Representing images of a rotating object with cyclic permutation for view-based pose estimation[☆]

Toru Tamaki[a], Toshiyuki Amano[b], Kazufumi Kaneda[a]

[a] *Department of Information Engineering, Hiroshima University*
1-4-1 Kagamiyama, Higashi-hiroshima, Hiroshima, 739-8527 Japan
[b] *Graduate School of Information Science, NAIST*
8916-5 Takayama, Ikoma, Nara, 630-0192 Japan

Abstract

In this paper, we propose a novel approach using a cyclic group to model the appearance change in an image sequence of an object rotated about an arbitrary axis (1DOF out-of-plane rotation). In the sequence, an image $\boldsymbol{x}_j$ is followed by an image $\boldsymbol{x}_{j+1}$. We represent the relationship between images by a cyclic group as $\boldsymbol{x}_{j+1} = G\boldsymbol{x}_j$, and obtain the matrix G by real block diagonalization. Then, G to the power of a real number is used to represent the image sequence and also for pose estimation. Two estimation methods are proposed and evaluated with real image sequences from the COIL-20, COIL-100, and ALOI datasets, and also compared to the Parametric Eigenspace method. Additionally, we discuss the relationship of the proposed approach to the pixel-wise Discrete Fourier Transform (DFT) and to linear regression, and also outline several extensions.

Key words:
view-based pose estimation, global appearance, cyclic group, column permutation matrix, block diagonalization, subspace methods

1. Introduction

When a three-dimensional object rotates about an axis, as shown in Fig. 1, the sequence of images of the object is cyclic: the last image is followed by the first image. When we have that kind of a sequence of n images $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$, this cyclic property can be represented by the action of a cyclic group:

$$\boldsymbol{x}_{j+1 \bmod n} = G\boldsymbol{x}_j, \tag{1}$$

[☆] This paper extends the conference version [1] with additional experimental results and extra detailed discussions.

Email addresses: tamaki@hiroshima-u.ac.jp (Toru Tamaki), amano@is.naist.jp (Toshiyuki Amano), kin@hiroshima-u.ac.jp (Kazufumi Kaneda)

where G is an element of the cyclic group. Although this relationship is essential for images of one parameter (1DOF) rotation, no attention has been paid to it before. Utilizing this cyclic property, we first propose a novel representation of an image sequence of a rotating object, and then use it for pose estimation.

Figure 1: Images of an object obtained by (a) in-plane and (b) out-of-plane rotations.

Previously, several analytical studies have been done on the representation of an image sequence of a rotating object. The primary motivation has been to efficiently compute the Eigenspace of the sequence. For example, Uenohara and Kanade [2] proposed an efficient Eigenspace computation by DCT (or DFT, DHT [3]) for images of an object rotating about the optical axis of the camera as shown in Fig. 1(a). This case corresponds to two-dimensional image rotation, therefore being called *in-plane* rotation [4]. The above analytical method has been extended to handle other in-plane cases, such as translation [5], multiple objects [6], and the case when the number of images becomes infinite [4].

However, extending the above analysis to a general 1DOF rotation is difficult. Usually, an object is rotated about an arbitrary axis as shown in Fig. 1(b), and this rotation is called *out-of-plane* rotation [4]. Previous efforts to analyze and represent out-of-plane rotation [5, 4, 7] have resulted only in approximations and the problem has not been solved yet in its full generality.

In this paper, we propose a novel approach for representing out-of-plane rotation with a cyclic group acting on an image sequence. The proposed method focuses on a transformation from an image to another image in the sequence, and both out-of-plane and in-plane rotations can be represented by the transformation. Our goal here is to use this approach for pose estimation rather than for efficient Eigenspace computation. Applying the first equation (1) several times, we can obtain another equation, $\boldsymbol{x}_j = G^j \boldsymbol{x}_0$. Actually here G is an operator of a cyclic group, but we can think of it as a matrix, and $\boldsymbol{x}$ is an image vector. Then this equation can be regarded as a linear equation. Now, pose estimation can be formulated as the following problem: find some optimal j that gives $\boldsymbol{x} = G^j \boldsymbol{x}_0$ for a given image $\boldsymbol{x}$ with the first image $\boldsymbol{x}_0$. Moreover, we can use the equation in a different way: if j is known then $\boldsymbol{x}_j = G^j \boldsymbol{x}_0$ would represent a novel view.

However, then the following question would arise: How the jth image $\boldsymbol{x}_j$ can be obtained from the first image $\boldsymbol{x}_0$ by just multiplying a matrix j times? For example, when $\boldsymbol{x}_0$ is a frontal image and $\boldsymbol{x}_j$ is an image of the back side, due to self-occlusion, $\boldsymbol{x}_j$ does not seem to have any common information with $\boldsymbol{x}_0$. But

the answer is simple: G is a very large matrix whose dimension is $N \times N$ (N is the number of pixels), and it contains all the necessary information needed to obtain $\boldsymbol{x}_j$ from $\boldsymbol{x}_0$.

G incorporates information about the specific object, rather than being a generic rotation matrix (or operator). This object-specific information comes from n images of the object used to construct G, and this leads to an efficient computation of G^j by block diagonalization with rank-n matrices, instead of $N \times N$ matrix computations. The rank-n matrices relate the matrix G to a subspace spanned by the n images of the object. Intuitively, generating novel views can be seen as DFT of the n images as shown in Section 5.

The decomposition of G by block diagonalization is then used to estimate the pose of an unknown view of the object. In this paper we propose two estimation methods. One is based on a distance in a subspace, and searches for the minimum over all possible values j. The other method is based on an angle in a subspace that uses some properties of block diagonalization.

Some of the limitations of the proposed method need also to be addressed. The proposed methods can be classified as global appearance-based (view-based) pose estimation, similar to [8, 9, 10, 11, 12, 13, 14]. In this paper we only consider 1DOF out-of-plane rotations. Although extending the proposed method to general 3DOF rotations, using cyclic permutations, would not be as straightforward as might be for other global appearance-based methods, possible ways to extend it to 3DOF rotations will also be discussed later on. Another limitation is that the proposed method cannot deal with cases in which the object is occluded or the background is changed. To handle such cases, recent object-specific pose estimation methods [15, 16] and object recognition and localization methods [17, 18, 19, 20, 21, 22, 23, 24] utilize local features for highly cluttered scenes.

The main contribution of this paper lies in demonstrating that the appearance change resulting from out-of-plane 1DOF rotation can be represented by a cyclic permutation, even though the proposed method is based on global appearance of the object. This enables us to simultaneously represent a sequence of images and estimate the pose of an unknown view, something which has never been achieved by any of the conventional pose estimation methods or by the analytical Eigenspace methods.

The paper is organized as follows: the cyclic property of an image sequence formulated by a matrix, and the decomposition of the matrix by block diagonalization are described in section 2. Based on this decomposition, two pose estimation methods are proposed in section 3. Section 4 shows experimental results for pose estimation using image sequences from several datasets. In section 5, we discuss several properties of the proposed formulation from the view point of DFT and regression, and then we conclude the paper in section 6.

2. Formulation of a cyclic image sequence with cyclic permutation

In this section, we describe the proposed method for representing an image sequence by a cyclic permutation. First, we introduce a matrix G to represent

a given cyclic image sequence, and show how the matrix G can be decomposed using a permutation matrix M. Then, the block diagonalization of M is shown and its interpretation from a subspace point of view is given. Finally, we describe how the matrix G can be used to represent object's pose.

2.1. Matrix representation of relationship between images

We represent n consecutive images in a given out-of-plane rotation sequence by vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$. Each image $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jN})^T$ is an N-dimensional vector taken at view angle $\theta_j = \frac{2\pi j}{n}$. Throughout the paper, we assume $N > n$ because usually the number of pixels is larger than the number of images.

First we consider the following transformation with an operator G:

$$\boldsymbol{x}_{j+1 \bmod n} = G\boldsymbol{x}_j, \quad \boldsymbol{x}_j = G^j \boldsymbol{x}_0, \quad \boldsymbol{x}_j = I_G \boldsymbol{x}_j. \tag{2}$$

Here G transforms an image $\boldsymbol{x}_j$ into $\boldsymbol{x}_{j+1}$ as the angle is incremented from θ_j to θ_{j+1}. This transformation can be seen as the action of a cyclic group $G_n = \{I_G, G, G^2, \ldots, G^{n-1}\}$ of degree n, acting from the left side of the image. G is called a generator (or primitive element) of G_n, and I_G is the identity element of G_n.

Although group theoretical transformations represent a much broader mathematical concept, here we restrict our attention only to linear transformations, that is, G is an $N \times N$ matrix. Therefore, the transformation can be written in a matrix form as

$$[\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \cdots \ \boldsymbol{x}_{n-1} \ \boldsymbol{x}_0] = G[\boldsymbol{x}_0 \ \boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_{n-2} \ \boldsymbol{x}_{n-1}], \tag{3}$$

or in a more compact form as

$$X_1 = GX_0, \tag{4}$$

where $X_1 = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \cdots \ \boldsymbol{x}_{n-1} \ \boldsymbol{x}_0]$, and $X_0 = [\boldsymbol{x}_0 \ \boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_{n-2} \ \boldsymbol{x}_{n-1}]$.

We obtain G with X_0^+, a Moore-Penrose generalized (pseudo) inverse of X_0 as follows:

$$G = X_1 X_0^+, \tag{5}$$

$$X_0^+ = (X_0^T X_0)^{-1} X_0^T = V \Sigma^{-1} E^T, \tag{6}$$

where the singular value decomposition of X_0 is $X_0 = E\Sigma V^T$. G is the minimum norm solution to Eq. (4) that is a rank-n approximation, and therefore G itself is not invertible. $X_0^T X_0$ should be full rank, or $\mathrm{rank}(X_0) = n$ for the inverse to be unique. In that case, $X_0^+ X_0 = I_n$, the $n \times n$ identity matrix. Note that this assumption is violated if an image is exactly identical to another image (i.e., repeated texture), or a linear combination of other images (which rarely happens).

When we consider the transformation from X_0 to X_1, it can also be represented by an $n \times n$ column permutation matrix M multiplied to the right side of X_0:

$$X_1 = X_0 M, \quad \text{where} \quad M = \begin{pmatrix} 0 & & & & 1 \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \\ & & & & 1 \end{pmatrix}. \tag{7}$$

Then Eq. (5) can be rewritten as follows:

$$G = X_0 M X_0^+. \tag{8}$$

Note that G represents the action of $X_0 M X_0^+$ on the column space of X_0, that is, the linear subspace spanned by $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$. Since G is an $N \times N$ matrix of rank n, G is not invertible and G^n is not the identity matrix. However, G^n acts as the identity operator I_G if the action is restricted to the column space, which means that $(X_0 M X_0^+)^n \boldsymbol{x}_j = X_0 M^n X_0^+ \boldsymbol{x}_j = \boldsymbol{x}_j$.

2.2. From discrete transformation to continuous rotation

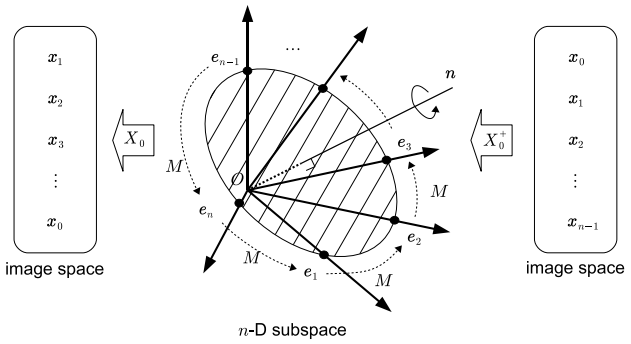


Figure 2: Two projections X_0^+, X_0 and a rotation M composing the transformation G.

In Eq. (8), it is interesting that we can interpret G as a combination of X_0^+, M, and X_0 (see Fig. 2):

1. X_0^+ transforms the sequence X_0 into I_n because of $X_0^+ X_0 = I_n$. This means that the $(j-1)$th image $\boldsymbol{x}_{j-1}$ is mapped to the jth standard unit vector $\boldsymbol{e}_j = (0, \ldots, 0, \overset{j}{\check{1}}, 0, \ldots, 0)^T$.
2. Next, M moves the jth standard unit vector $\mathbf{e}_j$ to the $(j+1)$th standard unit vector $\mathbf{e}_{j+1}$. This can be done by just shifting elements in $\boldsymbol{e}_j$: $M I_n = M$.
3. Finally, X_0 projects the vectors back to image space: $X_0 M = X_1$.

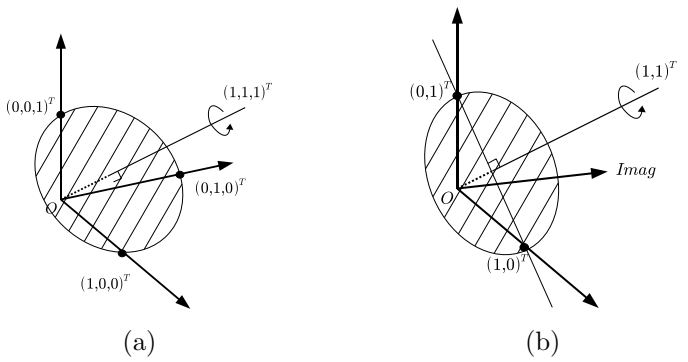


Figure 3: Interpretations of M as a continuous rotation. (a) $n = 3$, (b) $n = 2$.

From this observation, we can see that image $\boldsymbol{x}_j$ is projected by X_0^+ to point $\boldsymbol{e}_j$ in n-D subspace. There, the corresponding points are well separated, with distance $\sqrt{2}$ from each other, because $||\boldsymbol{e}_j - \boldsymbol{e}_k|| = \sqrt{1+1} = \sqrt{2}$ if $j \neq k$.

The permutation matrix M transforms $\boldsymbol{e}_j$ to $\boldsymbol{e}_{j+1}$, and this is a *discrete* transformation because j is just an integer between 0 and $n-1$. Our proposed method further extends the value of j from integer to all real values $0 \leq j < n$, so that any pose between given discrete poses can be represented. This idea is illustrated in Fig. 2. M can be regarded as a rotation in n-D subspace in the hyperplane with equation $x_1 + \cdots + x_n = 1$, which is orthogonal to the vector $(1, 1, \ldots, 1)^T \in \mathbb{R}^n$, and all $\boldsymbol{e}_j$ are on the hyperplane. Therefore, M is a *discrete rotation*, and if we extend it to a *continuous rotation*, then it would make $\boldsymbol{e}_j$ form a hypercircle on the hyperplane.

But in fact, as an $n \times n$ matrix, M is not always a rotation matrix. Indeed, if n is odd, then $|M| = 1$ and M is a rotation matrix; but when n is even, then $|M| = -1$ and M cannot be a rotation matrix. For example, if $n = 3$ in Fig. 3(a), there are three points $(1, 0, 0)^T$, $(0, 1, 0)^T$, and $(0, 0, 1)^T$ on a circle with a center $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$. In the case of $n = 2$, however, there is such a circle for the points $(1, 0)^T$ and $(0, 1)^T$, which are mirror images of each other about the axis $(1, 1)^T$. In this case, we consider an additional axis as an imaginary part, as shown in Fig. 3(b). Now, similar to Fig. 2, M can be regarded as a rotation in the plane with equation $x_1 + x_2 = 1$ that is orthogonal to the vector $(1, 1, 0)^T$, where the last component is considered as being in imaginary space. Although the circle M is no longer in the real space, both points $(1, 0)^T$ and $(0, 1)^T$ are on the circle. Therefore, the concept is applicable to both even and odd cases.

The discussions in the following sections naturally involve complex numbers to extend *discrete* rotation M to *continuous* rotation.

2.3. Decomposition of G

Now, let us consider again the equation $\boldsymbol{x}_j = G^j \boldsymbol{x}_0$. Substituting Eq. (8) into it, it can be written as:

$$\boldsymbol{x}_j = G^j \boldsymbol{x}_0 = (X_0 M X_0^+) \cdots (X_0 M X_0^+) \boldsymbol{x}_0 = X_0 M^j X_0^+ \boldsymbol{x}_0, \tag{9}$$

since $X_0^+ X_0 = I_n$.

To compute M^j for any value j, we decompose M as $M = WDW^T$, with a block diagonal matrix D and an orthogonal matrix W, where

$$D = \begin{cases} \mathrm{diag}(1, A_1, A_2, \ldots, A_s), & n \text{ is odd}, \\ \mathrm{diag}(1, A_1, A_2, \ldots, A_s, -1), & n \text{ is even}, \end{cases} \tag{10}$$

$$W = \begin{cases} \sqrt{\frac{2}{n}} \left(\frac{\boldsymbol{c}_0}{\sqrt{2}}, \boldsymbol{c}_1, \boldsymbol{s}_1, \boldsymbol{c}_2, \boldsymbol{s}_2, \ldots, \boldsymbol{c}_s, \boldsymbol{s}_s \right), & n \text{ is odd}, \\ \sqrt{\frac{2}{n}} \left(\frac{\boldsymbol{c}_0}{\sqrt{2}}, \boldsymbol{c}_1, \boldsymbol{s}_1, \boldsymbol{c}_2, \boldsymbol{s}_2, \ldots, \boldsymbol{c}_s, \boldsymbol{s}_s, \frac{\boldsymbol{c}_{\frac{n}{2}}}{\sqrt{2}} \right), & n \text{ is even}, \end{cases} \tag{11}$$

$$s = \begin{cases} \frac{n-1}{2}, & n \text{ is odd}, \\ \frac{n-2}{2}, & n \text{ is even}, \end{cases} \tag{12}$$

$$A_k = \begin{pmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{pmatrix}, \tag{13}$$

$$\boldsymbol{c}_k = (\cos(n-1)\theta_k, \cos(n-2)\theta_k, \ldots, \cos\theta_k, 1)^T, \tag{14}$$

$$\boldsymbol{s}_k = (\sin(n-1)\theta_k, \sin(n-2)\theta_k, \ldots, \sin\theta_k, 0)^T. \tag{15}$$

Here $\mathrm{diag}(\cdot)$ means that a matrix has blocks in its diagonal part. Details of the block diagonalization of M are given in the Appendix.

By combining $G = X_0 M X_0^+$ and $M = WDW^T$, we have the following decomposition: $G = U_2 D U_1$, where $U_1 = W^T X_0^+$ and $U_2 = X_0 W$.

Here, the matrix U_1 can be regarded as a projection from the image space onto an n-dimensional (n-D) subspace. Each pair of row vectors of U_1 corresponds to A_k. These rows project an image onto the two-dimensional (2-D) subspace spanned by the row vectors. These 2-D subspaces are independent and orthogonal to each other because the blocks in D do not overlap. Therefore, the projection $\boldsymbol{x}'$ obtained by U_1 is produced by a set of projections onto different 2-D subspaces, and multiplying $\boldsymbol{x}'$ by D corresponds to a set of 2-D rotations (with A_k by θ_k).

2.4. G to the power of a real number and its properties

Using the decomposition of G, the transformation from $\boldsymbol{x}_0$ to $\boldsymbol{x}_j$ can be rewritten as:

$$\boldsymbol{x}_j = G^j \boldsymbol{x}_0 = U_2 D^j U_1 \boldsymbol{x}_0, \tag{16}$$

which follows from $U_1 U_2 = I_n$. Therefore, j (the exponent of D^j) determines how much the image $\boldsymbol{x}_0$ is transformed in the image sequence.

We can easily calculate D^j for a real number j because D is a block diagonal matrix and the angles in the 2×2 blocks A_k are just multiplied by j:

$$D^j = \begin{cases} \mathrm{diag}(1, A_1^j, A_2^j, \ldots, A_s^j), & n \text{ is odd}, \\ \mathrm{diag}(1, A_1^j, A_2^j, \ldots, A_s^j, (-1)^j), & n \text{ is even}, \end{cases} \tag{17}$$

$$A_k^j = \begin{pmatrix} \cos j\theta_k & \sin j\theta_k \\ -\sin j\theta_k & \cos j\theta_k \end{pmatrix}. \tag{18}$$

When n is even, the last diagonal element in D becomes a complex number when j is not an integer. As we mentioned before, M can be regarded as a rotation if we use an additional 1-D imaginary space when n is even and j is not an integer. The last element in D corresponds to the imaginary space, and we use Euler's formula to compute it: $(-1)^j = e^{i\pi j} = \cos(\pi j) + i \sin(\pi j)$, where $i = \sqrt{-1}$.

This property is the most useful one in the proposed formulation, because G^j can be calculated by just multiplying the angle θ_k by j. If the block diagonalization were not used, it would not have been so straightforward to compute G^j for any real number j.

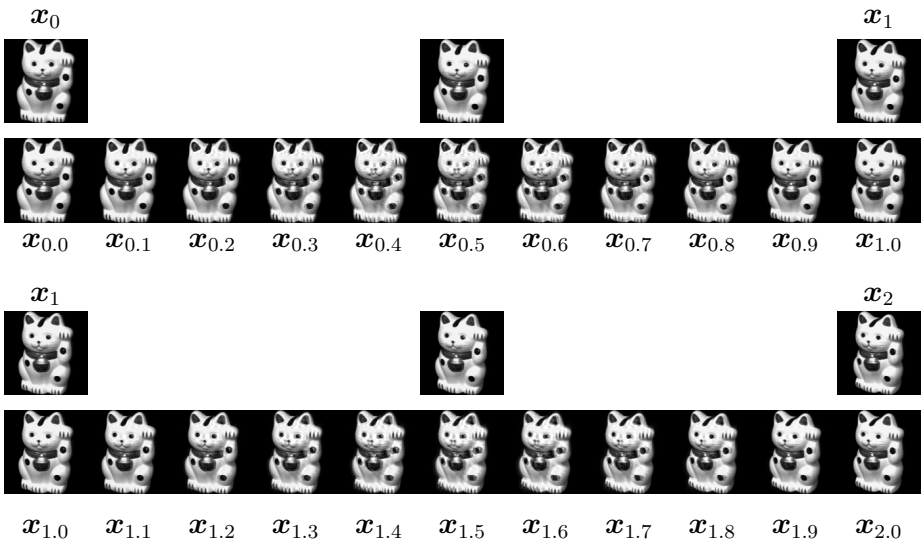


Figure 4: Generated images $\boldsymbol{x}_{0.1j}$. Images in the bottom row are created by repeatedly multiplying a matrix $G^{0.1}$ to the first image $\boldsymbol{x}_0$. Images in the upper row are taken from COIL-20 (object 4). 36 images including $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2$ (0, 10, 20[deg]) are used for learning. Two images (5,15 [deg]) corresponding to $\boldsymbol{x}_{0.5}, \boldsymbol{x}_{1.5}$ are shown for comparison. The lower row shows that 9 images are created between each learned images. The full sequence is available online as a supplemental material.

Now we are interested in extending the range of the exponent j from several integer numbers $(0, 1, \ldots, n-1)$ to the real numbers in $[0, n[$. Fig. 4 demonstrates

an example of view generation for an out-of-plane rotation sequence. After the first image $\boldsymbol{x}_0$, all other images $\boldsymbol{x}_{0.1}, \ldots, \boldsymbol{x}_{2.0}$ were created by Eq. (16). Thirty-six images $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{35}$ were used as training images for computing G, and images in-between (such as $\boldsymbol{x}_{0.5}, \boldsymbol{x}_{1.5}, \ldots$) were not used. Since the number of images is even, the generated images have complex values. In the figure, the moduli of the complex values were used as the gray values of the pixels.

The generated images $\boldsymbol{x}_{1.0}$ and $\boldsymbol{x}_{2.0}$ are exactly the same with the learned images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. For the other images generated between the learned images (for example, see $\boldsymbol{x}_{0.5}$ and $\boldsymbol{x}_{1.5}$ for comparison), the appearances are very similar to the actual intermediate images. They look as if they were made by blending two learned images. However, our goal here is not to make these generated images close to the real ones, but to utilize them for pose estimation. Pose estimation using G^j is described in the next section.

3. Pose estimation

In this section, we propose two methods (a distance-based and an angle-based one) for estimation of the pose of a new image by using the subspace described in the previous section.

3.1. Estimation by searching the minimum distance in the n-D subspace

In the previous section, we have shown that D to the power of a real number j generates images between the learned samples. Based on this observation and Eq. (2), we make the assumption that *a test image* $\boldsymbol{x}$ *matches* $G^j\boldsymbol{x}_0$ *for some* j. Further, we assume that this also holds in the n-D subspace: $\boldsymbol{x}'$ *matches* $D^j\boldsymbol{x}'_0$ *for some* j, where $\boldsymbol{x}' = U_1\boldsymbol{x}$ is a projection of $\boldsymbol{x}$ in the subspace.

For matching, we minimize the Euclidean distance in the n-D subspace:

$$\hat{j} = \underset{j\in[0,n)}{\operatorname{argmin}} \operatorname{dist}(\boldsymbol{x}', j), \quad \text{where} \quad \operatorname{dist}(\boldsymbol{x}', j) = ||\boldsymbol{x}' - D^j\boldsymbol{x}'_0||^2, \tag{19}$$

and pose is estimated as $\hat{\theta} = \theta_{\hat{j}} = \frac{2\pi\hat{j}}{n}$. We call this estimation *the distance-based method*. $\operatorname{dist}(\boldsymbol{x}', j)$ continuously changes as j changes from 0 to n continuously. Since $D^j\boldsymbol{x}_0$ has complex numbers when n is even, the Euclidean distance is defined as $||\boldsymbol{x}||^2 = \boldsymbol{x}^H\boldsymbol{x}$, where ^{H} denotes the transpose of a complex conjugate.

Although this minimization requires an exhaustive search over j and it is computationally expensive, we can use a coarse-to-fine strategy effectively. Fig. 5 shows distances $\operatorname{dist}(\boldsymbol{x}'_\ell, j)$ for some given test image $\boldsymbol{x}_\ell$. As in Fig. 4, only 36 images $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{35}$ were used as training images for computing G, and in-between images ($\boldsymbol{x}_{0.5}, \boldsymbol{x}_{1.5}, \ldots$) were not used during the training. For the learned samples $\ell = 5, 11, 17$, we can see that the distances have sharp minima at $j = \ell$ because $\boldsymbol{x}'_\ell$ and $D^\ell\boldsymbol{x}'_0$ are exactly the same. Even for images which were not learned ($\ell = 22.5, 28.5, 34.5$), the distance has a smooth minimum around the correct pose. The learned images have the same distance $\sqrt{2}$ to each other (as described before). When the exponent j is a real number, the distance deviates from $\sqrt{2}$, but it is so small that the search for minimum is not affected.

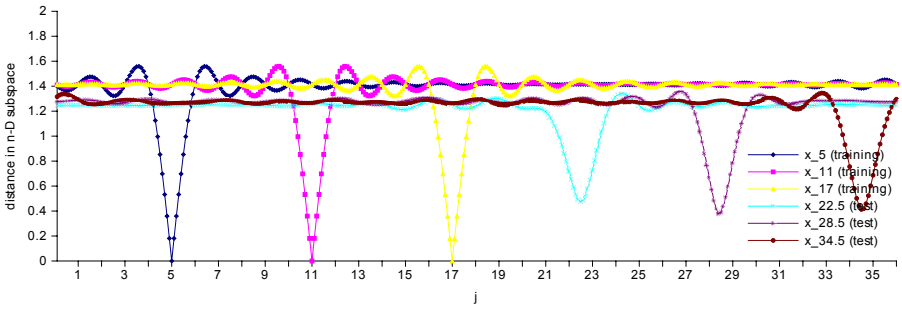


Figure 5: The Euclidean distance $\mathrm{dist}(\boldsymbol{x}'_\ell, j)$ for different values of j, where j is is the exponent of D^j. Some distance curves are shown as examples for learned images $\ell = 5, 11, 17$, and not learned images $\ell = 22.5, 28.5, 34.5$.

Based on this observation, first we search for a minimum of j using a large search step, then search around the minimum again with a smaller step, and gradually the search interval shrinks. This strategy reduces the computational cost and estimation at any pre-defined precision can be achieved.

3.2. Estimation using an angle between two vectors in a 2-D subspace

Here we propose a direct estimation method without any searching (in contrast to the distance-based method from the previous section, which involves iterative search, even if the algorithm is efficient).

As mentioned before, the projection of $\boldsymbol{x}$ by U_1 is a set of projections to many different 2-D subspaces, and in each 2-D subspace the projection is rotated by A_k. Now we focus on two elements corresponding to A_1, because A_1 is a 2×2 rotation matrix of θ_1 (the angle between images in the sequence). Since two images $\boldsymbol{x}_j$ and $\boldsymbol{x}_{j+1}$ are related by A_1 in the 2-D subspace corresponding to A_1, $\boldsymbol{x}_0$ and $\boldsymbol{x}_j$ are related by $j\theta_1$.

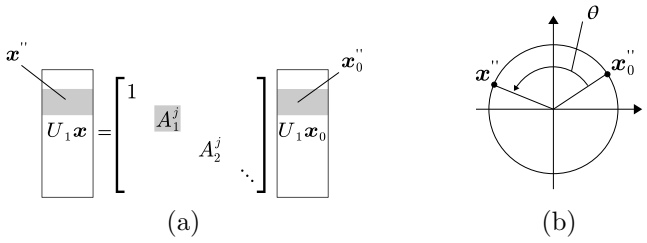


Figure 6: (a) Extraction of $\boldsymbol{x}''$ and $\boldsymbol{x}''_0$ from $\boldsymbol{x}$ and $\boldsymbol{x}_0$. (b) Relation between $\boldsymbol{x}''$ and $\boldsymbol{x}''_0$ in the 2-D subspace.

The two elements $\boldsymbol{x}'', \boldsymbol{x}''_0 \in \mathbb{R}^2$ are extracted from the second and the third

elements (corresponding to A_1) as shown in Fig. 6(a):

$$\boldsymbol{x}'' = \begin{pmatrix} 0\,1\,0\,0\cdots 0 \\ 0\,0\,1\,0\cdots 0 \end{pmatrix} \boldsymbol{x}' = \begin{pmatrix} 0\,1\,0\,0\cdots 0 \\ 0\,0\,1\,0\cdots 0 \end{pmatrix} U_1\boldsymbol{x} = U_1'\boldsymbol{x}, \tag{20}$$

$$\boldsymbol{x}_0'' = \begin{pmatrix} 0\,1\,0\,0\cdots 0 \\ 0\,0\,1\,0\cdots 0 \end{pmatrix} \boldsymbol{x}_0' = \begin{pmatrix} 0\,1\,0\,0\cdots 0 \\ 0\,0\,1\,0\cdots 0 \end{pmatrix} U_1\boldsymbol{x}_0 = U_1'\boldsymbol{x}_0, \tag{21}$$

where

$$U_1' = \begin{pmatrix} 0\,1\,0\,0\cdots 0 \\ 0\,0\,1\,0\cdots 0 \end{pmatrix} U_1, \tag{22}$$

i.e. U_1' is a $2 \times N$ matrix consisting of the second and the third row of U_1.

Those extracted vectors $\boldsymbol{x}'', \boldsymbol{x}_0''$ are just related by A_1 as $\boldsymbol{x}'' = A_1\boldsymbol{x}_0''$ (see Fig. 6(b)). Therefore, the angle θ between $\boldsymbol{x}''$ and $\boldsymbol{x}_0''$ is obtained by solving a system of equations:

$$\boldsymbol{x}'' = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_0'' \\ y_0'' \end{pmatrix} = \begin{pmatrix} x_0'' & y_0'' \\ y_0'' & -x_0'' \end{pmatrix} \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \tag{23}$$

where $\boldsymbol{x}_0'' = (x_0'', y_0'')^T$, and the solution is $\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} = \begin{pmatrix} x_0'' & y_0'' \\ y_0'' & -x_0'' \end{pmatrix}^{-1} \boldsymbol{x}''$. Finally $\theta = \tan^{-1}\left(\frac{\sin\theta}{\cos\theta}\right)$ is the angle between $\boldsymbol{x}''$ and $\boldsymbol{x}_0''$; the pose estimate of $\boldsymbol{x}$.

Note that A_k $(k \geq 2)$ can not be used in the same way as A_1 because the angle between vectors is not uniquely determined. Combining A_k with A_1 is our future work.

4. Experimental results

Figure 7: Images from the databases used in the experiments. (Top) All 20 objects in COIL-20, grayscale, 128×128. (Middle) First 20 of 100 objects in COIL-100, color, 128×128. (Bottom) First 20 of 1000 objects in ALOI, color, 192×144.

The proposed methods were implemented and evaluated on three datasets (Fig. 7): COIL-20 [25] (20 objects), COIL-100 [26] (100 objects), and ALOI [27] (1000 objects). Each object in the datasets has 72 images obtained by rotating the object in 5 degrees steps. The rotation is 1DOF out-of-plane rotation because the rotation axes are not identical to the optical axis. All grayscale images (COIL-20) have been represented as vectors $\boldsymbol{x}$. For color images (COIL-100 and ALOI) with N pixels, RGB values have been stacked to obtain vectors $\boldsymbol{x}$ with $3N$ elements.

For comparison, the Parametric Eigenspace Method [8] (in the following, we call it PEM for short), which is one of the conventional methods, was also implemented. This method is well known and widely used for performance evaluation of 1DOF pose estimation. The dimensionality of the Eigenspace was fixed to 11 for all experiments, based on preliminary experiments. The exhaustive search on a cubic spline in the Eigenspace was done by every 0.1 degrees. All images were normalized as described in [8] so that image vectors have norm 1, whereas the angle-based and distance-based methods do not require the normalization.

Computation times per test image for estimation by our implementation in C++ are about 1 [ms] with a 2.4-GHz CPU. Averages for 1440 images from COIL-20 are 0.0683±0.00454 [ms] for the angle-based method, 1.16±0.0373 [ms] for the distance-based method, and 1.99±0.0703 [ms] for PEM.

4.1. Overall performance of the proposed methods

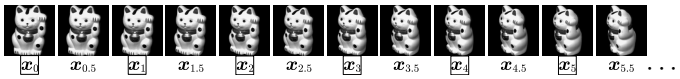


Figure 8: A subset of the images used in the experiments for $n = 36$. $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ are learned (designated by box marks), while $\boldsymbol{x}_{0.5}, \boldsymbol{x}_{1.5}, \ldots$ are test images.

We evaluated the performance of pose estimation by the error (difference) between the estimated and true angles. The root mean square error (RMSE) of pose estimates for an object was calculated for test images only (not including learned images). The number of images n used for learning was set to 36, 24, 18, and 12. For example, in the case of $n = 36$ (see Fig. 8), we used 36 images $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{35}$ corresponding to $0, 10, 20, \ldots, 350$ [deg]. The other 36 images $\boldsymbol{x}_{0.5}, \boldsymbol{x}_{1.5}, \ldots, \boldsymbol{x}_{35.5}$ corresponding to $5, 15, 25, \ldots, 355$ [deg] were used for evaluation.

Fig. 9 compares the performance of the different methods. Each figure shows the average and standard deviation of the RMSE for different n. The number of objects for taking the average is 20 for COIL-20, 100 for COIL-100, and 1000 for ALOI. The averages of the RMSE tend to increase as the number of images n decreases. In each case, the angle-based method is better than the distance-based method. The difference between the angle-based method and PEM seems to be small.

To see the differences, we performed tests for significance with the paired t-test (two-tailed). The results for COIL-20 are not significantly different in most cases because of the small number of objects. In all cases for COIL-100 and ALOI, the angle-based method outperformed the distance-based method. The results for the distance-based method are also worse than those for PEM, which means that the distance in the n-D subspace may not be a good metric for pose estimation.

Overall, the angle-based method seems to be competitive to PEM. The performance of the angle-based method is worse (in the sense of significance) only

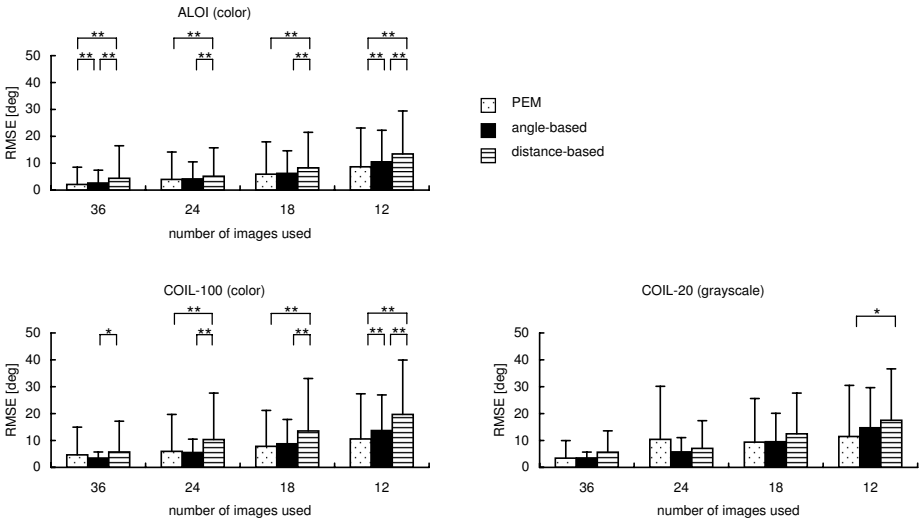


Figure 9: RMSE of pose estimation for three datasets. (Top) ALOI, (Bottom Left) COIL-100, (Bottom Right) COIL-20. RMSE averages with standard deviations of three methods are shown for fixed n. ** stands for significance $p < 0.01$ (and * for $p < 0.05$) by the paired t-test.

in three out of eight cases: $n = 12$ for COIL-100, and $n = 36, 12$ for ALOI. In all other cases, there is no significant difference between the two methods.

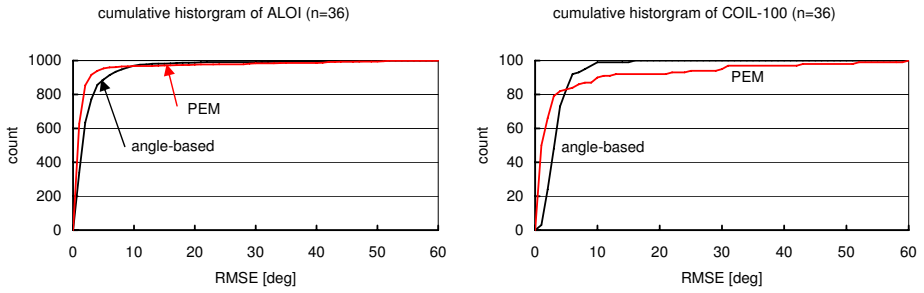


Figure 10: Cumulative histograms of RMSE for ALOI and COIL-100.

However, the standard deviations of the angle-based method are smaller than those of PEM in all cases, and this means that the angle-based method may be more stable than PEM. This fact may be supported by the cumulative histograms of RMSE shown in Fig. 10 for two cases: $n = 36$ for ALOI in which PEM outperforms the angle-based ($p < 0.01$), and $n = 36$ for COIL-100. In both cases, the cumulative histograms of PEM have longer tails than those of the angle-based method. This observation shows that: (1) in most of the

estimations ($80 \sim 95$ %) the RMSEs of the angle-based method are larger than those of PEM, and (2) in few cases PEM has quite a large RMSE, while the angle-based method does not have so many large errors. We will discuss later the reason for this by analyzing the worst cases.

4.2. Performance comparison with 20 objects

Next, we focus on the results for each object in COIL-20 because averages of the RMSE do not show how the methods differ for a certain type of objects and for what types the methods work well.

Fig. 11 shows the RMSEs for 20 objects. In general, all methods tend to have larger error as n decreases. The angle-based method performs as well as PEM when $n = 36$, but the distance-based method has a large error for some objects (object 6, and 12).

When $n \leq 24$, the angle-based method is not as good, especially for objects 6, 9, and 19. But there are still moderate cases, such as object 1, 4, and 20. Table 1 shows the RMSEs for object 4. If an error up to 5 [deg] is acceptable for an estimation, this moderate case needs at least 12 images ($n = 12$) for the proposed methods. Of course, the number of images for a good estimation depends on the type of objects. $n = 36$ is required for satisfactory results for all cases.

Table 1: RMSE [deg] for object 4 in COIL-20.

n	36	24	18	12	9	8	6	4	3	2
PEM	0.794	1.22	1.71	1.94	2.41	3.06	11.5	35.4	42.7	90.6
angle-based	1.23	1.64	2.42	3.20	6.23	6.75	12.3	29.1	38.2	74.2
distance-based	1.80	2.23	2.91	4.32	5.74	6.49	8.89	30.3	38.2	74.2

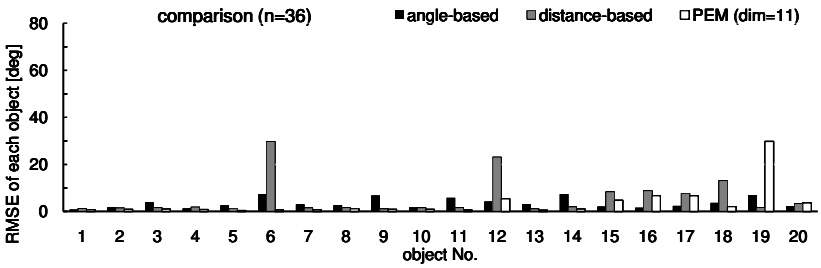
Note that the results for PEM on COIL-20 reported here seem to be quite different from those described in [8]. The reason is that in [8], a global Eigenspace has been used for estimation: that is, all 72 images of all 20 objects have been used to construct a single Eigenspace. This means that the number of images used for constructing the Eigenspace is very large (totally 1440). In contrast, the experiments reported here used a small number of images (up to 36 images).

4.3. Analysis of the worst cases

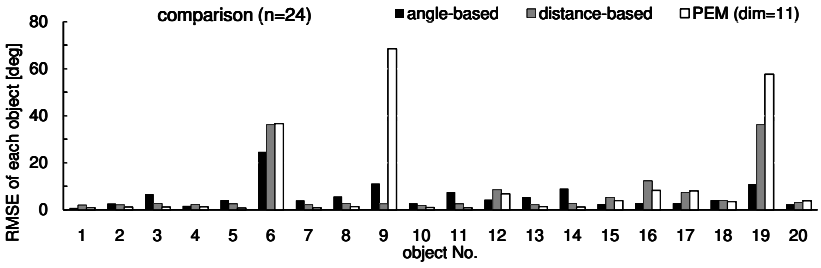
Here we discuss the cases where the proposed methods do not work well. According to the discussion above, we chose object 6 for the worst case.

First, we discuss the performance of the distance-based method and compare it to PEM, because the two methods are similar: both use distance in a subspace where a test image is projected, and compute a point in a pose manifold which gives the minimum distance. But they use different subspaces and different interpolation methods: PEM uses PCA and cubic spline, while the distance-based method uses DFT (as shown in Section 5).

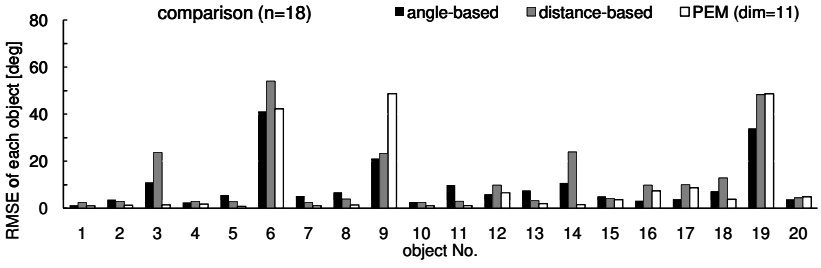
Fig. 12 shows distances to pose manifolds ($n = 36, 24$) from two test images of object 6 at 35 and 275 [deg]. There are two minima for each image because



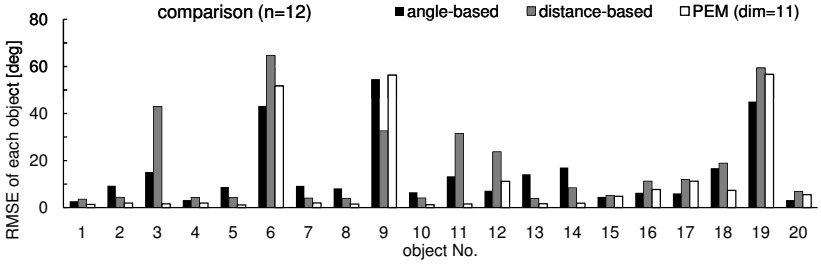
(a)



(b)



(c)



(d)

Figure 11: Estimation results for angle-based, distance-based and Parametric Eigenspace methods for (a) $n = 36$, (b) $n = 24$, (c) $n = 18$, (d) $n = 12$.

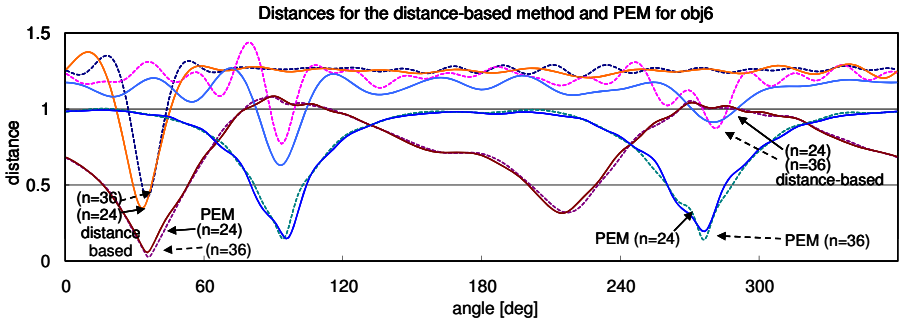


Figure 12: Comparison of the distance-based method and PEM for object 6 ($n = 36, 24$). The horizontal axis is pose angle, and the vertical axis is the distance to the pose on the manifold.

the appearance of object 6 is quite similar to that from the opposite side (180 [deg]). The image at 35 [deg] has a minimum around 215 [deg], but the minimum around 35 [deg] is still smaller than the other, and the pose is estimated (almost) correctly by both PEM and the distance-based method. However, the image at 275 [deg] has a minimum around 95 [deg], and this makes the estimation completely wrong. This is the reason why PEM and the distance-based method have large variances of RMSE.

This "similar appearance from opposite side" effect occurs for objects 6, 9, and 19, which have oblong shapes. The distance-based method is affected by this effect. The distance curves in Fig. 12 for the distance-based method are not as flat as those in Fig. 5, which means that it is difficult to find the correct minimum. Also, it is sensitive to the number of learning images because the distance differs greatly when n changes from 36 to 24. In contrast, the performance of PEM is bettter although it is also affected by this effect. The reason may be that the distance to the pose manifold retains a similar shape when n changes.

The angle-based method should be also affected by this effect, but in a different way. Eq. (21), the definition of a 2-D vector $\boldsymbol{x}''$ used for the angle-based method, can be rewritten as: $\boldsymbol{x}'' = U_1'\boldsymbol{x} = \begin{pmatrix} \boldsymbol{c}_1^T \\ \boldsymbol{s}_1^T \end{pmatrix} X_0^+ \boldsymbol{x}$ (see Section 5 for a detailed explanation). This means that vector $\boldsymbol{x}''$ is actually a linear combination of $(\cos j\theta_1, \sin j\theta_1)^T$, with different phases of cos and sin, with a weight $\boldsymbol{b} = X_0^+ \boldsymbol{x}$. For a learned image $\boldsymbol{x}_0$, $\boldsymbol{b}$ becomes a standard vector $\boldsymbol{e}_1$ and the estimated pose is exactly correct because the inappropriate terms are eliminated from the linear combination. If the weight has a large value around a correct phase for a test image to be estimated, then as a result of the linear combination the phase comes close to the true phase of the pose.

Fig. 13 shows weights for two test images at 35 and 275 [deg] for object 6, and also for object 4 for comparison. The weights for object 4 behave as we expect: the two test images have larger values around the correct phases than for other phases. However, the image at 275 [deg] of object 6 has large weights at a phase opposite to the true pose, and as a result, pose estimation

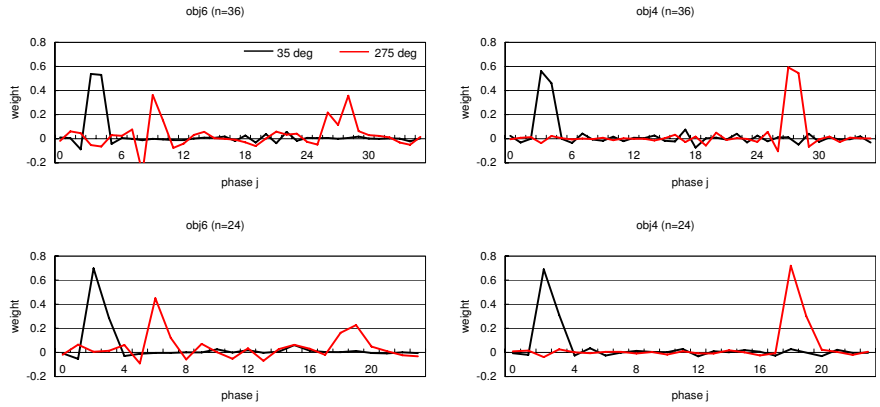


Figure 13: Weights of angle-based method for (Top) image at 35 [deg], (Bottom) image at 275 [deg]. $n = 36, 24$. Horizontal axis is j of phase $j\theta_1$, and vertical axis is the weight value.

fails for such oblong objects. Weight $\boldsymbol{b}$ which includes the pseudoinverse of X_0 is sensitive to similarity (or correlation) between images. Further discussion on the pseudoinverse can be found in [28, 29].

4.4. Estimation for noisy images

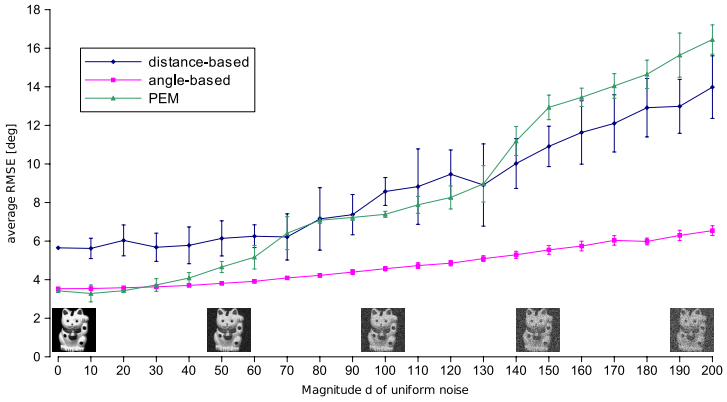


Figure 14: RMSEs of estimation with std. for 10 trials for noisy images ($n = 36$). The horizontal axis is the magnitude $[-d, d]$ of uniform noise added. The vertical axis is the average RMSE for 20 objects in COIL-20 for only test images with noise.

The proposed methods estimate pose exactly for the learned images, but if we get the image at 0 [deg] again after learning, it is not the one used for learning, because of image noise.

Fig. 14 compares the robustness of the methods for the noisy images shown in the figure. These images are contaminated by uniform noise without any intensity normalization (negative pixel values and large values also included). The range of pixel values are between 0 and 255, and the magnitude of the uniform noise is up to ± 200. Even when the largest uniform noise is added, the average RMSE of the angle-based method is less than 7 [deg], while error for the distance-based method increases to 14 [deg]. The performance degradation for PEM may be caused by the normalization required by PEM.

This result demonstrates how robust the angle-based method is. The reason for this lies in the weight of different phases of sin and cos, mentioned above. Contribution from noise to the weight is small relative to that from pixel values from the image. Even if noise is not so small, weights from noise might cancel each other. Then, the weights from the image have larger contribution to the estimate pose.

5. Discussion

Here we discuss some topics related to the proposed method. We have shown that the matrix G is decomposed into $U_2 D U_1$, and the power of D to the real number j is used to generate images in-between the learned sample images. D^j is also used for pose estimation. Now, we will discuss what does the decomposition mean, and to what kind of image sequences the proposed method is applicable.

5.1. Interpreting G as a DFT

First, we show why we can generate $\boldsymbol{x}_{j+1}$ by $G\boldsymbol{x}_j$.

This generation can be seen as *a DFT for each pixel* because $U_2 = X_0 W$. The first row of X_0 is a function of the first pixel, and the first column of W is the first basis function of a DFT basis. Therefore, U_2 stores coefficients for all pixels, and element j-k in U_2 is the coefficient of the jth pixel for the kth basis function of DFT.

Using these coefficients, the original images can be reconstructed with the DFT basis. Since pixel value functions are stored in rows in X_0 (not in column), the transpose of W is used as $X_0 = U_2 W^T$. To reconstruct X_1, shift the column of W^T (not W) by one because X_1 is just a column shift of X_0 by one: this can be done with $W^T M$. This means shifting the phase of the basis W^T by one step $\theta_1 = \frac{2\pi}{n}$.

As a result, X_1 is reconstructed by $U_2 W^T M$. Here $W^T M$ can be replaced with DW^T because $M = WDW^T$. Then we have $X_1 = U_2 DW^T$, and finally multiply it by $X_0^+ X_0 = I_n$ from the right side,

$$X_1 = U_2 DW^T X_0^+ X_0 = U_2 D U_1 X_0 = G X_0, \tag{24}$$

and this is the reason why Eq. (4) holds.

For the equation above, the first assumption $N > n$ is necessary: the number of pixels N of an image is larger than the number of images n. In this case, Eq. (4) is an under-determined system, and of course, the pseudoinverse in

Eq. (5) is not a unique (it is unique in the sense that a minimum norm solution is given). However, if $N < n$ then the solution of Eq. (5) becomes least-square solution and Eq. (4) does not hold.

5.2. Shifting the phase of the DFT basis with D^j

Here we show that in Eq. (24) D^j shifts the basis in phase continuously, while D shifts it discretely. We focus on the k-th pair of rows in $D^j W^T$: $\boldsymbol{c}_k^T$ and $\boldsymbol{s}_k^T$ corresponding to A_k (other rows are not involved because D is block diagonal). The phases of $\boldsymbol{c}_k$ and $\boldsymbol{s}_k$ are shifted by multiplying with A_k from the left side:

$$A_k^j \begin{pmatrix} \boldsymbol{c}_k^T \\ \boldsymbol{s}_k^T \end{pmatrix} = \begin{pmatrix} \cos j\theta_k & \sin j\theta_k \\ -\sin j\theta_k & \cos j\theta_k \end{pmatrix} \begin{pmatrix} \cos(n-1)\theta_k, \ \dots \\ \sin(n-1)\theta_k, \ \dots \end{pmatrix}, \tag{25}$$

$$= \begin{pmatrix} \cos j\theta_k \cos(n-1)\theta_k + \sin j\theta_k \sin(n-1)\theta_k, \ \dots \\ \cos j\theta_k \sin(n-1)\theta_k - \sin j\theta_k \cos(n-1)\theta_k, \ \dots \end{pmatrix}, \tag{26}$$

$$= \begin{pmatrix} \cos(n-1-j)\theta_k, \ \dots \\ \sin(n-1-j)\theta_k, \ \dots \end{pmatrix}. \tag{27}$$

Therefore, D^j shifts the DFT basis W in phase *continuously* by $j\theta_k$, while D does so discretely by θ_k.

As a result, each pixel in the generated image sequence $\boldsymbol{x}_j = G^j \boldsymbol{x}_0$ (as shown in Fig. 4) changes its value according to a sinusoidal curve. In other words, the pixel values in an intermediate image are calculated by interpolation with DFT. This is different from image blending that uses the same weights for all pixels.

5.3. Generating images in sequences with varying illumination

As shown above, the proposed scheme uses pixel-wide DFT to reconstruct images with continuously phase-shifted DFT basis. From this point of view, Eq. (4) can be applied to any cyclic image sequence, for example one obtained as a light source turns around in front of a face (even though the proposed method is formulated for a single axis rotation).

Fig. 15 illustrates an example of face images with different light directions. However, discussing the estimation of the light direction is out of the scope of this paper. This example implies that the proposed method can be also used for estimating illumination change.

5.4. On the estimations

Here we show how U_1 can be used for the estimation in the angle-based method. It uses U_1' which is constructed from two rows extracted from U_1. Expanding Eq. (22), we obtain the following relationship:

$$U_1' = \begin{pmatrix} 0\,1\,0\,0\,\cdots\,0 \\ 0\,0\,1\,0\,\cdots\,0 \end{pmatrix} U_1 = \begin{pmatrix} 0\,1\,0\,0\,\cdots\,0 \\ 0\,0\,1\,0\,\cdots\,0 \end{pmatrix} W^T X_0^+ = \begin{pmatrix} \boldsymbol{c}_1^T \\ \boldsymbol{s}_1^T \end{pmatrix} X_0^+. \tag{28}$$

Multiplying both sides by X_0, we obtain a system of equations $\begin{pmatrix} \boldsymbol{c}_1^T \\ \boldsymbol{s}_1^T \end{pmatrix} = U_1' X_0$, and Eq. (28) is its solution. In this system, each image $\boldsymbol{x}_j$ on the right-hand

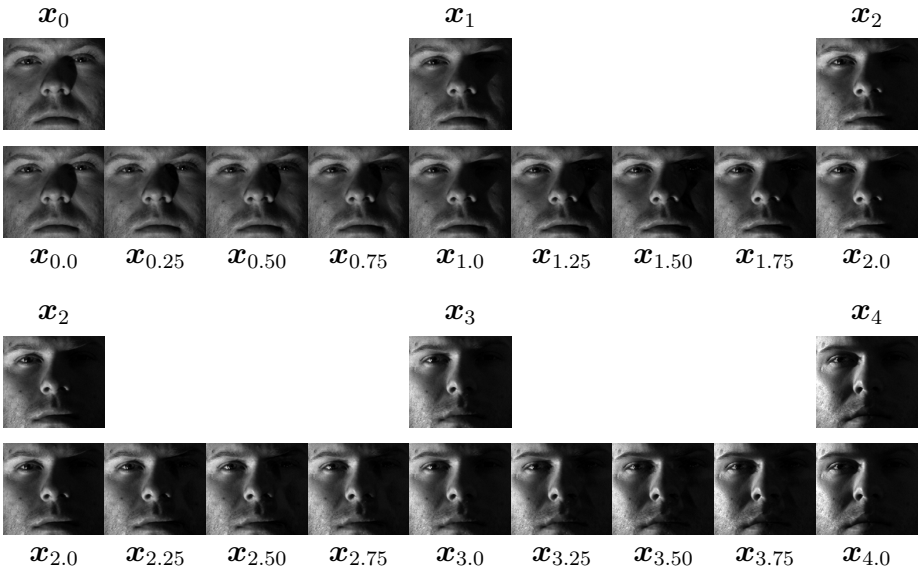


Figure 15: Images created by multiplying the first image $\boldsymbol{x}_0$ with the matrix $G^{0.25}$. The images in the upper row are taken from the Yale face database B [30, 31]. 20 images including $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5$ are used for learning. The lower row shows every three images created between each pair of the learned images. The full sequence is available online as a supplemental material.

side (a column of X_0) is related by U_1' to the corresponding pose parameters $(\cos j\theta_1, \sin j\theta_1)^T$ (the order of elements is reversed from Eq. (14) and (15) to simplify explanation). Since the system has two equations for a number of unknowns, it is underdetermined and the solution is a minimum-norm solution. Therefore, the angle-based method is theoretically identical with linear underdetermined regression[9].

Similarly, $U_1 = W^T X_0^+$ (the definition of U_1) is a minimum-norm solution of the system of equations $W^T = U_1 X_0$. In this system, each image $\boldsymbol{x}_j$ is related by U_1 to the corresponding pose parameters $(1, \cos j\theta_1, \sin j\theta_1, \cos j\theta_2, \sin j\theta_2, \ldots, \cos j\theta_s, \sin j\theta_s)^T$ where n is odd (again the order of the components is reversed from that in Eq. (14) and (15) to simplify explanation). The distance-based method compares the Euclidean distance between vectors in this form.

5.5. Overview of the proposed scheme and its extensions

Above we have shown what the matrix G^j actually does, and here we summarize that information. G^j can be thought of as consisting of three parts:

- Phase estimation part: U_1 estimates the phase $j\theta_1$ corresponding to image $\boldsymbol{x}_j$. This phase information is stacked in cos and sin vectors.
- Phase shifting part: D^j shifts the phase of the vector.

- Reconstruction part: U_2 stores the coefficients for all images, and reconstructs an image from the phase-shifted vector.

Now we outline several possible extensions based on this decomposition into three building blocks.

One straightforward extension would be to replace the estimation part U_1 with any other estimation method, such as CCA [12, 32], nonlinear regression[11, 33], Gaussian processes [34, 35], or even using the Parametric Eigenspace method [8]. Subspace based methods [36] might be more useful because certain preferable properties could be naturally included. For example, classification can be done simultaneously [37], robust recognition can be achieved under occlusion and nonuniform background [38, 39], and so on.

Another extension would address the case when the rotation angle increment is not constant. Until now, we have assumed that the images are taken at regular angles. The proposed method can estimate poses for non-regular samples, but the estimates should be corrected. For example, if $n = 4$ but the given images are taken at 0, 89, 180, and 270 [deg]. The image at 89 [deg] is estimated as 90 [deg] and the image at 90 [deg] may be estimated to be around 91 [deg]. This error is not random — it is systematically distributed among the images from 0 to 180 [deg]. Images at 0 and 180 [deg] are estimated correctly. Therefore, the error is so small and the estimation is not sensitive to the irregularity. To correct for the systematic error, we can introduce a function that converts 0, 89, 180, and 270 [deg] to 0, 90, 180, and 270 [deg]. For general non-regular rotation angles, we may replace the regular DFT with a non-regular DFT that would be applicable to nonequispread data [40].

The most interesting extension would be to deal with 3DOF rotation. In general, a pose of an object is described as a 3×3 rotation matrix R in $SO(3)$. When we have images $\boldsymbol{x}_j$ for different poses R_j, it is necessary to find a way how to fit them into the proposed scheme. To do so, we have to realize that the 1DOF case (this paper) uses the DFT basis that is an orthonormal basis for periodic functions defined on S^1 (the unit circle). U_1 estimates the phase of the basis for all frequencies, and U_2 uses the basis for reconstruction. Therefore, we may use an orthonormal basis for a function defined on $SO(3)$, the so-called spherical functions [41]. This is similar to the spherical harmonics [42, 43] which form an orthonormal basis for S^2 (the unit sphere). Using spherical functions for $SO(3)$, we may modify U_1 to estimate the phases of the spherical functions for all frequencies, and D^j to shift the phases, then U_2 would reconstruct with coefficients that are estimated in advance. Reconstruction and coefficient estimation are well developed for spherical harmonics (such as [42]), and we can use these techniques. However, the phase shifting D^j may no longer be a (block) diagonal matrix, but rather something like a procedure to interpolate 3DOF pose [44, 45]. This extension is the subject of our future work. In addition to rotation, translation can be incorporated with a basis for rigid motion[46].

6. Conclusions

We have proposed a framework to use a cyclic group for appearance change of an image sequence of a rotating (1DOF, out-of-plane) object. The matrix G that transforms one image in the sequence to another, is decomposed as $U_2 D U_1$ by block diagonalization of the column permutation matrix M. Then, we extended the exponent j of G^j from an integer to a real number, and showed how G to the power of a real number j transforms the images. In section 3, we proposed two methods for pose estimation. One is the distance-based method that finds the minimum distance in n-D subspace, and the other is the angle-based method, that uses an angle between two vectors in 2-D subspace. Experimental results with real datasets of 1120 objects in total demonstrated that the angle-based method is robust against noise and performs better than the distance-based method. In Section 5, we have shown the relation of G^j to linear regression and pixel-wise DFT. A limitation of the proposed method in addition to those described in the introduction is that it works only for sequences in which the images are revolved. There are many inapplicable cases, for example, a face sequence taken from the left side to the right side, with a frontal face included, would have no images of the back of the head. However, we have demonstrated an example of view generation for a non-rotation sequence.

To summarize, the cyclic property of the proposed formulation is based on DFT, and the proposed angle-based estimation method is equivalent to a linear regression. However, it should be noted that before the proposed formulation no other methods have ever combined DFT and regression in a unified approach, and this is the main contribution of this paper.

Acknowledgments

We would like to thank Hiroshi Tamaru and Yumiko Ichihara at Hiroshima University for discussions on the subspaces. Also we are grateful to Hitoshi Sakano at NTT Communication Science Laboratories and Bisser Raytchev at Hiroshima University for their comments which have helped us to improve the current paper. This work was supported in part by KAKENHI (20700163) from JSPS.

References

[1] T. Tamaki, T. Amano, K. Kaneda, The secret of rotating object images —using cyclic permutation for view-based pose estimation—, Proc. of Subspace2007 (2007) 24–31 `http://ir.lib.hiroshima-u.ac.jp/00020419`.

[2] M. Uenohara, T. Kanade, Optimal approximation of uniformly rotated images: Relationship between Karhunen-Loeve expansion and discrete cosine transform, IEEE Trans. on Image Processing 7 (1) (1998) 116–119, `http://www.ri.cmu.edu/pubs/pub_928.html`.

[3] R.-H. Park, Comments on "Optimal approximation of uniformly rotated images: Relationship between Karhunen-Loève expansion and discrete cosine transform", IEEE Trans. on Image Processing 11 (3) (2002) 332–334, http://www.ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=988965&isnumber=21305.

[4] M. Sengel, H. Bischof, Efficient representation of in-plane rotation within a PCA framework, Image and Vision Computing 23 (2005) 1051–1059, http://dx.doi.org/10.1016/j.imavis.2005.07.007.

[5] C.-Y. Chang, A. Maciejewski, V. Balakrishnan, Fast eigenspace decomposition of correlated images, IEEE Trans. on Image Processing 9 (9) (2000) 1937–1949, http://www.ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=877214.

[6] M. Jorgan, E. Žagar, A. Leonardis, Karhunen-Loéve expansion of a set of rotated templates, IEEE Trans. on Image Processing 12 (7) (2003) 817–825, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=27275&arnumber=1212657&count=12&index=8.

[7] K. Saitwal, A. A. Maciejewski, R. G. Roberts, Eigendecomposition of correlated images characterized by three parameters, 2006 IEEE Southwest Symposium on Image Analysis and Interpretation (2006) 203–207 http://www.engr.colostate.edu/~aam/pdf/conferences/91.pdf.

[8] H. Murase, S. K. Nayar, Visual learning and recognition of 3-D objects from appearance, International Journal of Computer Vision 14 (1) (1995) 5–24, http://dx.doi.org/10.1007/BF01421486.

[9] T. Okatani, K. Deguchi, Yet another appearance-based method for pose estimation based on a linear model, IAPR Workshop on Machine Vision Applications 2000 (2000) 258–261 http://www.mva-org.jp/Proceedings/CommemorativeDVD/2000/papers/2000258.pdf.

[10] T. Amano, T. Tamaki, An appearance based fast linear pose estimation, IAPR Conference on Machine Vision Applications 2009 (2009) 182–186 http://www.mva-org.jp/Proceedings/2009CD/papers/06-03.pdf.

[11] S. Ando, Y. Kusachi, A. Suzuki, K. Arakawa, Appearance based pose estimation of 3D object using support vector regression, ICIP2005 1 (2005) I–341–344, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1529757&isnumber=32660.

[12] T. Melzer, M. Reiter, H. Bischof, Appearance models based on kernel canonical correlation analysis, Pattern Recognition 36 (2003) 1961–1971, http://dx.doi.org/10.1016/S0031-3203(03)00058-X.

[13] L.-W. Zhao, S.-W. Luo, L.-Z. Liao, 3D object recognition and pose estimation using kernel PCA, Proc. of Intl. Conf. Machine Learning and

Cybernetics 5 (2004) 3258–3262, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1378598.

[14] T. Vik, F. Heitz, P. Charbonnier, Robust pose estimation and recognition using non-gaussian modeling of appearance subspaces, IEEE Trans. on Pattern Analysis and Machine Intelligence 29 (5) (2007) 901–905, http://doi.ieeecomputersociety.org/10.1109/TPAMI.2007.1028.

[15] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110, http://www.springerlink.com/content/h4l02691327px768.

[16] F. Rothganger, S. Lazebnik, C. Schmid, J. Ponce, 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints, International Journal of Computer Vision 66 (3) (2006) 231–259, http://www-cvr.ai.uiuc.edu/ponce_grp/publication/paper/ijcv04d.pdf.

[17] V. Ferrari, T. Tuytelaars, L. V. Gool, Simultaneous object recognition and segmentation from single or multiple model views, International Journal of Computer Vision 67 (2) (2006) 159–188, http://dx.doi.org/10.1007/s11263-005-3964-7.

[18] A. Kushal, J. Ponce, Modeling 3D objects from stereo views and recognizing them in photographs, ECCV2006 (2006) 563–574 http://www-cvr.ai.uiuc.edu/~kushal/papers/conf/ECCV06/eccv06.pdf.

[19] A. Kushal, C. Schmid, J. Ponce, Flexible object models for category-level 3D object recognition, CVPR2007 (2007) 1–8 http://www2.computer.org/portal/web/csdl/doi/10.1109/CVPR.2007.383149.

[20] H.-P. Chiu, L. P. Kaelbling, T. Lozano-Pérez, Virtual training for multi-view object class recognition, CVPR2007 (2007) 1–8 http://www2.computer.org/portal/web/csdl/doi/10.1109/CVPR.2007.383044.

[21] P. Yan, S. M. Khan, M. Shah, 3D model based object class detection in an arbitrary view, ICCV2007 (2007) 1–6 http://www2.computer.org/portal/web/csdl/doi/10.1109/ICCV.2007.4409042.

[22] D. Hoiem, C. Rother, J. Winn, 3D LayoutCRF for multi-view object class recognition and segmentation, CVPR2007 (2007) 1–8 http://www2.computer.org/portal/web/csdl/doi/10.1109/CVPR.2007.383045.

[23] S. Savarese, L. Fei-Fei, 3D generic object categorization, localization and pose estimation, ICCV2007 (2007) 1–8 http://www2.computer.org/portal/web/csdl/doi/10.1109/ICCV.2007.4408987.

[24] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, L. V. Gool, Towards multi-view object class detection, CVPR2006 2 (2006) 1589–1596, http://www2.computer.org/portal/web/csdl/doi/10.1109/CVPR.2006.311.

[25] S. A. Nene, S. K. Nayar, H. Murase, Columbia object image library (COIL-20), Tech. Rep. CUCS-005-96, Columbia University, http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php (1996).

[26] S. A. Nene, S. K. Nayar, H. Murase, Columbia object image library (COIL-100), Tech. Rep. CUCS-006-96, Columbia University, http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php (1996).

[27] J. M. Geusebroek, G. J. Burghouts, A. W. M. Smeulders, The Amsterdam library of object images, International Journal of Computer Vision 61 (1) (2005) 103–112, http://www.science.uva.nl/~mark/pub/2005/GeusebroekIJCV05a.pdf.

[28] A. Ben-Israel, T. N. E. Greville, Generalized Inverses: Theory and Applications, Wiley, 1977.

[29] S. L. Campbell, C. D. Meyer, Generalized Inverses of Linear Transformations, Dover, 1991.

[30] A. S. Georghiades, The Yale face database B, online, http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html (accessed 2008.6.9).

[31] A. S. Georghiades, P. N. Belhumeur, D. J. Kriegman, From few to many: Illumination cone models for face recognition under variable lighting and pose, IEEE Trans. on Pattern Analysis and Machine Intelligence 23 (6) (2001) 643–660, http://doi.ieeecomputersociety.org/10.1109/34.927464.

[32] T. Sun, S. Chen, Locality preserving CCA with applications to data visualization and pose estimation, Image and Vision Computing 25 (5) (2007) 531–543, http://www.sciencedirect.com/science/article/B6V09-4K9C6GM-2/2/a1930d130b2c0f9fec65465579d50d11.

[33] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, http://www.kernel-machines.org/papers/tr-30-1998.ps.gz (1998).

[34] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006, http://www.gaussianprocess.org/gpml/chapters/.

[35] C. E. Rasmussen, Advances in gaussian processes, NIPS2006 Tutorial, http://www.kyb.tue.mpg.de/bs/people/carl/gpnt06.pdf (2006).

[36] F. De la Torre, Component analysis for computer vision, ECCV2006 Tutorial, http://eccv2006.tugraz.at/tutorials.html#M1 (2006).

[37] S. Fidler, D. Skočaj, A. Leonardis, Combining reconstructive and discriminative subspace methods for robust classification and regression by subsampling, IEEE Trans. on Pattern Analysis and Machine Intelligence 28 (3) (2006) 337–350, http://doi.ieeecomputersociety.org/10.1109/TPAMI.2006.46.

[38] A. Leonardis, H. Bischof, Robust recognition using eigenimages, Computer Vision and Image Understanding 78 (1) (2000) 99–118, http://dx.doi.org/10.1006/cviu.1999.0830.

[39] C.-Y. Chang, A. A. Maciejewski, V. Balakrishnan, R. G. Roberts, K. Saitwal, Quadtree-based eigendecomposition for pose estimation in the presence of occlusion and background clutter, Pattern Analysis & Applications 10 (1) (2007) 15–31, http://dx.doi.org/10.1007/s10044-006-0046-6.

[40] D. Potts, G. Steidl, M. Tasche, Fast fourier transforms for nonequispaced data: A tutorial, in: J. J. Benedetto, P. J. S. G. Ferreira (Eds.), Modern Sampling Theory: Mathematics and Applications, Birkhäuser, 2001, Ch. 12, pp. 249–274, http://www.math.mu-luebeck.de/mitarbeiter/potts/paper/ndft.pdf.

[41] M. Takeuchi, Modern Spherical Functions, American Mathematical Society, 1994.

[42] P.-M. Lam, C.-S. Leung, T.-T. Wong, Noise-resistant fitting for spherical harmonics, IEEE Trans. Visualization and Computer Graphics 12 (2) (2006) 254–265, http://doi.ieeecomputersociety.org/10.1109/TVCG.2006.34. doi:http://doi.ieeecomputersociety.org/10.1109/TVCG.2006.34.

[43] A. Matheny, D. B. Goldgof, The use of three- and four-dimensional surface harmonics for rigid and nonrigid shape recovery and representation, IEEE Trans. on Pattern Analysis and Machine Intelligence 17 (10) (1995) 967–981, http://doi.ieeecomputersociety.org/10.1109/34.464561.

[44] K. Shoemake, Animating rotation with quaternion curves, SIGGRAPH'85 19 (3) (1985) 245–254, http://doi.acm.org/10.1145/325165.325242. doi:http://doi.acm.org/10.1145/325165.325242.

[45] J. Morrison, Quaternion interpolation with extra spins, Graphics Gems III (1992) 96–97.

[46] R. Lenz, Group Theoretical Methods in Image Processing, Vol. 413 of Lecture Notes in Computer Science, Springer Verlag, 1990, http://staffwww.itn.liu.se/~reile/LNCS413/index.htm.

[47] I. Satake, Linear Algebra, Marcel Dekker Inc., 1975.

Appendix

A. Complex diagonalization of M

Here we show how matrix M can be diagonalized. This diagonalization of M is nothing new, it is just a standard exercise in linear algebra [47]. It can also be found in studies on the analytical Eigenspace approach [5, 3, 6].

The characteristic equation of M is:

$$|M - \lambda I| = (-1)^n \begin{vmatrix} \lambda & & & -1 \\ -1 & \lambda & & \\ & -1 & \lambda & \\ & & \ddots & \ddots \\ & & & -1 \ \lambda \\ & & & -1 \ \lambda \end{vmatrix} = (-1)^n(\lambda^n - 1), \tag{29}$$

so the eigenvalues λ are ζ_n, n different primitive nth roots of unity: $\lambda_k = \zeta_n^k = e^{\theta_k i}$ for $k = 0, 1, 2, \ldots, n-1$, where $i = \sqrt{-1}$ and $\theta_k = \frac{2\pi}{n}k$.

Let $\boldsymbol{w}_k = (w_1, w_2, \ldots, w_n)^T \in \mathbb{R}^n$ be the eigenvector corresponding to ζ_n^k: $M\boldsymbol{w}_k = \zeta_n^k \boldsymbol{w}_k$. From the elements on both sides,

$$(w_n, w_1, w_2, \ldots, w_{n-1})^T = (\zeta_n^k w_1, \zeta_n^k w_2, \ldots, \zeta_n^k w_n)^T, \tag{30}$$

it follows that the eigenvector is $\boldsymbol{w}_k = (\zeta_n^{(n-1)k}, \ldots, \zeta_n^{2k}, \zeta_n^k, 1)^T$.

Then M is diagonalized as $M = W'D'W'^H$ with:

$$D' = \mathrm{diag}(1, \zeta_n, \zeta_n^2, \ldots, \zeta_n^{n-1}), \tag{31}$$

$$W' = \frac{1}{\sqrt{n}} \left(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{n-1} \right), \tag{32}$$

where ^{H} denotes the transpose of a complex conjugate, and W' is the basis of complex DFT.

B. Real block diagonalization of M

Here we give the block diagonalization of M. This can also be found in textbooks on linear algebra [47] and also in [5, 3, 6]. As Park [3] pointed out, this real block diagonalization is not unique, and there are other ways how it can be done.

ζ_n^k and ζ_n^{n-k} (two eigenvalues of M) are complex conjugate to each other. To make the corresponding $\boldsymbol{w}_k$ and $\boldsymbol{w}_{n-k}$ (a pair of complex conjugate eigenvectors) real vectors, divide them into real and imaginary parts: $\boldsymbol{w}_k = \boldsymbol{c}_k + i\boldsymbol{s}_k$ and $\boldsymbol{w}_{n-k} = \boldsymbol{c}_k - i\boldsymbol{s}_k$, where

$$\boldsymbol{c}_k = (\cos(n-1)\theta_k, \cos(n-2)\theta_k, \ldots, \cos\theta_k, 1)^T, \tag{33}$$

$$\boldsymbol{s}_k = (\sin(n-1)\theta_k, \sin(n-2)\theta_k, \ldots, \sin\theta_k, 0)^T. \tag{34}$$

Using $\boldsymbol{c}_k$ and $\boldsymbol{s}_k$, the multiplications of M by vectors $\boldsymbol{w}_k, \boldsymbol{w}_{n-k}$

$$M(\boldsymbol{w}_k, \boldsymbol{w}_{n-k}) = (\boldsymbol{w}_k, \boldsymbol{w}_{n-k}) \begin{pmatrix} \zeta_n^k & 0 \\ 0 & \zeta_n^{n-k} \end{pmatrix}, \tag{35}$$

are rewritten as $M(\boldsymbol{c}_k, \boldsymbol{s}_k) = (\boldsymbol{c}_k, \boldsymbol{s}_k)A_k$, where

$$A_k = \begin{pmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{pmatrix} \in \mathbb{R}^{2\times 2}. \tag{36}$$

Now $M = W'D'W'^H$ is rewritten as a block diagonalization $M = WDW^T$. Here, D is a block diagonal matrix, and W is the basis of real DFT, as follows:

$$D = \begin{cases} \begin{pmatrix} 1 & & & & \\ & A_1 & & & \\ & & A_2 & & \\ & & & \ddots & \\ & & & & A_s \end{pmatrix}, & n \text{ is odd}, \\ \begin{pmatrix} 1 & & & & & \\ & A_1 & & & & \\ & & A_2 & & & \\ & & & \ddots & & \\ & & & & A_s & \\ & & & & & -1 \end{pmatrix}, & n \text{ is even}, \end{cases} \tag{37}$$

$$W = \begin{cases} \sqrt{\frac{2}{n}}\left(\frac{\boldsymbol{w}_0}{\sqrt{2}}, \boldsymbol{c}_1, \boldsymbol{s}_1, \boldsymbol{c}_2, \boldsymbol{s}_2, \ldots, \boldsymbol{c}_s, \boldsymbol{s}_s\right), & n \text{ is odd}, \\ \sqrt{\frac{2}{n}}\left(\frac{\boldsymbol{w}_0}{\sqrt{2}}, \boldsymbol{c}_1, \boldsymbol{s}_1, \boldsymbol{c}_2, \boldsymbol{s}_2, \ldots, \boldsymbol{c}_s, \boldsymbol{s}_s, \frac{\boldsymbol{w}_{\frac{n}{2}}}{\sqrt{2}}\right), & n \text{ is even}, \end{cases} \tag{38}$$

$$s = \begin{cases} \frac{n-1}{2}, & n \text{ is odd}, \\ \frac{n-2}{2}, & n \text{ is even}, \end{cases} \tag{39}$$

where $\boldsymbol{w}_0 = (1, 1, \ldots, 1)^T = \boldsymbol{c}_0$, $\boldsymbol{w}_{\frac{n}{2}} = (-1, 1, -1, 1, \ldots, -1, 1)^T = \boldsymbol{c}_{\frac{n}{2}}$. The orthogonality of W is shown in an appendix available online as a supplemental material.

Note that there is another famous block diagonalization called the Jordan (normal or canonical) form. In our case, M has n different eigenvalues and therefore the Jordan form is no more block diagonal but just a diagonal form. It is the same as Eq. (32).

C. Orthogonality of W

For the sake of completeness, here we show that the columns of W are orthogonal to each other in n dimensional space. Intuitively, the columns $\boldsymbol{c}_k$ form a DCT basis, and $\boldsymbol{s}_k$ form a DST basis, and those are orthogonal bases and also orthogonal to each other. However, since the definitions of $\boldsymbol{c}_k$ and $\boldsymbol{s}_k$ used here differ somewhat from those widely used in the signal processing community, we give proofs for their orthogonality.

First, we show that the following equations hold for $0 < k$:

$$\begin{aligned}
\sum_{j=0}^{n-1} \cos\left(\frac{2\pi j}{n}k\right) &= \frac{1}{2}\sum_{j=0}^{n-1}\left(e^{i\frac{2\pi j}{n}k} + e^{-i\frac{2\pi j}{n}k}\right) \\
&= \frac{1}{2}\left(\frac{1-e^{i\frac{2\pi}{n}kn}}{1-e^{i\frac{2\pi}{n}k}} + \frac{1-e^{-i\frac{2\pi}{n}kn}}{1-e^{-i\frac{2\pi}{n}k}}\right) \\
&= \frac{1}{2}\left(\frac{1-e^{(2\pi k)i}}{1-e^{i\frac{2\pi}{n}k}} + \frac{1-e^{-(2\pi k)i}}{1-e^{-i\frac{2\pi}{n}k}}\right) \\
&= \frac{1}{2}\left(\frac{1-1}{1-e^{i\frac{2\pi}{n}k}} + \frac{1-1}{1-e^{-i\frac{2\pi}{n}k}}\right) = 0, \\
\sum_{j=0}^{n-1} \sin\left(\frac{2\pi j}{n}k\right) &= \frac{1}{2i}\sum_{j=0}^{n-1}\left(e^{i\frac{2\pi j}{n}k} - e^{-i\frac{2\pi j}{n}k}\right) = 0.
\end{aligned}$$

Here we have used the geometric series fomula with a common ratio $e^{\pm i\frac{2\pi}{n}k}$.

Now we show that columns $\boldsymbol{c}_k$ are orthogonal to each other for $0 < k \leq s, 0 < l \leq s$:

$$\begin{aligned}
\boldsymbol{c}_k^T\boldsymbol{c}_l &= \sum_{j=0}^{n-1}\cos(j\theta_k)\cos(j\theta_l) = \sum_{j=0}^{n-1}\cos(\frac{2\pi j}{n}k)\cos(\frac{2\pi j}{n}l) \\
&= 2\sum_{j=0}^{n-1}\left(\cos(\frac{2\pi j}{n}(k+l)) + \cos(\frac{2\pi j}{n}(k-l))\right) = 0, \\
\boldsymbol{c}_k^T\boldsymbol{c}_k &= \sum_{j=0}^{n-1}\cos^2(j\theta_k) = \sum_{j=0}^{n-1}\cos^2(\frac{2\pi j}{n}k) \\
&= \frac{1}{2}\sum_{j=0}^{n-1}\left(1+\cos(2\frac{2\pi j}{n}k)\right) = \frac{1}{2}\sum_{j=0}^{n-1}1 = \frac{n}{2}.
\end{aligned}$$

Similarly, columns $\boldsymbol{s}_k$ are orthogonal to each other for $0 < k \leq s, 0 < l \leq s$:

$$\begin{aligned}
\boldsymbol{s}_k^T \boldsymbol{s}_l &= \sum_{j=0}^{n-1} \sin(j\theta_k)\sin(j\theta_l) = \sum_{j=0}^{n-1} \sin(\frac{2\pi j}{n}k)\sin(\frac{2\pi j}{n}l) \\
&= -2\sum_{j=0}^{n-1}\Big(\cos(\frac{2\pi j}{n}(k+l)) - \cos(\frac{2\pi j}{n}(k-l))\Big) = 0, \\
\boldsymbol{s}_k^T \boldsymbol{s}_k &= \sum_{j=0}^{n-1} \sin^2(j\theta_k) = \sum_{j=0}^{n-1} \sin^2(\frac{2\pi j}{n}k) \\
&= \frac{1}{2}\sum_{j=0}^{n-1}\Big(1 - \cos(2\frac{2\pi j}{n}k)\Big) = \frac{1}{2}\sum_{j=0}^{n-1} 1 = \frac{n}{2}.
\end{aligned}$$

Also, columns $\boldsymbol{c}_k$ and $\boldsymbol{s}_k$ are orthogonal to each other:

$$\begin{aligned}
\boldsymbol{s}_k^T \boldsymbol{c}_l &= \sum_{j=0}^{n-1} \sin(j\theta_k)\cos(j\theta_l) = \sum_{j=0}^{n-1} \sin(\frac{2\pi j}{n}k)\cos(\frac{2\pi j}{n}l) \\
&= 2\sum_{j=0}^{n-1}\Big(\sin(\frac{2\pi j}{n}(k+l)) + \sin(\frac{2\pi j}{n}(k-l))\Big) = 0, \\
\boldsymbol{s}_k^T \boldsymbol{c}_k &= \sum_{j=0}^{n-1} \sin(j\theta_k)\cos(j\theta_k) = \sum_{j=0}^{n-1} \sin(\frac{2\pi j}{n}k)\cos(\frac{2\pi j}{n}k) \\
&= 2\sum_{j=0}^{n-1}\Big(\sin(\frac{2\pi j}{n}(2k)) + \sin(\frac{2\pi j}{n}(0))\Big) = 0.
\end{aligned}$$

Finally, columns $\boldsymbol{w}_0$ and $\boldsymbol{w}_{\frac{n}{2}}$ are orthogonal to the other columns:

$$
\begin{aligned}
\boldsymbol{w}_0^T \boldsymbol{w}_0 &= \boldsymbol{c}_0^T \boldsymbol{c}_0 = \sum_{j=0}^{n-1} 1 = n, \\
\boldsymbol{w}_{\frac{n}{2}}^T \boldsymbol{w}_{\frac{n}{2}} &= \boldsymbol{c}_{\frac{n}{2}}^T \boldsymbol{c}_{\frac{n}{2}} = \sum_{j=0}^{n-1} 1 = n, \\
\boldsymbol{w}_0^T \boldsymbol{c}_k &= \boldsymbol{c}_0^T \boldsymbol{c}_k = \sum_{j=0}^{n-1} \cos\left(\frac{2\pi j}{n} k\right) = 0, \\
\boldsymbol{w}_0^T \boldsymbol{s}_k &= \boldsymbol{c}_0^T \boldsymbol{s}_k = \sum_{j=0}^{n-1} \sin\left(\frac{2\pi j}{n} k\right) = 0, \\
\boldsymbol{w}_{\frac{n}{2}}^T \boldsymbol{c}_k &= \boldsymbol{c}_{\frac{n}{2}}^T \boldsymbol{c}_k = -\sum_{j=0}^{n-1} (-1)^j \cos(\frac{2\pi j}{n} l) = -\frac{1}{2} \sum_{j=0}^{n-1} e^{i\pi j} \left(e^{i\frac{2\pi j}{n}k} + e^{-i\frac{2\pi j}{n}k}\right) \\
&= \frac{1}{2}\left(\frac{1 - e^{i\pi n} e^{i\frac{2\pi}{n}kn}}{1 - e^{i\pi} e^{i\frac{2\pi}{n}k}} + \frac{1 - e^{i\pi n} e^{-i\frac{2\pi}{n}kn}}{1 - e^{i\pi} e^{-i\frac{2\pi}{n}k}}\right) \\
&= \frac{1}{2}\left(\frac{1-1}{1 - e^{i\pi} e^{i\frac{2\pi}{n}k}} + \frac{1-1}{1 - e^{i\pi} e^{-i\frac{2\pi}{n}k}}\right) = 0, \\
\boldsymbol{w}_{\frac{n}{2}}^T \boldsymbol{s}_k &= \boldsymbol{c}_{\frac{n}{2}}^T \boldsymbol{s}_k = -\sum_{j=0}^{n-1} (-1)^j \sin(\frac{2\pi j}{n} l) = -\frac{1}{2i} \sum_{j=0}^{n-1} e^{i\pi j} \left(e^{i\frac{2\pi j}{n}k} - e^{-i\frac{2\pi j}{n}k}\right) = 0.
\end{aligned}
$$

Although $\boldsymbol{c}_k$ and $\boldsymbol{s}_k$ are not normal vectors, defining matrix W with the factor $\sqrt{\frac{2}{n}}$ makes it an orthonormal matrix.