}{k}.N.m.n^2 + \frac{k^2}{n^2}.[N.m.n^2].r^2 \\
\Rightarrow T_L &= \frac{1}{k}.N.m.n^2 + \frac{r^2}{u^2}.[N.m.n^2] \text{ (because } u = \frac{n}{k}) \\
\Rightarrow T_L &= (\frac{1}{k} + \frac{r^2}{u^2}).[N.m.n^2] \\
\Rightarrow T_L < N.m.n^2 \text{ if } (\frac{1}{k} + \frac{r^2}{u^2}) < 1 \\
\Rightarrow T_L < N.m.n^2 \text{ if } \frac{r^2}{u^2} < (1 - \frac{1}{k})
\end{align*}
\[ T_L < T_I, \forall r < u, \sqrt{\frac{(k-1)}{k}} \] (from eq. T_I of Table 5.1)

Hence the theorem follows.

**Theorem 12** \[ \lim_{r \to 1, k \to 2} [T_L \approx T_M], \text{ where } 2 \leq k \leq \frac{n}{2} \text{ is the number of sub-images; } 1 \leq r \leq u \text{ is the number of eigenvectors chosen from each sub-image set; } m \times u \text{ is the size of sub-image.} \]

**Proof 12** Consider the second term of eq. \( T_L \) of Table 5.1 that is \( O(N.m.(k.r)^2) \)

\[ O(N.m.(k.r)^2) \text{ is minimum if } (k.r)^2 \text{ is minimum.} \]

\[ \Rightarrow O(N.m.(k.r)^2) \text{ is minimum if (i) } k \text{ and (ii) } r \text{ are minimum.} \]

\[ \Rightarrow O(N.m.(k.r)^2) \text{ is minimum if (i) } k = 2 \text{ and (ii) } r = 1 \text{ (because } 2 \leq k \leq \frac{n}{2} \text{ and } 1 \leq r \leq u). \]

Thus \( O(N.m.(k.r)^2) \) becomes insignificant for smaller values of \( k \) and \( r \). It is to be noted that First term of \( T_L \) is equal to \( T_M \).

Hence the theorem follows from eqs. \( T_M \) and \( T_L \) of Table 5.1.

**Theorem 13** \[ \lim_{r \to 1, k \to 2} [T_L \approx \frac{1}{k} T_I], \text{ where } 2 \leq k \leq \frac{n}{2} \text{ is the number of sub-images; } 1 \leq r \leq u \text{ is the number of eigenvectors chosen from each sub-image set; } m \times u \text{ is the size of sub-image.} \]

**Proof 13** From Theorem 10, \( T_M = \frac{1}{k} T_I \).

From Theorem 12, \( T_L \approx T_M \) as \( k \to 2 \) and \( r \to 1 \).

Therefore, \( T_L \approx \frac{1}{k} T_I \) as \( k \to 2 \) and \( r \to 1 \).

Hence the theorem follows.
Theorem 14 \( T_M = \frac{1}{u_1}.T_o \). Here we assume that both SIMPCA and modular PCA divide each image into sub-images of size \( u_1 \times u_2 \) in the same way, where \( 2 \leq u_1 \leq \frac{m}{2} \), and \( 2 \leq u_2 \leq \frac{n}{2} \), the number of rows and columns of a sub-image respectively.

Proof 14 From eq. \( T_M \) of Table 5.1, \( T_M = O(k.N.m.u^2) \), for sub-images of size \( m \times u \).
\[
\Rightarrow T_M = O(k.N.u_1.u_2^2) \text{, for sub-images of size } u_1 \times u_2.
\]
\[
\Rightarrow T_M = (\frac{1}{u_1}).k.N.u_1^2.u_2^2
\]
\[
\Rightarrow T_M = \frac{1}{u_1}.T_o \text{ (from eq. } T_o \text{ of Table 5.1).}
\]
Hence the theorem follows.

Theorem 15 \( T_L < T_o \), if \( k_2 < (u_1 - 1) \). Here we assume that both FLPCA and modular PCA divide each image into sub-images of size \( u_1 \times u_2 \) in the same way, where \( 2 \leq u_1 \leq \frac{m}{2} \), and \( 2 \leq u_2 \leq \frac{n}{2} \), the number of rows and columns of a sub-image; \( 1 \leq r < u_2 \) is the number of eigenvectors chosen from each sub-image set; \( k = k_1.k_2 \), the number of sub-images of an image.

Proof 15 From eq. \( T_L \) of Table 5.1, \( T_L = O(k.N.m.u^2 + N.m.(k.r)^2) \) for sub-images of size, \( m \times u \).
\[
\Rightarrow T_L = O(k.N.u_1.u_2^2 + N.m.(k_2.r)^2) \text{ for sub-images of size, } u_1 \times u_2.
\]
\[
\Rightarrow T_L = O(k.N.u_1.u_2^2 + N.k_1.u_1.(k_2.r)^2) \text{ (because } u_1 = \frac{m}{k_1}, u_2 = \frac{n}{k_2} \text{ from Table 5.1)}
\]
\[
\Rightarrow T_L = \frac{1}{u_1}.T_o + N.(k_1.k_2).u_1.k_2.r^2 \text{ (from eq. } T_o \text{ of Table 5.1)}
\]
\[
\Rightarrow T_L = \frac{1}{u_1}.T_o + N.k.u_1.k_2.r^2 \text{ (because } k = k_1.k_2 \text{ from Table 5.1)}
\]
\[
\Rightarrow T_L = \frac{1}{u_1}.T_o + \left[ \frac{(u_1-1)}{u_1}.\frac{u_1}{(u_1-1)} \right].N.k.u_1.k_2.r^2
\]
\[
\Rightarrow T_L = \frac{1}{u_1}.T_o + \frac{(u_1-1)}{u_1}.N.k.u_1^2 \cdot \frac{k_2}{u_1-1}.r^2
\]
\[ T_L = \frac{1}{u_1} T_o + \frac{(u_1-1)}{u_1} N.k.\frac{u_2}{u_1} \cdot \frac{u_2^2}{u_2^2} \]

\[ T_L = \frac{1}{u_1} T_o + \frac{(u_1-1)}{u_1} T_o \cdot \frac{u_2^2}{u_1-1} \cdot \frac{u_2^2}{u_2^2} \] (from eq. \( T_o \) from Table 5.1)

\[ T_L < T_o \text{ if } \left[ \frac{u_2^2}{u_1-1} \cdot \frac{u_2^2}{u_2^2} \right] < 1 \]

It is given in the statement that \( r < u_2 \), which implies \( \frac{u_2^2}{u_2^2} < 1 \).

\[ T_L < T_o \text{ if } \frac{k_2}{u_1-1} < 1 \]

\[ T_L < T_o \text{ if } k_2 < (u_1 - 1) \]

Hence the theorem follows.

From the above theorems we have proved that FP-PCA approaches (SIMPCA and FLPCA) show superior time complexities over Classical PCA, Efficient classical PCA, modular PCA and IMPCA techniques and the same is demonstrated by our experimentation in the next section. From theorems 6, 7 and 12 it is to be noted that both SIMPCA and FLPCA can be ideally \( k.m \) times faster than classical PCA. It is also evident from theorems 10-13 that FP-PCA approaches, SIMPCA and FLPCA, can be ideally nearly \( k \) times faster than IMPCA. Similarly, from theorems 12, 14 and 15 it is clear that both SIMPCA and FLPCA can be ideally \( m \) times faster than modular PCA, where \( k \) is the number of sub-images per image and \( m \) is the number of rows in an image. It is to be noted that we considered only computation of covariance matrix(ies) [because of its high time complexity as compared to other tasks] to arrive at time complexity of PCA variations considered. In fact, the total time complexity also includes finding eigenvectors, eigenvalues, matrix multiplications, etc.. Therefore, in practice, theoretical and actual time complexities may not match. However, theoretical complexities discussed above give a rough comparison of various PCA methods.
5.4 Experimental Results and Analysis

In this section, we report our experimental results based upon a benchmarking approach [135]. We used test data sets of impostor and clients from different subjects, and the subjects used for testing are not used for training. We explain the experiments conducted using our implementation and compare the results of PCA, IMPCA (2DPCA), modPCA (i.e. an existing FP-PCA approach), SIMPCA and FLPCA. We considered 3 face data sets [119] [166] [184] and PolyU palmprint data [126] for our experiments and we summarize the results in the following subsections.

In our experimentation we observe the performance of these approaches across the number of sub-images (blocks). We want to see the possible improvements due to partitioning and compare the performance of the FP-PCA approaches.

5.4.1 Data Sets

(i) ORL face data set [119] contains face images of 40 persons (subjects), each subject contains 10 images and there are 400 images in total. Each image is of dimension, 112 × 92 pixels (PGM format). Some images were taken at different times, with variation in lighting, facial expressions (open/closed eyes, smiling/not smiling) and with/without glasses. We selected all 200 images of first 20 subjects for training (i.e. to find principal components), next 13 subjects are taken as client data (legal users). From each client subject, we take first 5 images as templates (total enrollment of 65) and rest of them (65 images) for client testing. The last 7 subjects (total of 70 images) are taken as impostors (illegal users).

(ii) Yale face data [184] contains 165 gray scale GIF images of 15 individuals. There
are 11 images per subject indicating different expression or illumination conditions. We convert all the images into PGM format, of dimension $243 \times 320$ pixels. We selected all 77 images of first 7 subjects for training (i.e. to find principal components), next 5 subjects are taken as client data (legal users). From each client subject, we take first 5 images as templates (total enrollment of 25) and rest of them (30 images) for client testing. The last 3 subjects (total of 33 images) are taken as impostors (illegal users).

(iii) UMIST face data set [166] contains images of 20 individuals and a total of 565 images. Each covering a range of poses from profile to frontal views. Subjects cover race, sex, appearance. Each image is of dimension, $112 \times 92$ pixels (PGM format). We selected all 255 images of first 10 subjects for training (i.e. to find principal components), next 8 subjects are taken as client data (legal users). From each client subject, we take first 16 images as templates (total enrollment of 128) and rest of them (100 images) for client testing. The last 2 subjects (total of 82 images) are taken as impostors (illegal users).

(iv) PolyU palmprint data. We chose 498 images from first 25 subjects of PolyU palmprint database [126]. The data set contains around twenty samples from each of these subjects collected in two sessions, separated by a collection time interval of two months. The palmprint images of BMP format were converted to PGM format, ($284 \times 384$ pixels) and were used in our experiments.
5.4.2 Experimental Setup

We have chosen training, clients (for enrollment and testing) and impostor data sets from different subjects without overlapping as described in [135]. First, we find eigenvectors and eigenvalues using training data, then client and impostor data sets are projected on selected eigenvectors to get the data in reduced form. For each reduced client testing and impostor testing data we follow the steps: (i) Find similarity of test data to every client template (i.e. enrolled client) by using Euclidean distance measure, (ii) Next, find the maximum among similarity values found in the previous step, (iii) Accept testing data, if its maximum similarity found in step (ii) is greater than some threshold, $\delta \in (0,1)$, otherwise reject it. False Rejection Ratio (FRR), False Acceptance Ratio (FAR), Total Error Rate (TER) and Recognition Rates are calculated using the formulae: $\text{FAR} = \frac{\text{Number of Impostor data accepted}}{\text{Number of testing data or attempts}}$; $\text{FRR} = \frac{\text{Number of client data rejected}}{\text{Number of testing data or attempts}}$; $\text{TER} = \text{FAR} + \text{FRR}$; Recognition Rate $= 100 - \text{TER}$.

We conducted experiments by varying the number of sub-images per image ($k$). The number of sub-images are varied for different data sets as given: (i) ORL face data: For SIMPCA/FLPCA: $k = 2, 3, 4, 5, 7, 10$; For modPCA: $k_1 \times k_2 : 2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5, 6 \times 6$, (ii) Yale face data: For SIMPCA/FLPCA: $k = 2, 4, 5, 8, 10, 16, 20$; For modPCA: $k_1 \times k_2 : 2 \times 2, 3 \times 3, 4 \times 4, 8 \times 8, 16 \times 16, 32 \times 32$, (iii) UMIST face data: For SIMPCA/FLPCA: $k = 2, 3, 4, 5, 7, 10$; For modPCA: $k_1 \times k_2 : 2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5, 6 \times 6$, (iv) PolyU Palmprint data: For SIMPCA/FLPCA: $k = 4, 8, 9, 16, 25, 32$; For modPCA: $k_1 \times k_2 : 2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5, 10 \times 10, 16 \times 16, 32 \times 32$. For each $k$ value, we find maximum recognition rate by varying number of projection vectors.
Maximum recognition rate and execution times thus obtained are plotted as shown in Figs. 5.3 to 5.10. Overall maximum recognition rates and corresponding FAR, FRR, TER and execution times are shown in Tables 5.2 to 5.5. A novel plot (Fig. 5.11) is created between recognition rate and execution time based on the values shown in Tables 5.2 to 5.4 and the recognition rate is normalized with respect to maximum value for each data set. Similarly, execution time is also normalized.

The C language implementation of FLPCA is also used to compute results of IMPCA ($k$ is set to 1 and Step-4 is omitted) and SIMPCA (Step-4(B) is omitted). Classical PCA is implemented in C language using the efficient procedure [103]. We have not used original implementation of classical PCA because it used to take enormous amount of time for completion. We used a Pentium 4 based system with a CPU clock speed of 2.4 GHz, 256MB RAM and Fedora Core 5 Linux running on it for face data sets. For PolyU palmprint data, we used Pentium D based system with a CPU clock speed of 3.4 GHz, 4GB RAM and openSUSE Linux running on it.

### 5.4.3 Discussion of Results

The experimental results shown in Tables 5.2 to 5.5 reveal that the proposed FP-PCA approaches (SIMPCA and FLPCA) perform better than the other variations of PCA (including an existing FP-PCA approach, modPCA) based on overall performance criteria of recognition and computational time. For Yale face data, the proposed FP-PCA approaches show much better recognition rates (90%) than PCA (67%). Global structure among different kinds of ORL faces helps PCA to show reasonably good recognition rates (84%), but local structure plays a dominant role which
makes FP-PCA approaches (SIMPCA and FLPCA) to improve their recognition rates enormously (90% and 91% respectively) over PCA. FLPCA shows the highest recognition rate as compared to all the methods. UMIST face data contains images ranging from side to frontal views of images, hence full face is not captured in many images. Hence we expect all the methods to show relatively lower recognition rates, in comparison to other two data sets. In this case the FP-PCA approaches (SIMPCA and FLPCA) show relatively good recognition rate (75%), IMPCA (2DPCA) shows 72%, modPCA shows 73.6% and PCA shows 64%. The proposed FP-PCA approaches showcase their superior time complexities over other methods for all the data sets (Figs. 5.4, 5.6, 5.8 and 5.10). We understand that the original implementation of classical PCA shows huge computational complexity for high dimensional data, hence we use the efficient implementation of classical PCA [103] for our experimentation. The efficient implementation of PCA shows higher or competitive computational complexity as compared to our proposed FP-PCA approaches.

It is also observed that IMPCA shows better recognition rates for YALE and ORL data sets (90%) in comparison to classical PCA. Similar results are obtained for palmprint data. Modular PCA, although competitive to SIMPCA and FLPCA in terms of maximum recognition rates, but is not consistent in performance across the different number of sub-images as compared with FLPCA and SIMPCA. That is an important result which implies that, to get the best performance of modular PCA, a large amount of experimentation with the $k$ value is required.

To see the effect of local structure on recognition rate and execution time, we took results with varied number of sub-images ($k$) and the results are plotted in Figs. 5.3
FLPCA and SIMPCA show marginal variation in recognition rate and total error rate (except for UMIST face data) with different number of sub-images on face and palmprint data sets and modPCA shows drastic variation in recognition rate and total error rate with varying number of sub-images. We observed that FLPCA is less sensitive to number of sub-images as compared to modPCA and SIMPCA (Figs. 5.3, 5.5, 5.7 and 5.9).

A novel plot between Execution time and Recognition rate is shown in Fig. 5.11 to group methods based on overall performance. Such a novel plot allows one to conclude as to the method that gives high recognition rate at less execution time. It is clear from the Fig. 5.11 that both the FP-PCA approaches (SIMPCA and FLPCA) form a cluster (top left corner of the figure) of superior overall performance, i.e. superior recognition rates at less execution time as compared to all the other methods.

We observed that FLPCA is flexible to capture the best recognition rates of IM-PCA (2DPCA) and SIMPCA. Experimental results confirm the exceptional superiority of both FP-PCA approaches (FLPCA and SIMPCA) as compared to PCA, modPCA and IMPCA in terms of overall performance of recognition and execution time. Interestingly, FLPCA is able to adapt to different scenarios by taking both local and global variations into account. FLPCA does so by exploiting inter-block correlations, to remove redundancy (noise), among local features. In contrast to FLPCA, other two FP-PCA methods (modPCA and SIMPCA) extract only local features and do not exploit inter-block correlations.
5.5 Summary

In this chapter, we have introduced two novel FP-PCA approaches, SIMPCA and FLPCA, which use matrix data arrangement and are more appropriate for image pattern recognition. The approaches use feature partitioning framework providing local feature extraction, superior recognition and lower computational time. Theoretical properties related to the time complexity of the proposed approaches are proved. Experimental results reveal that SIMPCA and FLPCA perform better than IMPCA (2DPCA), modPCA (i.e. an existing FP-PCA approach) and PCA in terms of overall recognition and execution times. Both SIMPCA and FLPCA show better recognition and time complexities by taking local structure into account. FLPCA extracts local features and also combines locally-extracted features globally, to adapt to different image data sets. Our analysis shows that IMPCA (2DPCA) is a special case of FLPCA. FLPCA method is relatively less sensitive to the partitioning effects as compared to the other FP-PCA approaches (SIMPCA and modPCA). The applicability of SIMPCA and FLPCA techniques is demonstrated upon 3 standard face image data sets and palmprint data. The approaches are general and they can be extended to any image data set as well. SIMPCA and FLPCA may be extensively used in face recognition, palmprint recognition, data mining applications to image data and real time image recognition applications.

It is well known that IMPCA (2DPCA) method reduces small sample size (SSS) problem by computing $n \times n$ compact covariance matrix instead of computing $m.n \times m.n$ matrix. Both SIMPCA and FLPCA methods reduce the SSS problem much better by computing more compact $u \times u$ covariance matrices ($u < n$) because they
Table 5.2: Comparison of maximum recognition rates of proposed FP-PCA approaches over other PCA methods for Yale face data

<table>
<thead>
<tr>
<th>Method</th>
<th>Reco. Rate</th>
<th>FAR</th>
<th>FRR</th>
<th>TER</th>
<th>Time (secs.)</th>
<th>k</th>
<th>u</th>
<th>PVs</th>
<th>Total PVs used</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>66.67</td>
<td>30.16</td>
<td>3.17</td>
<td>33.33</td>
<td>175</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>85</td>
</tr>
<tr>
<td>modPCA</td>
<td>90.48</td>
<td>0.0</td>
<td>9.52</td>
<td>9.52</td>
<td>24</td>
<td>1024</td>
<td>7×10</td>
<td>5</td>
<td>5120</td>
</tr>
<tr>
<td>IMPCA (2DPCA)</td>
<td>90.48</td>
<td>3.17</td>
<td>6.35</td>
<td>9.52</td>
<td>219</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>14</td>
</tr>
<tr>
<td>SIMPCA</td>
<td>90.48</td>
<td>3.17</td>
<td>6.35</td>
<td>9.52</td>
<td>45</td>
<td>4</td>
<td>80</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>FLPCA</td>
<td>90.48</td>
<td>3.17</td>
<td>6.35</td>
<td>9.52</td>
<td>23</td>
<td>8</td>
<td>40</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

k: Sub-images per image; For modPCA, u is sub-image size and for other methods, \( m \times u \).
PVs: Number of Projection Vectors (eigenvectors chosen for projection) per sub-image set

extract image features by dividing each image of size \( m \times n \) into sub-images of size \( m \times u \).

In the next chapter, we study theoretical properties of FP-PCA methods including SubXPCA, SIMPCA and FLPCA with respect to summarization of variance.
Figure 5.3: Comparison of recognition rate for ORL face data. FLPCA shows more consistent performance across the number of sub-images as compared to modPCA and SIMPCA. FLPCA shows highest recognition rate of all the methods. FLPCA and SIMPCA also outperform PCA.
Figure 5.4: Comparison of computational time for ORL face data. FLPCA and SIMPCA show better efficiency across the number of sub-images as compared to modPCA. FLPCA and SIMPCA also show less computational time as compared to IMPCA. PCA shows competitive complexity to SIMPCA and FLPCA because we used the efficient implementation [103] instead of the original implementation of PCA.
Figure 5.5: Comparison of recognition rate for Yale face data. FLPCA shows consistently good performance irrespective of number of sub-images. SIMPCA shows more consistency as compared to modPCA. FLPCA and SIMPCA also outperform PCA.
Figure 5.6: Comparison of computational time for Yale face data. FLPCA and SIMPCA show better efficiency across various number of sub-images as compared to modPCA. FLPCA and SIMPCA show better computational time as compared to IMPCA (2DPCA) and the efficient implementation of PCA[103].
Figure 5.7: *Comparison of recognition rate for UMIST face data.* FLPCA shows better consistency across various number of sub-images as compared to modPCA and SIMPCA. FLPCA and SIMPCA show highest recognition rate as compared to PCA, IMPCA (2DPCA) and modPCA methods.
Figure 5.8: *Comparison of computational time for UMIST face data.* FLPCA and SIMPCA show better efficiency across various number of sub-images as compared to modPCA. FLPCA and SIMPCA show better computational time as compared to IMPCA and the efficient implementation of PCA [103].
Chapter 5: SIMPCA and FLPCA: Feature Partitioning Approaches to PCA for Image Data

Figure 5.9: Comparison of recognition rate for PolyU palmprint data. FLPCA and SIMPCA show better consistency across various number of sub-images as compared to modPCA. FLPCA and SIMPCA also outperform PCA.
FLPCA and SIMPCA show better efficiency across various number of sub-images as compared to modPCA. FLPCA and SIMPCA also show better computational time as compared to IMPCA (2DPCA). PCA shows competitive time complexity to SIMPCA and FLPCA because we used the efficient implementation [103] instead of the original implementation of PCA.
Chapter 5: SIMPCA and FLPCA: Feature Partitioning Approaches to PCA for Image Data

Figure 5.11: *Execution time versus Recognition rate with respect to 3 face data sets (UMIST, ORL, Yale).* FLPCA and SIMPCA points occupy left top corner part of the chart, forming a cluster of superior recognition rate at less computational overhead as compared to other methods.
### Table 5.3: Comparison of maximum recognition rates of proposed FP-PCA approaches over other PCA methods for ORL face data

<table>
<thead>
<tr>
<th>Method</th>
<th>Reco. Rate</th>
<th>FAR</th>
<th>FRR</th>
<th>TER</th>
<th>Time (secs.)</th>
<th>k</th>
<th>u</th>
<th>PVs</th>
<th>Total PVs used</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>84.44</td>
<td>3.71</td>
<td>11.85</td>
<td>15.56</td>
<td>6</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>9</td>
</tr>
<tr>
<td>modPCA</td>
<td>90.37</td>
<td>0.00</td>
<td>9.63</td>
<td>9.63</td>
<td>67</td>
<td>16</td>
<td>–</td>
<td>–</td>
<td>112</td>
</tr>
<tr>
<td>IMPCA (2DPCA)</td>
<td>90.38</td>
<td>2.22</td>
<td>7.40</td>
<td>9.62</td>
<td>17</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>4</td>
</tr>
<tr>
<td>SIMPCA</td>
<td>90.37</td>
<td>0.00</td>
<td>9.63</td>
<td>9.63</td>
<td>9</td>
<td>2</td>
<td>46</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>FLPCA</td>
<td>91.11</td>
<td>1.48</td>
<td>7.41</td>
<td>8.89</td>
<td>9</td>
<td>2</td>
<td>46</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

k: Sub-images per image; For modPCA, $u$ is sub-image size and for other methods, $m \times u$.
PVs: Number of Projection Vectors (eigenvectors chosen for projection) per sub-image set.

### Table 5.4: Comparison of maximum recognition rates of proposed FP-PCA approaches over other PCA methods for UMIST face data

<table>
<thead>
<tr>
<th>Method</th>
<th>Reco. Rate</th>
<th>FAR</th>
<th>FRR</th>
<th>TER</th>
<th>Time (secs.)</th>
<th>k</th>
<th>u</th>
<th>PVs</th>
<th>Total PVs used</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>63.74</td>
<td>1.65</td>
<td>34.61</td>
<td>36.26</td>
<td>28</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>35</td>
</tr>
<tr>
<td>modPCA</td>
<td>73.63</td>
<td>0.55</td>
<td>25.82</td>
<td>26.37</td>
<td>111</td>
<td>16</td>
<td>–</td>
<td>–</td>
<td>16</td>
</tr>
<tr>
<td>IMPCA (2DPCA)</td>
<td>71.98</td>
<td>12.09</td>
<td>15.93</td>
<td>28.02</td>
<td>22</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>2</td>
</tr>
<tr>
<td>SIMPCA</td>
<td>75.27</td>
<td>0.00</td>
<td>24.73</td>
<td>24.73</td>
<td>11</td>
<td>2</td>
<td>46</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>FLPCA</td>
<td>75.27</td>
<td>0.00</td>
<td>24.73</td>
<td>24.73</td>
<td>10</td>
<td>2</td>
<td>46</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

k: Sub-images per image; For modPCA, $u$ is sub-image size and for other methods, $m \times u$.
PVs: Number of Projection Vectors (eigenvectors chosen for projection) per sub-image set.
<table>
<thead>
<tr>
<th>Method</th>
<th>Reco. Rate</th>
<th>FAR</th>
<th>FRR</th>
<th>TER</th>
<th>Time (secs.)</th>
<th>k</th>
<th>u</th>
<th>PVs</th>
<th>Total PVs used</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>63.43</td>
<td>36.11</td>
<td>0.46</td>
<td>36.57</td>
<td>29</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>15</td>
</tr>
<tr>
<td>modPCA</td>
<td>94.91</td>
<td>0.0</td>
<td>5.09</td>
<td>5.09</td>
<td>377</td>
<td>256</td>
<td>17×24</td>
<td>300</td>
<td>76800</td>
</tr>
<tr>
<td>IMPCA (2DPCA)</td>
<td>94.45</td>
<td>0.46</td>
<td>5.09</td>
<td>5.09</td>
<td>238</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>SIMPCA</td>
<td>94.91</td>
<td>0.0</td>
<td>5.09</td>
<td>5.09</td>
<td>26</td>
<td>9</td>
<td>42</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>FLPCA</td>
<td>94.91</td>
<td>0.0</td>
<td>5.09</td>
<td>5.09</td>
<td>26</td>
<td>9</td>
<td>42</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

k: Sub-images per image; For modPCA, u is sub-image size and for other methods, m×u. PVs: Number of Projection Vectors (eigenvectors chosen for projection) per sub-image set.