# 3D Object Pose Inference via Kernel Principal Component Analysis with Image Euclidian Distance (IMED)

T. Tangkuampien and D. Suter
Institute for Vision Systems Engineering
Monash University, Australia
{therdsak.tangkuampien,d.suter}@eng.monash.edu.au

**Abstract**

Kernel Principal Component Analysis (KPCA) is a powerful non-linear unsupervised learning technique for high dimensional pattern analysis. KPCA on images, however, usually considers each image pixel as an independent dimension and does not take into account the spatial relationship of nearby pixels. In this paper, we show how the Image Euclidian Distance (IMED), which takes into account local pixel intensities, can efficiently be embedded into KPCA via the Kronecker product and Eigenvector projections, whilst still retaining desirable properties of Euclidian distance (such as kernel positive definitiveness and effective image de-noising). We demonstrate that KPCA with embedded IMED is a more intuitive and accurate technique than standard KPCA through a 3D object pose estimation application.

## 1 Introduction

Kernel techniques such as Kernel Principal Component Analysis (*KPCA*) [7] and Support Vector Machines (*SVM*) have both been shown to be powerful non-linear techniques in the analysis of pixel patterns in images. In supervised *SVM* image classification, images are usually vectorized before embedding (via a kernel function) for discrete classification in high dimensional feature space. In unsupervised *KPCA*, vectorized images are also embedded to a high dimensional feature space, but instead of determining optimal separating hyper planes, linear Principal Component Analysis (*PCA*) is performed. Both *KPCA* and *SVM* take advantage of the *kernel trick* (in order to avoid explicitly mapping input vectors), which involves defining the '*distance*' between two vectors. In both cases, the traditional Euclidian distance is used for embedding. Each pixel, in this case, is usually considered as an independent dimension, and therefore these approaches do not take into account the spatial relationship of nearby pixels on the image plane.

In this paper, we show how the Image Euclidian Distance (*IMED*) [9] (which takes into account spatial pixel relation on the image plane) is a better distance criterion, especially for 3D pose estimation. We begin by summarizing how *IMED* can be embedded into *KPCA* via the use of the Standardizing Transform (*ST*) [9], as well as how this can be implemented efficiently using a combination of the Kronecker product and Eigenvector projections [1]. A major significance of *ST*, is that it can alternatively be viewed as a

BMVC 2006 doi:10.5244/C.20.15

pre-processing transformation (on the pixel intensities), after which traditional Euclidian distance can be applied [9]. Effectively, this means that all desirable properties (such as positive definiteness, non-linear de-noising [6] and pre-image approximation using gradient descent [7]) of using traditional Euclidian distances in *KPCA* still applies. A minor disadvantage of *ST* is that it can be expensive in its memory consumption. For an image of size $M \times N$, the full *ST* matrix needs to be of size $MN \times MN$. Fortunately, for the case of Gaussian kernel embedding, this matrix is separable [9] and can be stored as the Kronecker product [1] of two reduced matrices of size $M \times M$ and $N \times N$.

Our significant contributions of this paper is the demonstration of a practically viable embedding of *IMED* (with Gaussian kernel) into *KPCA*. By separating the originally proposed Standardizing Transform (*ST*) into the Kronecker product of two identical reduced transform [9], we show how *IMED* is a better image '*distance*' criterion than traditional vectorized Euclidian distance. To support this, we present results using *IMED* embedded *KPCA* for 3D object pose estimation (compared to [3]), using kernel subspace mapping (*KSM*) [8]. We show how the accuracy of IMED embedded *KPCA* (for 3D pose estimation) is more accurate than its traditional Euclidian embedded counterpart and other pose estimation techniques [2, 3, 8, 10].

## 2 Related & Previous Work

The problem of 3D object pose estimation using machine learning can be summarized as follows: Given images of the object, from different known orientations, as the training set: how do we optimally learn a mapping to accurately determine the orientation of unseen images of the same object from a 'static' camera. Alternatively, instead of calculating the object's orientation, we could determine the orientation and position of a moving camera that the 'static' object is viewed from. It is also usual that the unseen input images are corrupted by noise, in which case, image de-nosing techniques, such as *KPCA*, can be applied. Zhao *et al* [10] used *KPCA* with a neural network architecture to determine pose of objects form the Columbia Object Image Library (*COIL-20*) database [2]. In that case, however, the pose is only restricted to rotations about a single vertical axis. In this paper, we consider the more complex pose estimation problem, that of determining the camera position where an object is viewed from anywhere on the upper hemisphere of an object's viewing sphere.

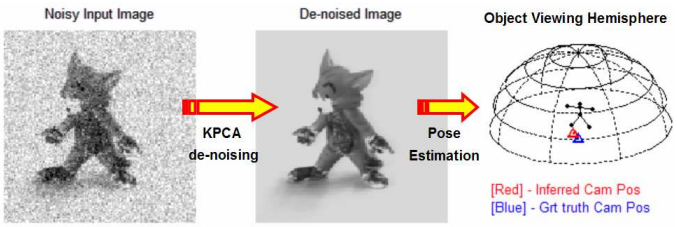


Figure 1: Diagram to summarize the pose estimation problem of an object viewed from the upper viewing hemisphere.

Peters *et al* [3] introduced the notion of *view-bubbles*, which are area views where the object remains visually the same (up to within a pre-defined criterion). The problem of hemispherical object pose estimation can then be re-formulated as the selection of the correct view bubble and interpolating from within the bubble. A disadvantage of this technique, however, is the need for a large and well-sampled training set (up to 2500 images in [3]), to select the view bubble training images from. In section 5, we present comparable results using only 30 randomly selected training images of the same object.

# 3 Image Euclidian Distance (IMED)

We now summarize the Image Euclidian Distance as introduced by Wang *et al* [9]. We begin by vectorizing each $M \times N$ input image to a vector $\boldsymbol{x}$ where $\boldsymbol{x} \in R^{MN}$. The intensity at the $(m,n)$ pixel is then represented as the $(mN+n)$th dimension in $\boldsymbol{x}$. The standard vectorized Euclidian distance $d_E(\boldsymbol{x},\boldsymbol{y})$ between vectorized images $\boldsymbol{x}$ and $\boldsymbol{y}$ is then defined as

$$d_E^2(\boldsymbol{x},\boldsymbol{y}) = \sum_{k=1}^{MN} (x^k - y^k)^2 = (\boldsymbol{x}-\boldsymbol{y})^\mathsf{T}(\boldsymbol{x}-\boldsymbol{y}). \tag{1}$$

Alternatively, the *IMED* between images, introduces the notion of a metric matrix $G$ of size $MN \times MN$, where the element $g_{ij}$ represents how the dimension $\boldsymbol{x}^i$ affects the dimension $\boldsymbol{x}^j$. To avoid confusion, it is important to note that $i,j,k \in [1,MN]$, whereas $m \in [1,M]$ and $n \in [1,N]$. Provided that the metric matrix $G$ is known, the Image Euclidian Distance can then be calculated as

$$d_{IM}^2(\boldsymbol{x},\boldsymbol{y}) = \sum_{i,j=1}^{MN} g_{ij}(\boldsymbol{x}^i - \boldsymbol{y}^i)(\boldsymbol{x}^j - \boldsymbol{y}^j) = (\boldsymbol{x}-\boldsymbol{y})^\mathsf{T} G (\boldsymbol{x}-\boldsymbol{y}). \tag{2}$$

The metric matrix $G$, therefore solely defines how the *IMED* deviates from the standard Euclidian distance (which is equivalent to replacing $G$ with the identity matrix). The main constraints for *IMED* [9] are that the element $g_{ij}$ is dependent on the pixel distance between pixels $P_i$ and $P_j$, that is $g_{ij} = f(|P_i - P_j|)$, and that $g_{ij}$ monotonically decreases as $|P_i - P_j|$ increases. A constraint on $f$ is that it must be a continuous positive definite function, thereby ensuring that $G$ is positive definite and excluding the tradition Euclidian distance (which is not continuous along the image plane) as a subset of *IMED*. Note that in the present form, the calculation of *IMED* is not memory efficient, due to the need to store the metric $G$, which is of size $MN \times MN$. We show how to improve on this in section 3.2, but first, we need to examine the Standardizing Transform (*ST*)[9], which allows *IMED* to be embedded into more powerful learning algorithms such as *KPCA* and *SVM*s.

## 3.1 Standardizing Transform (ST)

As suggested in [9], the calculation of *IMED* can be simplified by decomposing $G$ to $A^\mathsf{T}A$, leading to

$$d_{IM}^2(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^\mathsf{T} A^\mathsf{T} A (\boldsymbol{x}-\boldsymbol{y}) = (\boldsymbol{u}-\boldsymbol{v})^\mathsf{T}(\boldsymbol{u}-\boldsymbol{v}), \tag{3}$$

where $\boldsymbol{u} = A\boldsymbol{x}$ and $\boldsymbol{v} = A\boldsymbol{y}$. The Standardizing Transform (*ST*) is then merely a special case of $A$, where $G = A^\mathsf{T}A = G^{\frac{1}{2}}G^{\frac{1}{2}}$. This reveals symmetric, positive definite and unique solutions for both $G$ and $G^{\frac{1}{2}}$, where $G = \Gamma\Lambda\Gamma^\mathsf{T}$, and $G^{\frac{1}{2}} = \Gamma\Lambda^{\frac{1}{2}}\Gamma^\mathsf{T}$. In this case, $\Gamma$ is

the orthogonal column matrix of the Eigenvectors of $G$, and $\Lambda$ the diagonal matrix of the corresponding Eigenvalues. The Standardizing Transform (*ST*) [9] is then the transformation $G^{\frac{1}{2}}(\bullet)$. In order to embed *IMED* into other powerful learning algorithms based on standard Euclidian distance, we simply apply the transformation $G^{\frac{1}{2}}(\bullet)$ to the vectorized images, to let's say $\boldsymbol{x}$ and $\boldsymbol{y}$, and obtain $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. From (3), the *IMED* is then simply the traditional Euclidian distance between the transformed vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.

## 3.2 Kronecker Product IMED

Up until now, we have merely defined *IMED* and its corresponding Standardizing Transform $G^{\frac{1}{2}}(\bullet)$, but not how to construct the matrix $G$ itself. We now reveal how to do so efficiently using the Kronecker Product and concentrating specifically on the Gaussian function, where

$$g_{ij} = \frac{1}{2\pi\sigma^2}\exp^{\{\frac{-|P_i-P_j|^2}{2\sigma^2}\}}. \tag{4}$$

Note that in [9], the use of Kronecker product on Gaussian function was only briefly mentioned, but neither detailed explanation nor proof was provided. In (4), the free variable $\sigma$ controls the spread of the Gaussian signal, which acts as a bandlimited filter. For large values of $\sigma$, the influence of, let's say $P_i$ to its neighboring pixel $P_j$ on the image plane, is more significant (compared to when $\sigma$ is small). The Standardizing Transform, in this case, acts like a low pass filter, since the signal induced at each pixel (after applying the Standardizing Transform) is smoother and flatter. As $\sigma$ decreases, the Gaussian signal induced becomes steeper and thinner, reducing its influence on neighboring pixels, and requiring higher frequency signals for reconstruction in the Fourier domain. For the limit, where $\sigma\rightarrow 0$, we induce the dirac delta signal which requires infinite bandwidth, leading to the traditional Euclidian distance.

By letting $P_i$ and $P_j$ be the pixels at location $(m^i,n^i)$ and $(m^j,n^j)$ respectively, the corresponding squared pixel distance between the two pixels on the image plane is then given by $|P_i - P_j|^2 = (m^i - m^j)^2 + (n^i - n^j)^2$. Substituting this into (4), we obtain the separable, and therefore reducible metric representation

$$g_{ij} = \frac{1}{2\pi\sigma^2}[\exp^{\{\frac{-[(m^i-m^j)^2+(n^i-n^j)^2]}{2\sigma^2}\}}] = \frac{1}{2\pi\sigma^2}[\exp^{\{\frac{-(m^i-m^j)^2}{2\sigma^2}\}}\cdot\exp^{\{\frac{-(n^i-n^j)^2}{2\sigma^2}\}}]. \tag{5}$$

As previously mentioned, $m \in [1,M]$ and $n \in [1,N]$, leading to a re-formulation of (5) as the Kronecker product [1] of two smaller matrices $\Psi^M$ and $\Psi^N$ of size $M\times M$ and $N\times N$ respectively, where

$$\Psi^M(m^i,m^j) = \frac{1}{2\pi\sigma^2}\exp^{\{\frac{-(m^i-m^j)^2}{2\sigma^2}\}}, \quad \text{and} \quad \Psi^N(n^i,n^j) = \frac{1}{2\pi\sigma^2}\exp^{\{\frac{-(n^i-n^j)^2}{2\sigma^2}\}}. \tag{6}$$

This leads to a simplified and memory efficient version of the metric matrix, where $G = \Psi^M \otimes \Psi^N$. For the case of squared images, when $M{=}N$, $\Psi^M$ and $\Psi^N$ are identical and only one matrix copy of size $M\times N$ is required. For the remainder of this paper, we will consider only square images (as this can effective half the current memory consumption), and refer to the identical reduced matrix as $\Psi$.

We can now decompose $\Psi$ into its corresponding matrix of Eigenvectors $\Omega$ and diagonal Eigenvalues $\Theta$, where $\Psi = \Omega\Theta\Omega^{\mathsf{T}}$. Using well established compatible properties of

the Kronecker product with standard matrix multiplication [1], the metric matrix becomes

$$G = (\Omega\Theta\Omega^{\mathsf{T}}) \otimes (\Omega\Theta\Omega^{\mathsf{T}}) = (\Omega \otimes \Omega)(\Theta \otimes \Theta)(\Omega \otimes \Omega)^{\mathsf{T}} = \Gamma\Lambda\Gamma^{\mathsf{T}} \tag{7}$$

From this, we can derive the Standardizing Transform as $G^{\frac{1}{2}} = \Gamma\Lambda^{\frac{1}{2}}\Gamma^{\mathsf{T}}$. Furthermore, the reduced Eigenvalue matrix $\Theta$ is diagonal and can be vectorized as $\lambda$, where $\lambda_m$ represents $\Theta(m,m)$. The corresponding Kronecker product of the Eigenvalue matrices $(\Theta \otimes \Theta)^{\frac{1}{2}}$ can efficiently be represented as the outer product $(\Theta \otimes \Theta)^{\frac{1}{2}} = (R[\lambda\lambda^{\mathsf{T}}])^{\frac{1}{2}}$, where $R$ is the vectorization function followed by the diagonalization function.

On the other hand, if the image resolution is low, and speed is a factor, it is possible to construct the full $G^{\frac{1}{2}}$ matrix using a single Kronecker product. From (7), we can derive the following:

$$G^{\frac{1}{2}} = (\Omega \otimes \Omega)(\Theta \otimes \Theta)^{\frac{1}{2}}(\Omega \otimes \Omega)^{\mathsf{T}} = \Omega\Theta^{\frac{1}{2}}\Omega^{\mathsf{T}} \otimes \Omega\Theta^{\frac{1}{2}}\Omega^{\mathsf{T}} = \Psi^{\frac{1}{2}} \otimes \Psi^{\frac{1}{2}}, \tag{8}$$

with the constraints: $\Psi = (\Psi^{\frac{1}{2}})^{\mathsf{T}}\Psi^{\frac{1}{2}} = \Psi^{\frac{1}{2}}\Psi^{\frac{1}{2}}$. Visually, as mentioned in [9], $G^{\frac{1}{2}}$ is a domain smoothing, where the Eigenvectors with the higher Eigenvalues, represent the low frequency basis signals. Referring to figure 2 and equation 4, the larger the value of $\sigma$, the smoother the Standardizing Transform image will be, and therefore, the higher the Eigenvalues corresponding to the low frequency basis. The sharpest that the image can ever be (including noise) after the Standardizing Transformation is therefore when $\sigma \rightarrow 0$, where we are left with the original image itself.

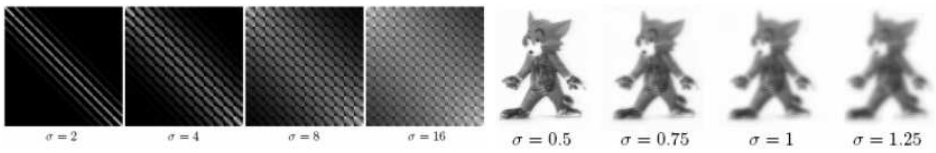


Figure 2: (Left) Images of the Standardizing Transform $G^{\frac{1}{2}}$ with different $\sigma$ values. Note how the images tends to the diagonal matrix as $\sigma \rightarrow 0$. (Right) Test images after applying the Standardizing Transform $G^{\frac{1}{2}}$ with different $\sigma$ values.

# 4 3D Pose Inference with Kernel Principal Components

We are interested in embedding *IMED* into unsupervised Kernel Principal Component Analysis (*KPCA*) [6, 7]. We begin by summarizing *KPCA* before showing how to apply *IMED* embedded *KPCA* in Kernel Subspace Mapping (*KSM*) [8]. *KSM* will be used in the 3D pose estimation problem reviewed in section 2.

## 4.1 Kernel PCA overview

In standard *KPCA* [6, 7] each vectorized image $\boldsymbol{x}_i$ (in the set of training images $X$) is non-linearly mapped via a semi-positive definite kernel function $k_v$. The kernel defines the non-linear relationship between the two vectorized images, where

$$k_v(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) \rangle, \;\; \Phi : X \rightarrow H \tag{9}$$

is a dot product in feature space $H$. When supplied with a vectorized image training set $X$ with $N$ exemplars, *KPCA* maps each vector $\boldsymbol{x}_1,...,\boldsymbol{x}_N$ in the set to a feature space as $\Phi(\boldsymbol{x}_1),...,\Phi(\boldsymbol{x}_N)$ and preforms linear *PCA* on the mapped vectors. Schölkopf *et al* [6] showed that the problem of finding the coefficients $\alpha$ of the principal components in feature space can be reduced to the diagonalization of the centered Kernel matrix $K_c$,

$$N\lambda\alpha = K_c\alpha, \tag{10}$$

where $K_c = (I - ee^{\mathsf{T}})K(I - ee^{\mathsf{T}})$, with $K_{ij} = k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $e = N^{-1/2}(1,...,1)^{\mathsf{T}}$. The *KPCA* projection (onto the $k$th principal axis) of a novel vectorized image $\boldsymbol{x}$ can be expressed implicitly via the *kernel trick* as

$$\langle V_v^k \cdot \Phi(\boldsymbol{x})\rangle = \sum_{i=1}^{N} \alpha_i^k \langle \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x})\rangle = \sum_{i=1}^{N} \alpha_i^k k_v(\boldsymbol{x}_i,\boldsymbol{x}). \tag{11}$$

For the problem of image de-noising [6] and 3D object pose estimation, we use the radial basis function (*RBF*) kernel, where

$$k_v(\boldsymbol{x}_i,\boldsymbol{x}) = \exp^{-\gamma\{(\boldsymbol{x}_i-\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{x}_i-\boldsymbol{x})\}}. \tag{12}$$

The *RBF* kernel was selected because the pre-image can be approximated using well established techniques such as the fixed-point algorithm [7]. Note that the Gaussian (*RBF*) kernel in (12) is different to the Gaussian used in the metric matrix $G$ in (4). In this case, the former encodes similarities between preprocessed vectors in the training set, whereas the latter encodes similarity between pixel intensities along the image plane. For *KPCA* de-noising [6] using the *RBF* kernel in (12), there are two free parameters to tune, these being $\gamma$ the Euclidian distance scale factor, and $\eta$ the number of principal axis projections to retain in the feature space. We tune these parameters using cross-validation to minimize the pre-image reconstruction cost function $C = \frac{1}{N}\sum_{i=1}^{N} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^p \parallel^2$, where $\boldsymbol{x}_i^p$ is the pre-image of $\boldsymbol{x}_i$ projected onto the first $\eta$ principal axis via *KPCA*.

In order to embed *IMED* into *KPCA*, we first apply the Standardization Transform $G^{\frac{1}{2}}$ to the vectorized training set $X$, to get standardized training set $U$, where $U = G^{\frac{1}{2}}X$. *KPCA* is then performed using the *IMED* embedded *RBF* kernel, which is defined as follows:

$$k_{im}(\boldsymbol{x}_i,\boldsymbol{x}_j) = \exp^{-\gamma\{(\boldsymbol{x}_i-\boldsymbol{x}_j)^{\mathsf{T}}G(\boldsymbol{x}_i-\boldsymbol{x}_j)\}} = \exp^{-\gamma\{(\boldsymbol{u}_i-\boldsymbol{u}_j)^{\mathsf{T}}(\boldsymbol{u}_i-\boldsymbol{u}_j)\}} = k_v(\boldsymbol{u}_i,\boldsymbol{u}_j). \tag{13}$$

This leads to the standard *RBF* kernel, as in (12), thereby ensuring that all desirable properties of the standard *RBF* kernel, such as positive definitiveness and gradient descent pre-image approximations [7], are valid. The free parameters are then tuned on $U$ and the *IMED* embedded *KPCA* projection becomes

$$\langle V_{im}^k \cdot \Phi(\boldsymbol{x})\rangle = \sum_{i=1}^{N} \alpha_i^k k_{im}(\boldsymbol{x}_i,\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i^k k_v(\boldsymbol{u}_i,\boldsymbol{u}). \tag{14}$$

## 4.2 IMED embedded Kernel Subspace Mapping

Kernel Subspace Mapping (*KSM*) [8] can be used to map data between two high dimensional correlated spaces. *KPCA* is initialized to learn the subspaces of the high dimensional data, for let's say the input training set $X$ and the output training set $Y$, and *LLE*

weight reconstruction [5] is used to map between the two subspaces. Given novel inputs, the data is projected through the two subspaces and the pre-image calculated to determine the input's representation in the output space. If any of the training vectors are derived from images, then *IMED* embedded *KPCA* can be applied. For generality, we will consider both *IMED* embedded *KPCA* (input - figure 3: left) and standard KPCA (output - figure 3: right), where the input set $X$ is constructed from images of a 3D object, and the output set $Y$ are three dimensional camera positions on the upper hemisphere of the object's viewing sphere. Note that this is a simple case (as the output space is only 3 dimensional) for *KSM*, which can map to higher dimensional output spaces as in markerless human motion capture [8].

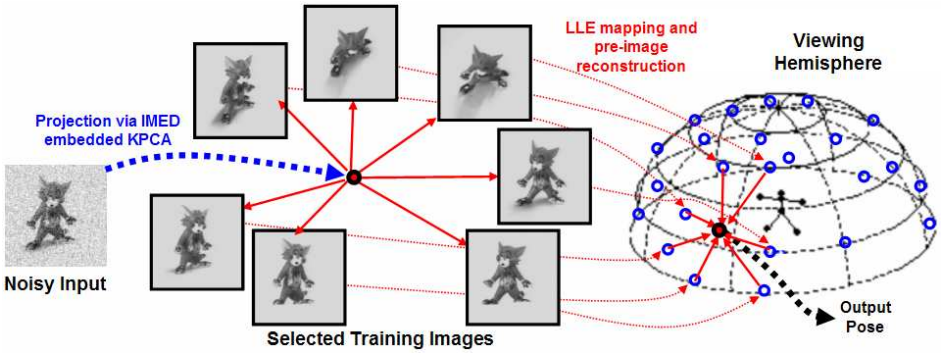


Figure 3: Diagram to Summarize Kernel Subspace Mapping [8] for 3D object pose estimation. Selected members of the image training set $X$ are highlighted with black bounding boxes, whereas the output training set $Y$ are represented as blue circles on the viewing hemisphere.

We initialize *KSM* by learning the subspace representation of $X$ and $Y$ using *IMED* embedded *KPCA* and standard (*RBF* kernel) *KPCA* respectively. From these, we can obtain the *KPCA* projected training sets $V_{im}^X$ and $V_v^Y$, where the corresponding instance in the sets have the following forms:

$$\begin{aligned} \boldsymbol{v}_i^X &= [\langle V_{im}^1 \cdot \Phi(\boldsymbol{x}_i)\rangle, .., \langle V_{im}^\eta \cdot \Phi(\boldsymbol{x}_i)\rangle]^\top, \ \forall\, \boldsymbol{v}^X \in R^\eta \\ \boldsymbol{v}_i^Y &= [\langle V_v^1 \cdot \Phi(\boldsymbol{y}_i)\rangle, ....., \langle V_v^\psi \cdot \Phi(\boldsymbol{y}_i)\rangle]^\top, \ \forall\, \boldsymbol{v}^Y \in R^\psi, \end{aligned} \tag{15}$$

where $\eta$ and $\psi$ are the training set's corresponding tuned projected feature dimensions (using the cross validated pre-image cost function). During run time, given a new unseen vectorized image $\boldsymbol{x}$ of the same 3D object, we can project it using *IMED* embedded *KPCA* to obtain $\boldsymbol{v}^X$. The projected vector is then mapped between the two feature spaces using *LLE* neighborhood reconstruction weight $\beta$, which is calculated by minimizing the reconstruction cost function $\varepsilon(\boldsymbol{v}^X) = \|\boldsymbol{v}^X - \sum_{i\in I} \beta_i \boldsymbol{v}_i^X\|^2$, where $I$ is the set of neighborhood indices of the projected vector $\boldsymbol{v}^X$ in the tuned feature space (as shown in [5],[8]). This leads to the mapping $\boldsymbol{v}^Y = \sum_{i\in I} \beta_i \boldsymbol{v}_i^Y$, from which we can determine the corresponding output hemispherical position $\boldsymbol{y}^p$, which is the pre-image [7] of $\boldsymbol{v}^Y$.

# 5 Experiments and Results

For 3D object pose estimation using *KSM*, we begin by selecting training images of the object 'Tom' (figure 4:left) from the data set described in [4]. From the original set of 2500 images, we randomly select sets of 30, 60 and 145 training images. The training size of 30 and 145 were chosen to match the experiments in [3]. Note that in [3] - figure 1, the parameters are given in terms of the 'tracking threshold' and their corresponding view bubbles, where each bubble consists of 5 training images. For view bubble counts of 6 and 29, these correspond to 30 and 145 training images respectively. A test set of 600 'unseen' images are then filtered from the remaining images at regular intervals, such as to maximize the distribution around the viewing hemisphere. To test the efficacy of *IMED* embedded *KPCA* on a more symmetrical object, we use the object 'Dwarf' (figure 4:right) from [4], which was also used in [3]. All experiments are performed in exactly the same was as before, and results summarized in Table 1. For the object 'Dwarf', we retain the training size of 30, 60 and 145 for comparison with the Object 'Tom'.

Table 1: 3D pose inference comparison using the mean angular error for 600 'unseen' views of the object **'Tom'** and object **'Dwarf'**.

| | **'Tom'** | | | **'Dwarf'** | | |
|---|---|---|---|---|---|---|
| **Training size** | **30** | **60** | **145** | **30** | **60** | **145** |
| **Clean data set** | | | | | | |
| Euclidian distance *KPCA* | 6.68° | 2.97° | 1.19° | 8.27° | 2.98° | 1.15° |
| *IMED* embedded *KPCA* | 3.38° | 1.92° | 0.76° | 3.99° | 1.67° | 0.81° |
| **Gaussian noise set** | | | | | | |
| Euclidian distance *KPCA* | 8.63° | 5.88° | 3.01° | 12.01° | 5.22° | 2.60° |
| *IMED* embedded *KPCA* | 3.44° | 2.05° | 0.89° | 4.04° | 1.73° | 0.87° |
| **Salt/Pepper noisy set** | | | | | | |
| Euclidian distance *KPCA* | 10.68° | 9.40° | 4.56° | 16.51° | 8.33° | 5.07° |
| *IMED* embedded *KPCA* | 4.58° | 2.40° | 1.34° | 4.46° | 1.90° | 1.07° |

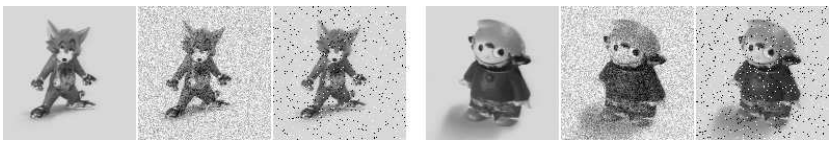

Figure 4: Selected images of the object **'Tom'** (left) and object **'Tom'** (right) used in the 3D pose estimation problem. We use the original clean image ($1^{st}$ & $4^{th}$ image), image corrupted with Gaussian noise ($2^{nd}$ & $5^{th}$ image) and Salt/Pepper noise ($3^{rd}$ & $6^{th}$ image).

We test our technique with clean images as well as with images corrupted with zero mean Gaussian noise (variance of 0.01) and salt/pepper noise (noise density of 0.05) as shown in figure 4. Table 1 shows that in all cases, *IMED* embedded *KSM* provides the most accurate pose estimation from both clean and noisy images (compared to when traditional

Euclidian is applied). Furthermore, it is also the most robust to noise and shows the least percentage increase in error for both Gaussian and salt/pepper noise. As expected, with any exemplar based learning algorithm, the mean error gradually decreases as more training images are included in the set.

For completeness, we also present another set of pose inference results for the same tuned parameters as in Table 1. In this case, however, we select 400 test images by randomly selecting from the original hemispherical data set (2500 images) and not constraining the selected images to be novel (*i.e.* the test images may also include images that were used in training). We do this because we believe this gives a more realistic representation of the input data in industrial applications, where there is no exclusion of the training set, let alone any idea of which images were used in training. As expected, *IMED* embedding performs even better in this case (Table 2), as *KSM* will project any training images to the correct position in the feature space, and this has zero error to begin with. We also include results from [3] for comparison in Table 2 even though only 30 test images were used here. Note that we have not considered the test images in [3] to be novel . This is because a sparse representation of the object is built from the original set of 2500 images using a greedy approach, which is then used for pose estimation. Since the 30 test images are also derived from the original 2500 images, they cannot be considered novel.

Table 2: 3D pose inference comparison using the mean angular error for 400 randomly selected views of the object **'Tom'** and object **'Dwarf'**.

| | **'Tom'** | | | **'Dwarf'** | | |
|---|---|---|---|---|---|---|
| **Training size** | **30** | **60** | **145** | **30** | **60** | **145** |
| **Clean data set** | | | | | | |
| View Bubble Method [3] | 36.51° | | 0.77° | | | |
| Euclidian distance *KPCA* | 4.44° | 2.32° | 0.84° | 9.40° | 3.14° | 1.23° |
| *IMED* embedded *KPCA* | 3.22° | 1.15° | 0.59° | 3.26° | 1.49° | 0.68° |
| **Gaussian noise set** | | | | | | |
| Euclidian distance *KPCA* | 6.28° | 5.13° | 2.72° | 12.72° | 5.60° | 2.46° |
| *IMED* embedded *KPCA* | 3.29° | 1.26° | 0.69° | 3.56° | 1.55° | 0.75° |
| **Salt/Pepper noisy set** | | | | | | |
| Euclidian distance *KPCA* | 9.47° | 8.64° | 4.38° | 14.64° | 8.57° | 4.92° |
| *IMED* embedded *KPCA* | 4.51° | 1.75° | 1.04° | 4.80° | 1.83° | 1.05° |

It must be noted that the accuracy of the pose inference is dependent on the complexity of the object. Clearly, it is not possible to infer the pose using a sphere of uniform colour. For cross comparison with [3] for object 'Dwarf', we are able to achieve more accurate results for a training set of only 30 images (3.26°) as compared to using the 'view bubble' set of 130 images (4.2°).

# 6 Discussions and Concluding Remarks

We have presented a practical solution that allows the efficient embedding of the Image Euclidian Distance (*IMED*) into non-linear Kernel Principal Component Analysis

(*KPCA*). We show that *IMED* gives better image distance relationships (than traditional vectorized Euclidian distance), especially when applied to the problem of 3D object pose inference. Using our technique based on *IMED*, we can infer pose with mean errors of less than $3.5^{\circ}$, whilst using only a training set of 30 images. This is a quite accurate, considering that fact that a human would not be able to achieve the same level of accuracy. Our technique is also robust to noise (especially Gaussian noise) and, in such a case, only shows minor percentage increase in mean angular error. Furthermore, we achieved this without explicit knowledge of the full training set of size 2500 images as is the case with the training set used in [3]. This is because we do not use a greedy approach to filter out (from the original set of 2500 images) training data, in order to build a sparse representation; but we simply randomly selected training images form it. The constraint, in this case, being that the training set is relatively evenly spread over the probability distribution of the data. Our test images are also novel (Table 1) and are regularly spread over the entire hemisphere, whereas in [3], only 30 test images were used from three different patches of the viewing hemisphere.

# References


[1] Laub A. J. Matrix analysis for scientists and engineers. pages 139–150, 2005.

[2] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-20). In *http://www.cs.columbia.edu/CAVE/*, 1996.

[3] G. Peters. Efficient pose estimation using view-based object representations. In *Machine Vision and Applications*, volume 16, pages 59–63, 2004.

[4] G. Peters, B. Zitova, and C. von der Malsburg. How to measure the pose robustness of object views. In *Image and Vision Computing*, volume 20, pages 249–256, 2002.

[5] L. K. Saul and S. T. Roweis. Think globally, fit locally:unsupervised learning of low dimensional manifolds. In *Journal of Machine Learning Research*, volume 4, pages 119–155, 2003.

[6] B. Schölkopf, S. Mika, A.J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature fpaces. In *Advances in Neural Information Processing Systems*, volume 11, pages 536–542, 1999.

[7] B. Schölkopf, S. Mika, A.J. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction via approximate pre-images. In *Int. Conf. on Artificial Neural Networks*, pages 147–152, 1998.

[8] T. Tangkuampien and D. Suter. Real-time human pose inference using kernel principal component pre-image approximations. In *British Machine Vision Conference, Edinburgh, UK*, 2006.

[9] L. Wang, Y.Zhang, and J. Feng. On the euclidian distance of images. In *IEEE trans. Patteren Analysis and Machine Intelligence*, volume 27, pages 1334–1339, 2005.

[10] L-W Zhao, S-W Luo, and L-Z Liao. 3D object recognition and pose estimation using kernel PCA. In *Int. Conf. on Machine learning & Cybernetics*, pages 3258–3262, 2004.