Noname manuscript No.
(will be inserted by the editor)

Lie-X: Depth Image Based Articulated Object Pose Estimation, Tracking, and Action Recognition on Lie Groups

Chi Xu · Lakshmi Narasimhan Govindarajan · Yu Zhang · Li Cheng

Received: date / Accepted: date

Abstract Pose estimation, tracking, and action recognition of articulated objects from depth images are important and challenging problems, which are normally considered separately. In this paper, a unified paradigm based on Lie group theory is proposed, which enables us to collectively address these related problems. Our approach is also applicable to a wide range of articulated objects. Empirically it is evaluated on lab animals including mouse and fish, as well as on human hand. On these applications, it is shown to deliver competitive results compared to the state-of-the-arts, and non-trivial baselines including convolutional neural networks and regression forest methods. Moreover, new sets of annotated depth data of articulated objects are created which, together with our code, are made publicly available.

Keywords Depth Images, Pose Estimation, Fish, Mouse, Human Hand, Lie Group

1 Introduction

With 3D cameras becoming increasingly ubiquitous in the recent years, there has been growing interest in utilizing depth images for key problems involving articulated objects (e.g. human full-body and hand) such as pose estimation [35, 42, 46, 47, 49, 56, 58], tracking [7, 11, 23, 36, 40], and action recognition [15, 29, 41, 52]. On the other hand, although they are closely related, most existing research efforts targeting these problems in literature are based on diverse and possibly disconnected principles. Moreover, existing algorithms typically focus on a unique type of articulated objects, such as human full-body, or human hand. This leads us to consider in this paper a principled approach to address these related problems across object categories, in a consistent and sensible manner.

Our approach possesses the following contributions: (1) A unified Lie group-based paradigm is proposed to address the problems of pose estimation, tracking, and action recognition of articulated objects from depth images. As illustrated in Fig. 1, a 3D pose of an articulate object corresponds to a point in the underlying pose manifold, a long-time track of its 3D poses amounts to a long curve in the same manifold, whilst an action is represented as a certain curve segment. Therefore, given a depth image input, pose estimation corresponds to inferring the optimal point in the manifold; Action recognition amounts to classifying a curve segment in the same manifold as a particular action type; Meanwhile for the tracking problem, Brownian motion on Lie groups is employed as the generator to produce pose candidates as particles. This paradigm is applicable to a diverse range of articulated objects, and for this reason it is referred to as *Lie-X*. (2) Learning based techniques are incorporated instead of the traditional Jacobian matrices for solving the incurred inverse kinematics problem, namely, presented with visual discrepancies of current results, how to improve on skeletal estimation results. More specifically, an iterative sequential learning pipeline is proposed: multiple initial poses are engaged simultaneously to account for possible location, orientation, and size variations, with each producing its corresponding estimated pose.

Chi Xu, Lakshmi Narasimhan Govindarajan, Yu Zhang
Bioinformatics Institute, A*STAR, Singapore
E-mail: xuchi,lakshming,zhangyu@bii.a-star.edu.sg

Li Cheng (Corresponding author)
Bioinformatics Institute, A*STAR, Singapore & School of Computing, National University of Singapore, Singapore
E-mail: chengli@bii.a-star.edu.sg

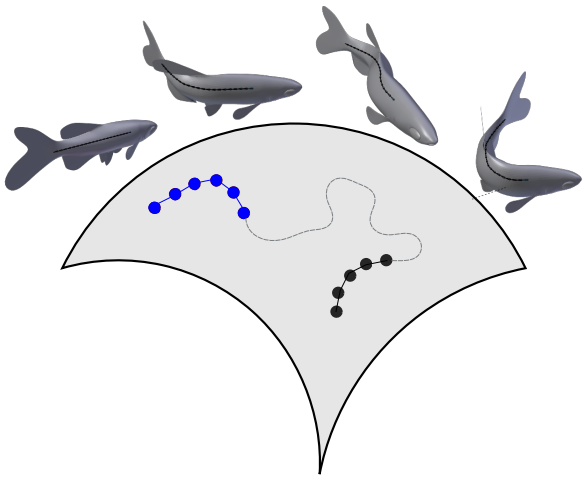


Fig. 1: A cartoon illustration of our main idea: An articulated object can be considered as a point in certain manifold. Its 3D pose, a long track of its 3D motion, and its action sequences (color-coded herein) correspond to a point, a curve, and curve segments in the underlying manifold respectively.

They then pass through a learned scoring metric to deliver the final estimated pose. Note the purpose of the learned metric in our approach is to mimic the behavior of the practical evaluation metric. (3) Empirically our approach has demonstrated competitive performance on fish, mouse, and human hand from different imaging modalities, where it is also specifically referred to as e.g. Lie-fish, Lie-mouse, Lie-hand, respectively. The runtime speed of our pose estimation system is more than real-time — it executes at around 83-267 FPS (frame per second) on a desktop computer without resorting to GPUs. Moreover, new sets of annotated depth images and videos of articulated objects are created. It is worth noting that the depth imaging devices considered in our empirical context are also diverse, including structured illumination and light field technologies, among others. These datasets and our code are to be made publicly available in support of the open-source research activities. [1].

2 Related Work

The recent introduction of commodity depth cameras has led to significant progress in analyzing articulated objects, especially human full-body and hand. In terms of pose estimation, Microsoft Kinect is already widely used in practice at the scale of human full-body, while it is still a research topic at human hand scale [34, 35, 46, 47, 49, 56, 58], partly due to the dexterous nature of hand articulations. [49] is among the first to develop a dedicated convolutional neural net (CNN) method for hand pose estimation, which is followed by [34]. [35] also utilizes deep learning in a synthesizing-estimation feedback loop. [58] further considers to incorporate geometry information in hand modelling by embedding a non-linear generative process within a deep learning framework. [56] studies and evaluates a theoretically motivated random forest method for hand pose estimation. A hierarchical sampling optimization procedure is adopted by [47] to minimize the error-induced energy functions, where a similar energy function is optimized via efficient gradient-based techniques in [46] for personalizing hand shapes to individual users. [43] instead casts hand pose estimation as a matrix completion problem with deeply learned features. Meanwhile, various tracking methods have been developed for full-body [23] and hand [7, 36, 40]. A particle swarm optimization (PSO) scheme is utilized in [36] to recover temporal hand poses by stochastically seeking solution to the induced minimization problem. A hybrid method is adopted in [40] that combines PSO further with the widely-used iterated closest point technique. [7] considers learning salient points on fingers, for which an objective function is introduced to jointly take into account of edges, flow and collision cues. [23] describes a tracking-by-detection [4] type method based on 3D volumetric representation. 3D action recognition has also drawn great amount of attentions lately [29, 33, 41, 52]. For example, [29] tackles action recognition using variants of recurrent neural nets. [41] considers a mapping to a view-invariant high-level space by CNNs and Fourier temporal pyramid. Moreover, the work of [33] discusses a method to jointly train models for human full-body pose estimation and action recognition using spatial temporal and-or graphs. On the other hand, it is a much harder problem when a color camera is used instead of a depth camera, such as [1], where pose estimation is formulated as a regression problem that is subsequently addressed by relevance vector machine and support vector machine. Now, let us look at the other two articulated objects to be described in this paper, i.e. fish and mouse. They are relatively simple in nature but are less studied. Existing literature [11, 15, 16] are mostly 2D-based, and the focus is mainly on pose estimation. [53] is a very recent work in analyzing group-level behavior of lab mice that relies on a simplified straight-line representation of a mouse skeleton. We also would like to point out that there are research efforts across object categories: [16] estimates poses of

[1] Our datasets, code, and detailed information pertaining to the project can be found at a dedicated project webpage `http://web.bii.a-star.edu.sg/~xuchi/Lie-X.html`.

zebrafish, lab mouse, and human face; Meanwhile there are also works that deal with more than one problem, such as [33]. They have achieved very promising results as discussed previously. Our work may be considered as a renewed attempt to address related problems and work with a broad range of articulated objects under one unified principle. For more detailed overview of related works, interested readers may consult to the recent surveys [8, 14, 37, 38].

Lie groups [6, 39] have been previously used in [50] for detection and tracking of relatively rigid objects in 2D, however this requires the expensive image warping operations. [44] reviews in particular the recent development of applying shape manifold based approaches in tracking and action recognition. Its application in articulated objects is relatively sparse. Lie algebraic representation is considered in [31] for human full-body pose estimation based on multiple cameras or motion captured data. Rather than resorting to the traditional Jacobian matrices as in [31], learning based modules are employed in our approach to tackle the incurred inverse kinematics problem. [51, 52] also extract Lie algebra based features for action recognition. Instead of focusing on specific problem and object, here we attempt to provide a unified approach.

Part-based models have long been considered in the vision community, such as the pictorial structures [17], the flexible mixtures-of-parts [57], the poslet model [10], the deep learning model [48], among others. Meanwhile the idea of characterizing the geometric deformations of shapes or poses can be dated back to the shape deformation based descriptors of of D'Arcy Thompson [9] in the early 1900s. The most related works are probably [16, 45], where the idea of group action has been utilized. Moreover, multiple types of objects are also evaluated in [16] that focuses on the 2D pose estimation problem, while [45] is dedicated to 3D hand pose estimation. On the other hand, our approach aims to address these three related problems altogether in 3D, and we explicitly advocate the usage of Lie group theory. Note that the concept of pose indexed feature has been coined and employed in [2, 18]. In addition, learning based optimization has been considered in e.g. [54], although in very different contexts. Finally, the idea of learning the internal evaluation metric is conceptually related to the recent learning to rank approaches [13] in the information retrieval community for constructing ranking models. In the meantime, the idea of learning based methods instead of Jacobian matrices to tackle inverse kinematics is related to the recent works that learn to descend instead of directly solving the optimization problems at hand [5, 54].

3 Notations and Mathematical Background

The skeletal representation is in essence based on the group of rigid transformations in 3D Euclidean $\mathbb{R}^3$, a Lie group that is usually referred to as the special Euclidean group SE(3). In what follows, we provide an account of the related mathematical concepts that will be utilized in our paper.

An articulated object, such as a human hand, a mouse or a fish, is characterized in our paper by a skeletal model in the form of a kinematic tree that contains one or multiple kinematic chains. As illustrated in Fig. 2, a fish or a mouse skeleton both possess one kinematic chain, while a human hand contains a kinematic tree structure of multiple chains. Note that only the main spine is considered herein for the mouse model. The skeletal model is represented in the form of J_o joints interconnected by a set of bones or segments of fixed lengths. Empirical evidence has suggested that it is usually sufficient to use such fixed skeletal models with proper scaling, when working with pose estimation of articulated objects in depth images [45]. The pose of this object can thus be defined as a set of skeletal joint locations. Furthermore, we define the home position of an articulated object as a set of default joint locations. Taking a mouse model as depicted in the middle panel of Fig. 2 for example, its home position could be a top-view upward-facing mouse with the full body straight-up, and the bottom joint at the coordinate origin. Note this bottom joint contains 6 degrees of freedom (DoF) of the entire object, and is also referred to as the base joint. Then the pose could also be interchangeably referred to as the sequence of SE(3) transformations or group actions applied to the home position, $\Theta = \{\theta_1, \ldots, \theta_{J_o}\}$. The estimated pose is denoted as $\tilde{\Theta} = \{\tilde{\theta}_1, \ldots, \tilde{\theta}_{J_o}\}$ to better differentiate from the ground-truth pose. Here θ could be either ξ or $\hat{\xi}$ (to be discussed later) when without confusion in the context. To simplify the notation, we assume a kinematic chain contains J joints. Clearly $J = J_o$ for fish and mouse models, while $J < J_o$ for human hand or human full-body, by focusing on one of the chains. A depth image is not only a 2D image but also a set of 3D points (i.e. a 3D point cloud) describing the surface profile of such object under a particular view. Ideally the estimated pose (the set of predicted joint locations) is expected to align nicely with the 3D point cloud of the object in the observed depth image.

Before proceeding further with the proposed approach, let us pause for a moment to have a concise review of the involved mathematical background. Motivated readers may refer to [22, 27, 32] for further details.

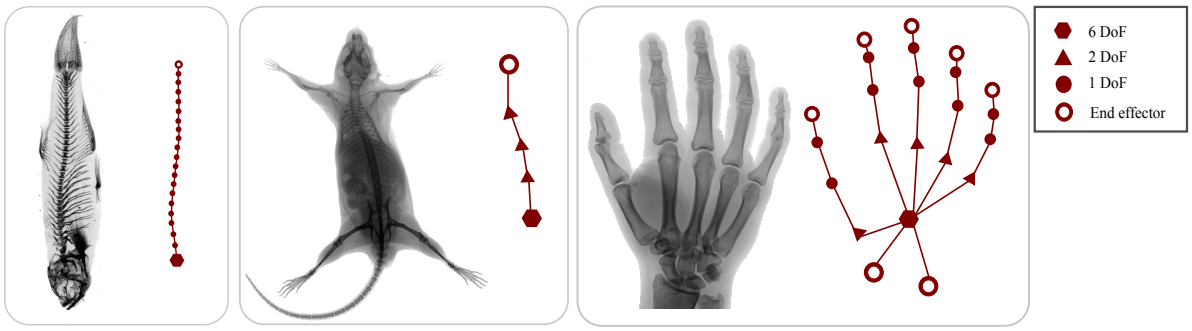


Fig. 2: A display of three different articulated objects considered in our paper, which are for (from left to right) fish, mouse, and human hand, respectively. Each of our skeletal models is an approximation of the underlying anatomy, presented as a side-by-side pair. Note end-effectors have zero degree of freedom (DoF). See text for details.

Lie Groups A Lie group G is a group as well as a smooth manifold such that the group operations of $(g, h) \mapsto gh$ and $g \mapsto g^{-1}$ are smooth for all $g, h \in G$. For example, the rotational group SO(3) is identified as the set of 3×3 orthonormal matrices $\{R \in \mathbb{R}^{3\times 3} : RR^\intercal = \mathrm{I}_3, \det(R) = 1\}$, with $R^\intercal$ denoting the transpose, $\det(\cdot)$ being the determinant, and I_3 being a 3×3 identity matrix. Another example is SE(3), which is defined as the set of rotational and translational transformations of the form $g(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$, with $R \in \mathrm{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$. In other words, g is the 4×4 matrix of the form

$$g = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\intercal & 1 \end{pmatrix}, \tag{1}$$

where $\mathbf{0} = (0, 0, 0)^\intercal$. Note the identity element of SE(3) is the 4×4 identity matrix I_4. Both I_3 and I_4 will be simply denoted as I if there is no confusion in the context. Now given a reference frame, a rigid-body transformation of two consecutive joints $\mathbf{x}$ and $\mathbf{x}'$ in a kinematic chain can be represented as $\begin{pmatrix} \mathbf{x}' \\ 1 \end{pmatrix} \leftarrow g \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$. Moreover the product of multiple SE(3) groups (i.e. a kinematic chain) is still a Lie group. In other words, as tree-structured skeletal models are considered in general for articulated objects, each of the induced kinematic chains forms a Lie group.

Lie Algebras and Exponential Map The tangent plane of Lie group SO(3) or SE(3) at identity I is known as its Lie algebra, $\mathrm{so}(3) := T_I\mathrm{SO}(3)$ or $\mathrm{se}(3) := T_I\mathrm{SE}(3)$, respectively. An arbitrary element of so(3) admits a skew-symmetric matrix representation parameterized by a three dimensional vector $\omega = (\omega_1, \omega_2, \omega_3)^\intercal \in \mathbb{R}^3$ as

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}. \tag{2}$$

In other words, the DoF of a full SO(3) is three. Note that a rotational matrix can alternatively be represented by the Euler angles decomposition [32]. A bijective mapping $\vee : so(3) \to \mathbb{R}^3$ and its reverse mapping $\wedge : \mathbb{R}^3 \to so(3)$ are defined as $\hat{\omega}^\vee = \omega$, and $\omega^\wedge = \hat{\omega}$, respectively. Let $\nu \in \mathbb{R}^3$, an element of se(3) can then be identified as

$$\hat{\xi} = \begin{pmatrix} \hat{\omega} & \nu \\ \mathbf{0}^\intercal & 0 \end{pmatrix}. \tag{3}$$

With a slight abuse of notation, similarly there exist the bijective maps $\hat{\xi}^\vee = \xi$, and $\xi^\wedge = \hat{\xi}$, with $\xi = (\omega^\intercal, \nu^\intercal)^\intercal$. Now a tangent vector $\xi \in \mathbb{R}^6$ (or its matrix form $\hat{\xi} \in \mathbb{R}^{4\times 4}$) is represented as $\hat{\xi} = \sum_{i=1}^{6} \xi^i \partial_i$, with ξ^i indexing the i-th component of ξ. Here $\partial_1 = (1, 0, \ldots, 0)^\intercal$, $\ldots$, $\partial_6 = (0, \ldots, 0, 1)^\intercal$, or in their respective matrix forms,

$$\partial_1 = \begin{pmatrix} 0&0&0&0 \\ 0&0&-1&0 \\ 0&1&0&0 \\ 0&0&0&0 \end{pmatrix}, \partial_2 = \begin{pmatrix} 0&0&1&0 \\ 0&0&0&0 \\ -1&0&0&0 \\ 0&0&0&0 \end{pmatrix}, \partial_3 = \begin{pmatrix} 0&-1&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix},$$

$$\partial_4 = \begin{pmatrix} 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}, \partial_5 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}, \partial_6 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{pmatrix}.$$

The exponential map $\mathrm{Exp} : \mathrm{se}(3) \to \mathrm{SE}(3)$ in our context is simply the familiar matrix exponential $\mathrm{Exp}_I(\hat{\xi}) = e^{\hat{\xi}} = I + \hat{\xi} + \frac{1}{2}\hat{\xi}^2 + \ldots$ for any $\hat{\xi} \in \mathrm{se}(3)$. From the Rodrigues's formula, it can be further simplified as

$$e^{\hat{\xi}} = \begin{pmatrix} e^{\hat{\omega}} & A\nu \\ \mathbf{0}^\intercal & 1 \end{pmatrix}, \tag{4}$$

with

$$A = I + \frac{\hat{\omega}}{\|\omega\|^2}\big(1 - \cos\|\omega\|\big) + \frac{\hat{\omega}^2}{\|\omega\|^3}\big(\|\omega\| - \sin\|\omega\|\big), \tag{5}$$

where $\|\cdot\|$ is the vector norm.

It has been known in the screw theory of robotics [32] that every rigid motion is a screw motion that can be realized as the exponential of a *twist* (i.e. a infinitesimal generator) $\hat{\xi}$, with its components ω and ν corresponding to the angular velocity and translation velocity of the segment (i.e. bone) around its joint, respectively.

Product of Exponentials and Adjoint Representation Consider a partial kinematic chain involving j joints with $j \in \{1, 2, \cdots, J\}$, which becomes the full chain when $j = J$. With a slight abuse of notation, let $g_{\Theta_{1:j}}$ be the Lie group action on the partial kinematic chain, and g_{θ_j} or simply g_j be the group action on the j-th joint. Its forward kinematics can be naturally represented as the product of exponentials formula, $g_{\Theta_{1:j}} = e^{\hat{\xi}_1} e^{\hat{\xi}_2} \cdots e^{\hat{\xi}_j}$. Therefore, for an end-effector from the home configuration (or home pose) $\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$, its new configuration is described by $\begin{pmatrix} \mathbf{x}' \\ 1 \end{pmatrix} = g_{\Theta_{1:j}} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = e^{\hat{\xi}_1} e^{\hat{\xi}_2} \cdots e^{\hat{\xi}_j} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$. As discussed in [32], this formula can be regarded as a series of transformations from the body coordinate b (for local joint) of each joint to the spatial coordinate s (for global kinematic chain). Let us focus on a joint j and denote $\hat{\xi}^{(b)}$ and $\hat{\xi}^{(s)}$ the twists of this joint in the body and spatial coordinates, respectively. Assume the transformation of this joint to the spatial coordinate is $g_{\Theta_{1:j-1}} = e^{\hat{\xi}_1} e^{\hat{\xi}_2} \cdots e^{\hat{\xi}_{j-1}}$. The two twists can be related by the adjoint representation

$$\hat{\xi}^{(s)} = \mathrm{Ad}_{g_{\Theta_{1:j-1}}} \left(\hat{\xi}^{(b)} \right) := g_{\Theta_{1:j-1}} \hat{\xi}^{(b)} g_{\Theta_{1:j-1}}^{-1},$$

which is obtained by

$$e^{\hat{\xi}^{(s)}} = g_{\Theta_{1:j-1}} e^{\hat{\xi}^{(b)}} g_{\Theta_{1:j-1}}^{-1} = e^{g_{\Theta_{1:j-1}} \hat{\xi}^{(b)} g_{\Theta_{1:j-1}}^{-1}},$$

and repeatedly applying the identity $g e^{\xi} g^{-1} = e^{g \xi g^{-1}}$ for $g \in \mathrm{SE}(3)$ and $\xi \in \mathrm{se}(3)$.

Geodesics It is known that SE(3) can not be endowed with a bi-invariant Riemannian metric. As a result several metric choices are proposed, as in e.g. [3]. In what follows we adopt the widely used Ad-invariant Riemannian metric [3]. Given two configurations g_1 and g_2, the geodesic curve between them is $g(\tilde{t}) = \begin{pmatrix} R(\tilde{t}) & A\mathbf{t}(\tilde{t}) \\ \mathbf{0}^\intercal & 1 \end{pmatrix}$, with $R(\tilde{t}) = R_1 e^{(\Omega_0 \tilde{t})}$, $\mathbf{t}(\tilde{t}) = (\mathbf{t}_2 - \mathbf{t}_1)\tilde{t} + \mathbf{t}_1$, $\tilde{t} \in [0, 1]$, and $\Omega_0 = \mathrm{Log}_I(R_1^{-1} R_2)$. Here the logarithm map Log_I or its simplified notion log can be regarded as the inverse of the exponential map.

Brownian Motion on Manifolds We refer interested readers to [22] for a more rigorous account of Brownian motion and stochastic differential geometry as they are quite involved. Here it is sufficient to know that Brownian motion can be regarded as a generalization of Gaussian random variables on manifolds, where the increments are independent and Gaussian distributed, and the generator of Brownian motion is the Laplace-Beltrami operator. In what follows we will focus more on the computational aspect [30]. Let $\tilde{t} \in \mathbb{R}$ denote a continuous variable, and $\delta > 0$ be a small step size. Let $\xi_{\tilde{t}} = (\xi_{\tilde{t}}^1, \cdots, \xi_{\tilde{t}}^6)^\intercal$ denote a random vector sampled from normal distribution $\mathcal{N}(0, C)$, for $k = 0, 1, \cdots$ with $C \in \mathbb{R}^{6 \times 6}$ being a covariance matrix. Then a left-invariant Brownian motion with starting point $g(0) \in \mathrm{SE}(3)$ can be approximated by

$$g\big((k+1)\delta\big) = g\big(k\delta\big) e^{\left\{ \sqrt{\delta} \sum_{i=1}^{6} \xi_k^i \partial_i \right\}}. \tag{6}$$

In addition, these sampled points can be interpolated by geodesics to form a continuous sample path. In other words, for $\tilde{t} \in \big(k\delta, (k+1)\delta\big)$ we have

$$g(\tilde{t}) = g\big(k\delta\big) e^{\left\{ \frac{\tilde{t} - k\delta}{\sqrt{\delta}} \sum_{i=1}^{6} \xi_k^i \partial_i \right\}}. \tag{7}$$

4 Our Approach

In what follows we describe the proposed *Lie-X* approach for pose estimation, tracking, and action recognition of various articulated objects.

Preprocessing & Initial Poses For simplicity we assume that there exists one and only one articulated object in an input depth image or patch. A simple preprocessing step is employed in our approach to extract individual foreground objects of interest. This corresponds to the point cloud of the object of interest extracted from the image. We then estimate the initial 3D location of base joint as follows: The 2D location of the base joint is set as the center of the point cloud, while its depth value is the average depth of the point cloud. Initial poses are obtained by first setting these poses as the home pose of the underlying articulate object, i.e. bones of the object are straight-up for the three empirical applications. For each of the initial poses, the initial orientation of the object is generated by perturbing the in-plane orientation of the above-mentioned base joint from a uniform distribution over $(-\pi, \pi)$. To account for the size variations, the bone lengths of each initial pose estimate are also scaled by a scalar that is uniformly distributed in the range of [0.9, 1.1].

Skeletal Models After preprocessing, an initial estimated pose is provided for an input depth image. The objects of interest are represented here in terms of kinematic chains. Without loss of generality, in this paper we focus on fish and mouse that both have one chain, as well as human hand that possesses multiple chains, as depicted in the respective panels of Fig. 2. Our fish and mouse models contain 21 and 5 joints (including the end-effectors) along the main spine, respectively, while our hand model has 23 joints. Their corresponding DoFs are 25, 12, and 26, respectively. Overall our models are designed as proper approximations following the respective articulate objects' anatomies. The base joint is fixed at coordinate origin which always has six DoFs describing 3D locations and orientations of the entire object; One DoF joints are applied to the rest fish joints characterizing the yaw of fish bones; Two DoFs are used for the rest mouse joints to account for both yaw and pitch; Similarly in our human hand model, two DoFs are used for each root joint of finger chain, while one DoFs are used for the rest joints. In all three models, zero DoFs are associated with the end-effectors, as each of them can entirely be determined by the preceding joints of the chain. Note that although simplified, the mouse model includes the most essential components (joints of the spine) at a reasonable resolution in our study. Our human hand model follows that of the existing literature (e.g. [35, 56]) that works with the widely-used NYU hand depth image benchmark [49].

4.1 Pose Estimation

Given a set of n_t training images, define

$$\overline{\Delta\theta}_j := \frac{1}{n_t} \sum_{i\in\{1,\cdots,n_t\}} \Delta\theta_j \tag{8}$$

the mean deviation over training images for the j-th joint of the set of J joints. The deviation $\Delta\theta_j$ characterizes the amount of changes between the estimated pose and the ground-truth pose, which is stated in Eq.(11). A global error function can be defined over a set of examples that evaluates the sum of differences from the mean deviation, as for example the following form,

$$\sum_{j\in\{1,\cdots,J\}} \|\overline{\Delta\theta}_j\|_2^2, \tag{9}$$

with $\|\cdot\|_2$ being the standard vector norm in Euclidean space. Presented with the above visual discrepancies of current results, our aim here is to improve skeletal estimation results.

Traditionally Jacobian matrices are employed for solving the incurred inverse kinematics problem. Here we instead advocate the usage of an iterative learning pipeline. Fig. 3 provides a visual mouse example that illustrates the execution pipeline of our pose estimation procedure at test run. This is also presented more formally in Algorithm 1. Meanwhile the corresponding training process is explained in Algorithm 2. Note that inside both the training and testing processes, an internal evaluation metric or scoring function is used, which is also learned from data. In what follows we are to explain each of the components in detail.

At test run, our approach behaves as follows: Assume for each of the J joints there are C rounds or iterations. As presented in Algorithm 1, given a test image and an initial pose estimation, for each joint $j \in \{1, \cdots, J\}$ following the kinematic chain of length J from the base joint, at current round $c \in \{1, \cdots, C\}$, the current pose of the joint will be corrected by the Lie group action $e^{r_j^{(c)}}$, with the twist $r_j^{(c)}$ being the output of a local regressor, $\mathcal{R}_{\mathrm{j}}^{(c)}$. In other words, denote the short-hand notations $g_{\tilde{\Theta}_{1:j-1}}^{(C)} := g_{\tilde{\theta}_1}^{(1:C)} g_{\tilde{\theta}_2}^{(1:C)} \cdots g_{\tilde{\theta}_{j-1}}^{(1:C)}$, $e^{\hat{\xi}_j^{(1:c-1)}} := e^{\hat{\xi}_j^{(1)}} e^{\hat{\xi}_j^{(2)}} \cdots e^{\hat{\xi}_j^{(c-1)}}$, and $g_{\tilde{\Theta}_{1:j}}^{(c-1)} := g_{\tilde{\Theta}_{1:j-1}}^{(1:C)} e^{\hat{\xi}_j^{(1:c-1)}}$, The j-th joint spatial coordinate can be updated by the following left group action

$$g_{\tilde{\Theta}_{1:j}}^{(c)} = g_{\tilde{\Theta}_{1:j}}^{(c-1)} e^{r_j^{(c)}}, \tag{10}$$

with $e^{r_j^{(c)}}$ being the latest group element used to further correct the spatial location of j-th joint at round c. It is worth emphasizing that this process requires learning the set of local regressors $\left\{\mathcal{R}_{\mathrm{j}}^{(c)}\right\}$, where the output of each regressor, $r_j^{(c)}$, is dedicated to a particular round c and joint j. Each of these local models is learned based on local features, i.e. the pose-indexed depth features that is described in details at section 4.4. In a sense, it endows our system with the ability to memorize local gradient updating rules from similar training patterns. This essentially forms the key ingredient that allows for removal of the commonly used Jacobian matrices for error-prorogation in our approach. Moreover, at test run, multiple initial poses are generated for each input image. They will then pass through our learned inverse kinematic regressors and produce corresponding candidate poses. These output poses will nevertheless be screened by our learned metric to be discussed later, where the optimal one is to be picked as the final estimated pose.

At training stage, a set of K initial poses of the input image is obtained in the same manner as those of the testing stage. The aforementioned local regressors are then learned as follows. Each example of the training dataset consists of an *instance*: a pair of poses including the estimated pose and the ground-truth pose,

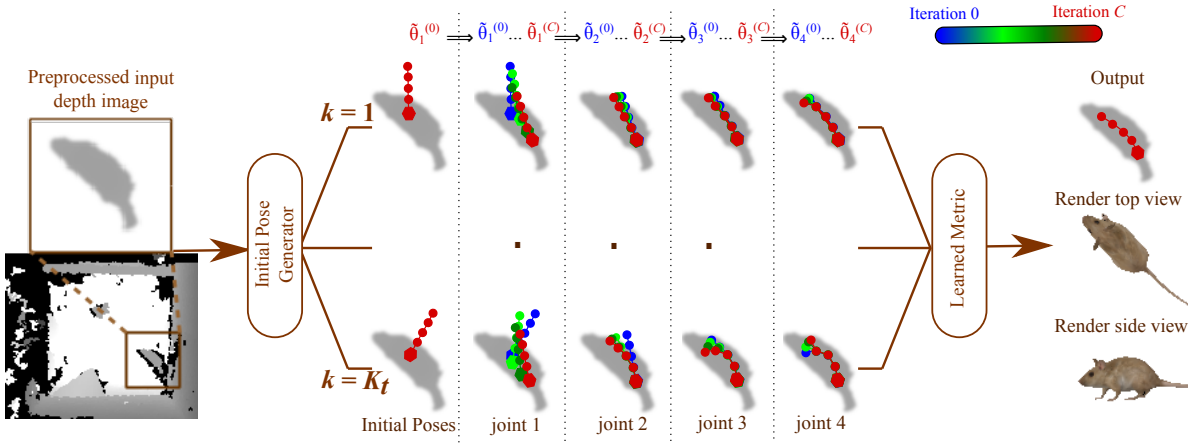


Fig. 3: An illustrative example of our pose estimation pipeline in Algorithm 1. In this example, a top-view depth image is used as the sole input. After a brief preprocessing and obtaining an initial pose, an iterative process is executed over each joint j and every round c, to produce its output estimate. For demonstration purpose, we also present a 3D virtual mouse fitted with the predicted skeletal model and with triangle meshing and skin texture mapping, which is then rendered in its top-view as well as side-view. Note the limbs of this virtual mouse are pre-fixed to default configuration.

$(\tilde{\theta}_j^{(c-1)}, \theta_j)$, as well as its *label*: the deviation of estimation $\tilde{\theta}_j$ from ground-truth θ_j, as

$$\Delta\theta_j = \log\left(g_{\tilde{\Theta}_{1:j}}^{(1:c-1)^{-1}} g_{\Theta_{1:j}}\right). \tag{11}$$

For the first joint $j = 1$ (the base joint in the kinematic chain) and at the first round $c = 1$, the label of an example will be the amount of changes from the first joint of the initial pose to that of the ground-truth. Then at any round c, its corresponding initial pose is obtained by executing the current partial kinematic model until the immediate previous round $c - 1$. Similarly for the second joint and at round c, the initial pose in each of the training examples is attained by executing the current partial model from base joint up to round $c - 1$ of the current joint, and its label is then the amount of changes from the current joint of the aforementioned initial pose to the second joint of the ground-truth. In this way, the training examples are prepared separately over joints and then across rounds until the very last joint J & round C. Algorithm 2 presents the procedure of learning the set of regressors $\left\{\mathcal{R}_{\mathrm{j}}^{(c)}\right\}$, with each regressor $\mathcal{R}_{\mathrm{j}}^{(c)}$ of round c and joint j being learned from its local context to make its prediction, $r_j^{(c)}$. Without loss of generality the random forest method [12] is engaged here as the learning engine.

Note that our hand skeletal model contains five kinematic chains, all of which share the hand base joint as root of the tree. For each chain, the sub-chain resulting from the exclusion of the base joint is independent of the other sub-chains given that the root is set. This motivates us to consider the following procedure: At test run, the base joint is first worked out, following the process described above. After this is done, Algorithm 1 is executed for each of the five sub-chains separately.

Algorithm 1 Pose Estimation: Testing Stage

Input: An unseen depth image
Output: Estimated skeletal joint locations and a prediction of its evaluation score
Preprocessing to obtain K_{t} initial poses by random perturbation of the home pose centered at the object point cloud.
for k=1:K_{t} **do**
  for j=1:J **do**
    for c=1:C **do**
      Twist prediction by applying a learned local regressor $\mathcal{R}_{\mathrm{j}}^{(c)} : \left(\tilde{\theta}_j^{(c-1)}, \theta_j\right) \mapsto r_j^{(c)}$.
      Update prediction of current joint spatial coordinate by applying the corresponding left group action of Eq.(10).
    end for
  end for
end for
Pick the best out of K_t candidates by applying the learned metric.

Learning the Internal Evaluation Metric Since multiple pose hypotheses are presented in our approach, it remains to decide on which one from the candidate pool we should choose as the final pose estimate. Traditionally this can be dealt with by either mode seeking or taking their empirical average as in e.g. Hough voting methods [19, 21, 28] or random forests [12], respectively; It could also be carried out by simply matching with a

Algorithm 2 Pose Estimation: Training Stage

Input: The set of training examples. For each example i, obtain K initial poses by random perturbations from the base system estimate.
Output: a series of learned regressors $\{\mathcal{R}_{\mathrm{j}}^{(c)} : j = 1, \cdots, J; c = 1, \cdots, C\}$.
for j=1:J **do**
  for c=1:C **do**
    Given the context, learn a local regressor $\mathcal{R}_{\mathrm{j}}^{(c)}$.
    Update prediction of current joint spatial coordinate by $\mathcal{R}_{\mathrm{j}}^{(c)}$ using Eq.(10).
  end for
  Prepare the training set of j+1-th joint spatial coordinate by applying $\left\{\mathcal{R}_{\mathrm{j}'}^{(c)}\right\}_{j'=1,c=1}^{j,C}$, the partial set of local regressors learned so far.
end for

small set of carefully crafted templates such as distance transform or DOT [20]. Instead we propose to learn a surrogate scoring function that is to be consistent with the *real* evaluation metric employed during practical quantitative analysis. This learned scoring function then becomes a built-in module in our approach to select the pose hypothesis with the least error.

More concretely, the widely used criteria of average joint error [55] is adopted as the evaluation metric for our scoring function to mimic. During training stage, a set of n_m training examples are generated, where a training example consists of an instance and a label: A training instance contains an input depth image, its ground-truth pose (i.e. skeletal joint locations) and an estimated pose as a set of corresponding joint locations after random perturbations from the ground-truth. Its label is the average joint error between the estimated and the ground-truth poses. Therefore a second type of regressor, $\mathcal{R}_{\mathrm{m}}$, is utilized here to learn to predict the error at test run. Namely, given an unseen depth image and an estimated pose, our regressor would produce a real-valued score mimicking the average joint error as where the ground-truth is known.

4.2 Tracking

Particle filters such as [24] have long been regarded as a powerful mean for tracking, and is also considered in our context to address the tracking problem. To facilitate a favorable balance between efficiency and effectiveness, we consider a probabilistic particle filter based approach only for the base joint, where particles are formed by Brownian motion based sampling in the pose manifold; Meanwhile the parameters of the remaining joints are obtained by invoking the same inference machinery as in our deterministic pose estimation algorithm. This design is also motivated from our empirical observation that often the object poses are also well-estimated when the prediction of the base joint is in close vicinity of the true values. That is, according to our observation the first joint is crucial in pose estimation: If ξ_1 is wrongly predicted, estimation results of the remaining joints could be seriously damaged; When our prediction of ξ_1 is accurate, the follow-up joints estimates would also be accurate. Algorithm 3 further presents the main procedure for our tracking task, which is also visually illustrated in Fig. 4, with a detailed description in the following paragraphs.

Following the particle filter paradigm [24] we consider a discretized time step t, and use x to denote a latent random variable as well as y for its observation. Here the state of a tracked object (i.e. the estimated pose $\tilde{\Theta}$ at time t) is denoted as x_t and its history is $x_{1:t} = (x_1, \cdots, x_t)$. Similarly, current observation is denoted as y_t, and its history as $y_{1:t} = (y_1, \cdots, y_t)$. The underlying first-order temporal Markov chain induces conditional independence property, which by definition gives $p(x_t|x_{1:t-1}) = p(x_t|x_{t-1})$. Following the typical factorization of this state-space dynamic model, we have

$$\begin{aligned} p(y_{1:t-1}, x_t|x_{1:t-1}) &= p(x_t|x_{1:t-1})p(y_{1:t-1}|x_{1:t-1}) \\ &= p(x_t|x_{t-1}) \prod_{i=1}^{t-1} p(y_i|x_i), \end{aligned}$$

with $p(y_{1:t-1}|x_{1:t-1}) = \prod_{i=1}^{t-1} p(y_i|x_i)$. We also need the posterior probability $p(x_t|y_{1:t})$ for filtering purpose, which in our context is defined as $p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1})$, with $p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}$. In other words, it is evaluated by considering the posterior $p(x_{t-1}|y_{1:t-1})$ from the previous time step in a recursively manner.

The realization of the particle filter paradigm in our context involves the three-step probabilistic inference process of selection-prorogation-measurement, which serves as the one time-step update rule in particle filter, and is also described in Algorithm 3. Specifically, the process at current time-step t corresponds to an execution of the selection-prorogation-measurement triplet steps: The output of previous time-step contains a set of K_r weighted particles

$$S_{t-1} := \left\{(s_{t-1}^{(i)}, \pi_{t-1}^{(i)})\right\}_{i=1}^{K_r}.$$

Here each particle i, $s_{t-1}^{(i)}$, corresponds to a particular realization of the set of tangent vector parameters $\tilde{\Theta}^{(i)}$ that uniquely determines a pose, where each of the vectors is attached to a joint following the underlying kinematic chain. The particle $s_{t-1}^{(i)}$ is also associated with its weight $\pi_{t-1}^{(i)} \in [0, 1]$. Collectively this set of weighted

particles is thus regarded as an approximation to the posterior distribution $p(x_{t-1}|y_{1:t-1})$. The *selection* step operates by uniform sampling from the cumulative distribution function (CDF) of $p(x_{t-1}|y_{1:t-1})$ to produce a new set of K_r particles with equal weights. It is followed by the *propagation* step where the manifold-based Brownian motion sampling of Eq.(7) is employed to realize $p(x_t|x_{t-1})$, i.e. to obtain the new state based on discretized Brownian motion deviation from the previous time-step. Note that this Brownian motion sampling is carried out only on the base joint, while the remaining joints are obtained by directly executing the same inference machinery of Eq.(10) as in our pose estimation algorithm. Now, the sample set constitutes an approximation to the predictive distribution function of $p(x_t|y_{1:t-1})$. The *measurement* step finally provides an updated weight $\pi_t^{(i)}$ for each particle $s_t^{(i)}$ as follows: Let $m_t^{(i)}$ be the predicted error value of the i-th particle $s_t^{(i)}$, obtained by applying our learned metric. The weight is thus evaluated as

$$\pi_t^{(i)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{m_t^{(i)2}}{2\sigma^2}}. \tag{12}$$

After obtaining all the K_r weights, each of the weights, $\pi_t^{(i)}$, is further normalized as

$$\pi_t^{(i)} \leftarrow \frac{\pi_t^{(i)}}{\sum_{i'=1}^{K_r} \pi_t^{(i')}}. \tag{13}$$

The updated sample set now collectively approximates the corresponding posterior distribution $p(x_t|y_{1:t})$ at time t.

The set of weighted particles allows us to represent the entire distribution instead of a point estimate as what we have done during the pose estimation task. The final pose estimate, x_t^* (i.e. $\tilde{\Theta}$ at time t), is produced by weighted averaging over this set of particles,

$$x_t^* \leftarrow \sum_{i=1}^{K_r} \pi_t^{(i)} s_t^{(i)}. \tag{14}$$

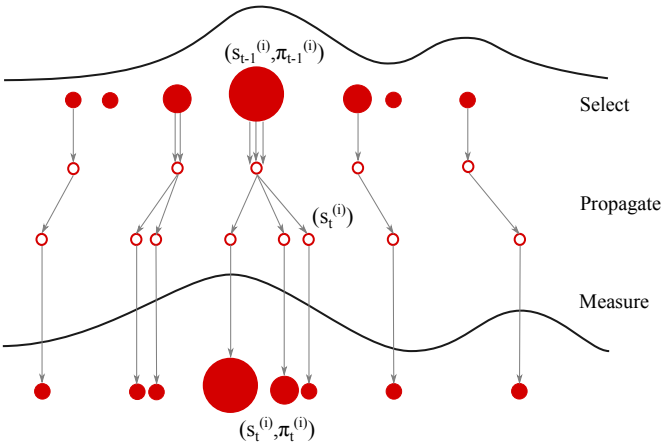


Fig. 4: A visual illustration of one time-step update process of the particle filter paradigm considered in our tracking task.

Algorithm 3 Tracking at time-step t

Input: S_{t-1}
Output: x_t^*, S_t
(1) Select:
calculate the normalized cumulative probabilities:
for $i = 1 \cdots K_r$ **do**
  Sample a particle $s_t'^{(i)}$ uniformly from the CDF of $p(x_{t-1}|y_{1:t-1})$.
end for
(2) Propagate:
for $i = 1 \cdots K_r$ **do**
  Obtain $s_t^{(i)}$ by sampling from $\left\{s_t'^{(i)}\right\}$ using the transition probability $p(x_t|x_{t-1})$, which is realized by manifold-based Brownian motion sampling of Eq.(7) of the tangent vector for the base joint, ξ_1, followed by directly executing Eq.(10) for each of the remaining joints following the kinematic chain.
end for
(3) Measure:
for $i = 1 \cdots K_r$ **do**
  Evaluate $\pi_t^{(i)}$ by Eq.(12).
end for
normalize $\pi_t^{(i)}$ by Eq.(13). Now $S_t = \left\{(s_t^{(i)}, \pi_t^{(i)})\right\}_{i=1}^{K_r}$ is ready.
(4) Estimate the pose
The estimated pose x_t^* is finally obtained by Eq.(14).

4.3 Action Recognition

Our approach can be further employed to work with the problem of action recognition. The key insight is that an action instance (i.e. a pose sequence) corresponds to a curve segment in the manifold, whereas the set of all instances of a particular action type corresponds to a group of curves. Therefore, the task of action recognition can be cast as separating different groups of action curves. It motivates us to consider a third type of learned predictor, $\mathcal{R}_\mathrm{a}$. Here dedicated features are extracted as to be described next, and the output concerns that of predicting its action categories.

Action Recognition Features As the length of action sequences may vary, they are firstly normalized to the same length (in practice 32 frames) using linear interpolation. Local features from each frame of a sequence can be obtained based on the tangent vectors (Lie algebras) of the estimated joints in the manifold. Each temporal sequence is further split into 4 equal-length segments, where the frames in a segment collectively contain the set of tangent vectors as local features. Moreover, a tem-

poral pyramid structural representation is utilized in a sense similar to that of the spatial pyramid matching [26], where features are extracted using hierarchical scales of {4,2,1}, where 4 corresponds to the 4 segments introduced previously, and the rest correspond to those coarser scales built over it layer by layer. In total it leads to 7 temporal segments (or sets of variable sizes) over these 3 scale spaces. For each such segment, the mean and standard deviation of its underlying pose representation (in terms of Lie algebras to be described below) are then used as features. Now let us investigate the details of these pose representations defined on single frames, which can be decomposed into joints following the aforementioned kinematic chain structure. For the base joint, we use the Lie algebras of the transformation from the current frame to the next frame. For the rest of the joints, we use the Lie algebras of the transformation from previous joint to the current joint and that of the transformation from current frame to the next frame. Besides, we also use the 3D location and orientation of the first joint (i.e. base joint) as features. In particular for fish-related actions, rather than using the Lie algebras of all 20 joints, we emphasize on robust estimation by considering a compact feature representation: The first component or sub-vector of the feature vector corresponds to the Lie algebra of the base joint; The second and the third components are the sub-vectors of the same length obtained by averaging over the set of Lie algebras from second to tenth joints, and from eleventh to the last joint, respectively. Overall a 252-dim feature vector is thus constructed to fully characterize an action sequence.

4.4 Random Forests, Pose-indexed Depth features, and Binary Tests

There are three types of learned predictors (namely the set of local regressors $\left\{\mathcal{R}_{\mathrm{j}}^{(c)}\right\}$, the learned internal metric $\mathcal{R}_{\mathrm{m}}$, and the action recognition predictor $\mathcal{R}_{\mathrm{a}}$) considered in our approach. In general any reasonable learning method can be used to realize these three types of predictors. In practice the random forest method [12, 19] is engaged for these learning tasks, so it is worthwhile to describe its details here.

For action recognition, a unique set of action features are used as stated previously. In what follows, we thus focus on the description of our pose-indexed depth features, which are used in the first two types of regressors, $\left\{\mathcal{R}_{\mathrm{j}}^{(c)}\right\}$, and $\mathcal{R}_{\mathrm{m}}$. Our feature representation can be regarded as an extension of the popular depth features as discussed in [42, 55] by incorporating the idea of pose-indexed features [2, 18] to model 3D objects. We start by focusing on a joint j with its current 3D location $\mathbf{x} \in \mathbb{R}^3$, where a 3D offset $\mathbf{u}$ can be obtained by random sampling from the home pose. Let $g_{\tilde{\Theta}_{1:j}}(\mathbf{u})$ denote the Lie group left action of current object pose $\tilde{\Theta}_{1:j}$ applied onto $\mathbf{u}$. Now the 3D location of the offset is naturally $\mathbf{x} + g_{\tilde{\Theta}_{1:j}}(\mathbf{u})$, and its projection onto 2D image plane under current camera view is denoted as $\bar{\mathbf{u}} = \mathrm{Proj}\big(\mathbf{x} + g_{\tilde{\Theta}_{1:j}}(\mathbf{u})\big)$. Similarly we can obtain another random offset $\bar{\mathbf{v}}$. For a 2D pixel location $\bar{\mathbf{x}} = \mathrm{Proj}\big(\mathbf{x}\big) \in \mathbb{R}^2$ of an image patch I containing the object of interest, its depth value can be denoted as $d_I(\bar{\mathbf{x}})$. Now we are ready to construct a feature $\phi_{I,(\bar{\mathbf{u}},\bar{\mathbf{v}})}(\bar{\mathbf{x}})$ or its short-hand notation ϕ, by considering two 2D offsets positions $\bar{\mathbf{u}}, \bar{\mathbf{v}}$ from $\bar{\mathbf{x}}$:

$$\phi_{I,(\bar{\mathbf{u}},\bar{\mathbf{v}})}(\bar{\mathbf{x}}) = d_I\Big(\bar{\mathbf{x}} + \bar{\mathbf{u}}\Big) - d_I\Big(\bar{\mathbf{x}} + \bar{\mathbf{v}}\Big). \tag{15}$$

Due to the visibility constraint, we are only able to obtain the depth values of the projected 2D locations $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ from the object surface. Thus Proj is a surjective map. Nevertheless, this serves our intention of sampling random features well. Following [12], a binary test is defined as a pair of elements, (ϕ, ϵ), with ϕ being the feature function, and ϵ being a real-valued threshold. When an instance with pixel location $\mathbf{x}$ passes through a split node of our binary trees, it will be sent to the left branch if $\phi(\mathbf{x}) > \epsilon$, and to the right side otherwise.

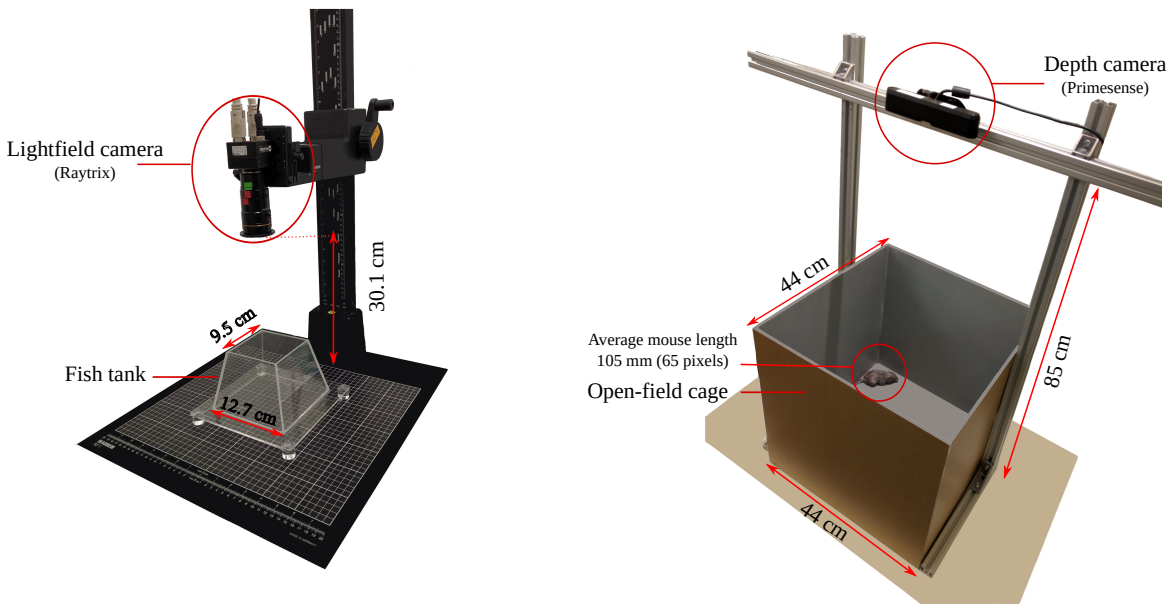
Our random forest predictors are constructed based on these features and binary tests for split nodes. Similar to existing regression forests in literature including e.g. [42], at a split node, we randomly select a relatively small set of m distinct features $\mathit{\Phi} := \{\phi_i\}_{n=1}^{m}$ as candidate features. For every candidate feature, a set of candidate thresholds $\mathit{\Lambda}$ is uniformly selected over the range defined by the empirical set of training examples in the node. The best test (ϕ^*, ϵ^*) is chosen from these features and accompanying thresholds by maximizing the following gain function. This procedure is then repeated until there are L levels in the tree *or* once the node contains fewer than l_n training examples. More specifically, the above-mentioned split test is obtained by

$$(\phi^*, \epsilon^*) = \arg \max_{\phi \in \mathit{\Phi}, \epsilon \in \mathit{\Lambda}} \mathcal{I}(\phi, \gamma),$$

where the gain $\mathcal{I}(\phi, \epsilon)$ is defined as:

$$\mathcal{I}(\phi, \epsilon) = E(S) - \left(\frac{|S_l|}{|S|} E(S_l) + \frac{|S_r|}{|S|} E(S_r)\right). \tag{16}$$

Here $|\cdot|$ denotes the cardinality of the set, S denotes the set of training examples arriving at current node, which is further split into two subsets S_l and S_r according to the test (ϕ, ϵ). Define $\overline{\Delta\theta}_j := \frac{1}{\|S\|} \sum_{i \in S} \Delta\theta_j$ the mean deviation of the set to j-th joint, and accordingly for S_l



(a) (b)

Fig. 5: The capture setups used in constructing our (a) fish and (b) mouse depth image datasets, respectively.

and S_r. The function E is defined over a set of examples that evaluates the sum of differences from the mean deviation:

$$E(S) = \sum_{i \in S} \|\overline{\Delta\theta}_j\|_2. \quad (17)$$

In the final decision stage, for the first two regression modules, the mean-shift mode searching in Hough voting space is used as e.g. in [19], while for the third module (action recognition) the classical random forest strategy [12] is used to pick the category with largest counts from the averaged histogram.

5 Empirical Evaluations

Empirically our *Lie-X* approach is examined on three different articulated objects: fish, mouse, and human hand.

Performance Evaluation Metric Our performance evaluation metric is based on the commonly-used *average joint error*, computed as the averaged Euclidean distance in 3D space over all the joints. Formally, let $\mathbf{v}_g$ and $\mathbf{v}_e$ be the ground-truth and estimated joint locations, respectively. The joint error of the pose estimate $\mathbf{v}_e$ is defined as $e = \frac{1}{m}\sum_i \|v_{gi} - v_{ei}\|$, where $\|\cdot\|$ is the Euclidean norm in 3D space. When dealing with test images, let $k = 1, \ldots, n_{\text{tst}}$ index over the test images, and their corresponding joint errors denoted as $\{e_1, \cdots, e_{n_{\text{tst}}}\}$. The *average joint error* is then defined as $\frac{1}{n_{\text{tst}}}\sum_j e_j$.

Internal Parameters Throughout experiments, a fixed set of values is always used for the internal parameters of our approach, unless otherwise stated, as follows. For the first type of regressors (namely the set of local regressors $\left\{\mathcal{R}_{\text{j}}^{(c)}\right\}$), the number of trees is fixed to (3, 10, 10), while the tree depth is (24, 24, 24) for the triplet of articulated objects (fish, mouse, hand), respectively. For the second type (the learned internal metric $\mathcal{R}_{\text{m}}$), the number of trees is (20, 20, 20), while the tree depth is (15, 15, 20) for the triplet of articulated objects (fish, mouse, hand), respectively. For the third type (the action recognition predictor) $\mathcal{R}_{\text{a}}$, the number of trees in the forest is 50, and tree depth is 20. The number of features is m=8,000, and the maximum number of examples in the leaf node is l_n=5. The number of rounds at each joint is set to C=7, 3, and 3, for fish, mouse, and hand, respectively. The local image patch sizes considered in our approach for fish, mouse, and hand are normalized to 25×25, 100×100, 100×100 mm^2, respectively. These patches are used as input to the local random forest regressors in our approach to estimate the spatial coordinate of next joint based on current joint following the kinematic chain. One important parameter is K_{t}, the number of initial poses in pose estimation. In practice, after factoring-in the efficiency consideration, K_{t} is set to 40, 40, 20 for

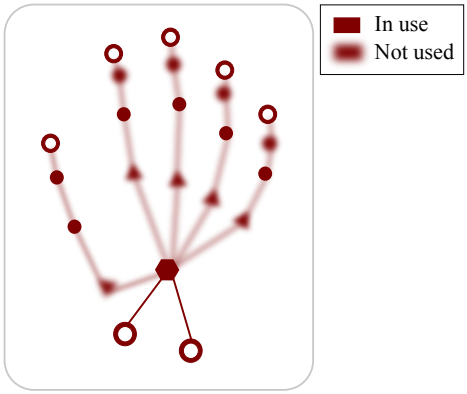


Fig. 6: Following the evaluation protocol of [34, 35, 49, 58], for NYU hand depth image dataset, only a subset of 14 joints out of the total 23 hand skeletal joints is considered during performance evaluation for hand pose estimation.

pose estimation of fish, mouse, and hand, respectively, throughout experiments. Similarly, the number of initial poses for tracking is set to K_{r}=200. For learning the internal evaluation metric, the number of candidates is set to 8. Namely, given a training dataset of size n_t, the number of training examples for pose estimation becomes $K_{\mathrm{t}} \times n_t$, while the number of training examples for learning the internal metric is $n_m = 8 \times n_t$.

5.1 Datasets

To examine the applicability of our approach on diverse articulated objects, we demonstrate in this paper its empirical implementation for fish, mouse, and human hand, respectively, where three distinct real-life datasets are employed. In particular, here we introduce our home-grown 3D image datasets of zebrafish and lab mouse that are dedicated to the related problems of pose estimation, tracking, and action recognition. The images have been captured and annotated by experts to provide the articulated skeleton information describing the pose of the subject. The popular NYU hand depth image dataset [49] is also considered here. More details of the datasets are discussed next. It is worth noting that different imaging modalities are utilized across the three datasets: light-field depth images are used for fish, while structured illumination depth cameras are employed for mouse and human hand objects. Regardlessly our approach is demonstrated to work well across these diverse image modalities.

Our Fish Dataset Depth images are acquired with a top-mount Raytrix R5 light-field camera at a frame rate of 50 FPS and a resolution of $1,024 \times 1,024$, as displayed in Fig. 5(a). The depth images are obtained from the raw plenoptic images by utilizing Raytrix on-board SDK. In total 7 different adult zebrafish of different genders and sizes are engaged in our study. From the captured images, 2,972 images containing distinct poses are annotated. The training dataset of n_t =95,104 images is thus formed by augmenting each fish object of these images with 31 additional transformations, where each transformation comes with a random scaling within [0.9, 1.1] and with a random in-plane rotation within (-π, π). The test dataset of pose estimation problem contains n_{tst} =1,820 distinct fish images.

In addition to single-frame based pose annotations, we also record, annotate, and make available a *fish action* dataset. [25] provides a comprehensive catalogue of zebrafish actions, from which a subset of 9 action classes are considered in this paper, which is listed below as well as illustrated in Fig.19:

Scoot: Moves along a straight line.
J-turn: Fine reorientation during which the body slightly curves ($30° - 60°$), with a characteristic bend at the tail.
C-turn: Fish body curves to form a C-shape en route to a near 180° turn.
R-turn: Involves routine angular turn of greater than 60°.
Surface: Moves up towards the water surface.
Dive: Moves towards the tank bottom.
Zigzag: Contains erratic movements with multiple darts in various directions.
Thrash: Consists of forceful swimming against the side or bottom walls of the tank.
Freeze: Refers to complete cessation of movement.

Our fish action dataset contains 426 training sequences and 173 testing sequences, respectively, from 7 different fish over the aforementioned 9 categories. The length of each fish action sequence varies from 7 frames to 135 frames.

Our Mouse Dataset Mouse depth images are collected using a top-mount Primesense Carmine depth camera at a frame rate of 30 FPS and with a resolution 640 × 480. Fig. 5(b) presents our dedicated imaging set-up. Two different lab mouse are engaged in our study. We select 3,253 images containing distinct poses and depth noise patterns, and augment them with additional transformations following the same protocol as above, which gives rise to the training dataset here containing n_t =104,096 images. The testing dataset of pose estimation problem contains n_{tst} =4,125 distinct depth images. For tracking problem, the test set consists of two sequences of length 511 and 300 frames, respectively.

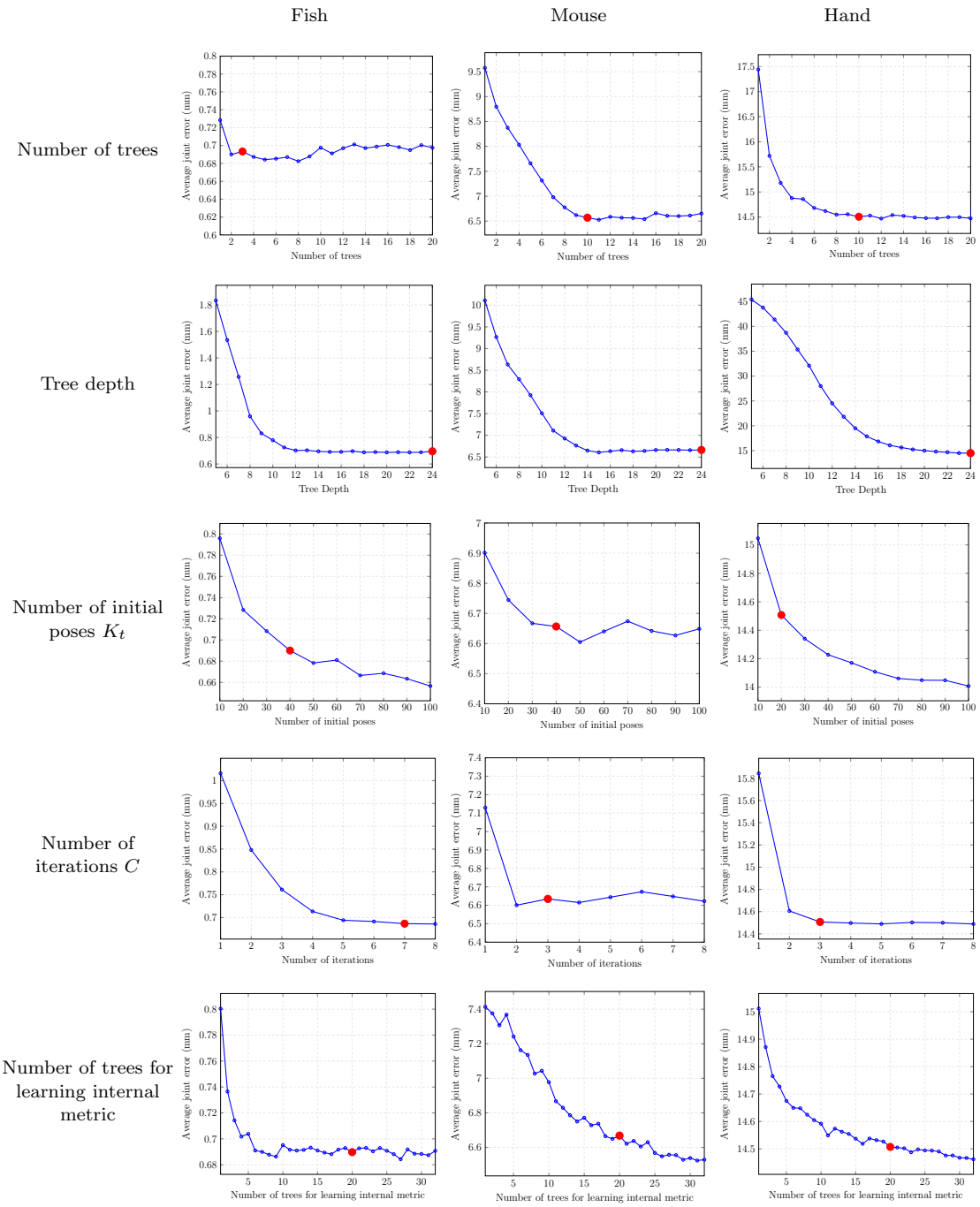
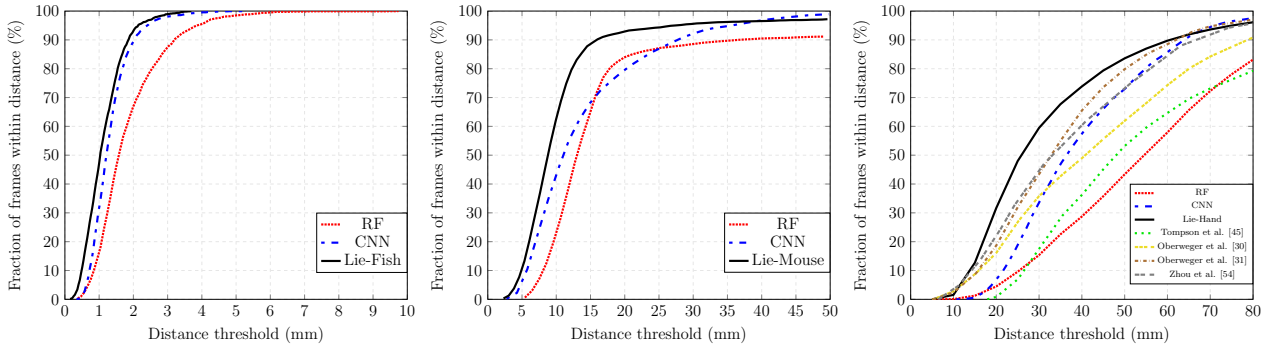


Fig. 7: Sensitivity analysis of our *Lie-X* approach w.r.t. internal parameters for pose estimation tasks: In each of the five rows, average joint error is plotted as a function of the respective internal parameter. It is further displayed in three columns for fish, mouse, and hand, respectively. In each of the panels, a red dot is placed to indicate the specific parameter value empirically employed in our approach.

The NYU Hand Dataset We also evaluate our approach on the benchark NYU hand depth image dataset[49] [2]. It contains n_t =72,757 depth images for training and n_{tst}=8,252 frames for testing. All images are depth images captured by Microsoft Kinect using the structured illumination technique, which is the same as the Primesense camera used in our mouse dataset. Depth images in the training set are from a single user, while images in the test set are from two users. While a ground-truth hand label contains 36 annotated joints, only 14 of these

[2] The NYU dataset is publicly available at `http://cims.nyu.edu/~tompson/NYU_Hand_Pose_Dataset.htm`.



(a) (b) (c)

Fig. 8: Cumulative error distribution curves for pose estimation of (a) fish, (b) mouse and (c) hand, respectively. Horizontal axis displays the distance amount in mm of the estimated poses from ground-truths. Vertical axis presents the fraction of examples where their corresponding estimated poses possess average joint errors within the current distance range.

Comparison methods on articulated objects	fish	mouse
RF	1.28	12.24
CNN	0.79	9.17
Lie-X (w/o multiple initial poses)	3.28	13.27
Lie-X (w/o learned metric)	1.71	9.82
Lie-X	**0.68**	**6.64**

Table 1: Quantitative evaluation of competing methods on pose estimation problem for fish and mouse respectively. Performance is measured in terms of average joint error (mm).

Comparison methods on hand pose estimation	average joint error (mm)
RF	24.81
CNN	18.82
Tompson et. al. [49]	21.00
Oberweger et. al. [34]	20.00
Oberweger et. al. [35]	16.50
Zhou et. al. [58]	16.90
Lie-X (w/o multiple initial poses)	20.50
Lie-X (w/o learned metric)	16.72
Lie-X	**14.51**

Table 2: Quantitative evaluation of competing methods on the benchmark NYU dataset [49] for hand pose estimation task. Performance is in terms of average joint error (mm).

joints are considered in many existing efforts using this dataset, such as [34, 35, 49, 58], which is followed during our experiments. This is also presented in Fig. 6: Important hand joints are included in this subset of 14 joints, such as all the finger tips and the hand base.

5.2 Pose Estimation of Fish, Mouse, and Human Hand

In this subsection, we focus on the problem of pose estimation for articulated objects such as fish, mouse, and hand. To make a fair comparison with existing methods, we specifically implement two non-trivial baseline methods, namely regression forest (RF), and convolutional neural network (CNN). The RF method is a re-implementation of the classical regression method used by Microsoft Kinect [42], with the only difference being that our RF implementation explicitly utilizes a skeletal model, instead of estimating joint locations without skeletal constraints as in [42]. Two separate regression forests, F_1 and F_2, are trained for this purpose. Here F_1 is used to estimate the 3D location and in-plane orientation of the subject, followed by F_2 which produces a set of 3D pose candidates. The number of trees trained are set to 7 and 12 for F_1 and F_2, respectively. In both cases, the maximum tree depth is fixed to L=20. The standard depth invariant two-point offset features of [42] are also used. The CNN method is obtained as follows: The pre-trained AlexNet CNN model from ImageNet is engaged as the initial model. To tailor the training data for our CNN, objects of interest from the training depth images are cropped according to their bounding boxes. The depth values in each patch are rescaled to be in the range of 0 to 255. Each object patch is replicated three times to form into a RGB image, which is then resized as an input instance of size 224 $\times$ 224. This together with its corresponding annotation prepares a training example. Then our CNN model is finally obtained by executing the MatConvNet package to train on these training examples for 50 epochs.

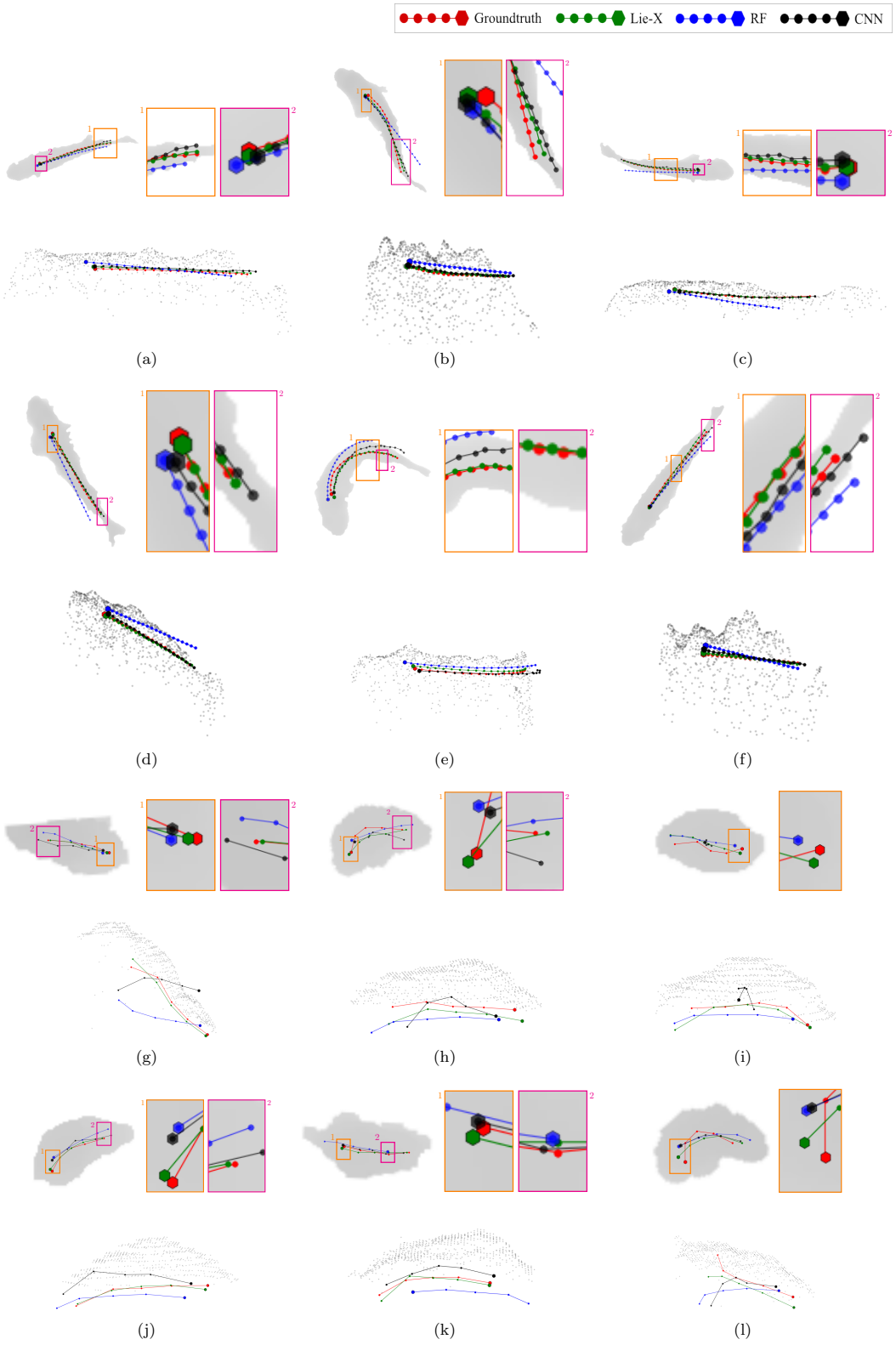


Fig. 9: Visual comparison of fish and mouse examples. Here pose estimates of RF, CNN, as well as our *Lie-X* approach are compared together with respective human-annotated ground-truths. Panels (a)–(f) present six fish examples, which is followed by panels (g)–(l) for six exemplar mouse results. In each of the twelve panels, top row displays the full top-view together with one or two zoom-in visual examinations. Meanwhile, the bottom row also provides a side-view. Best viewed in color.

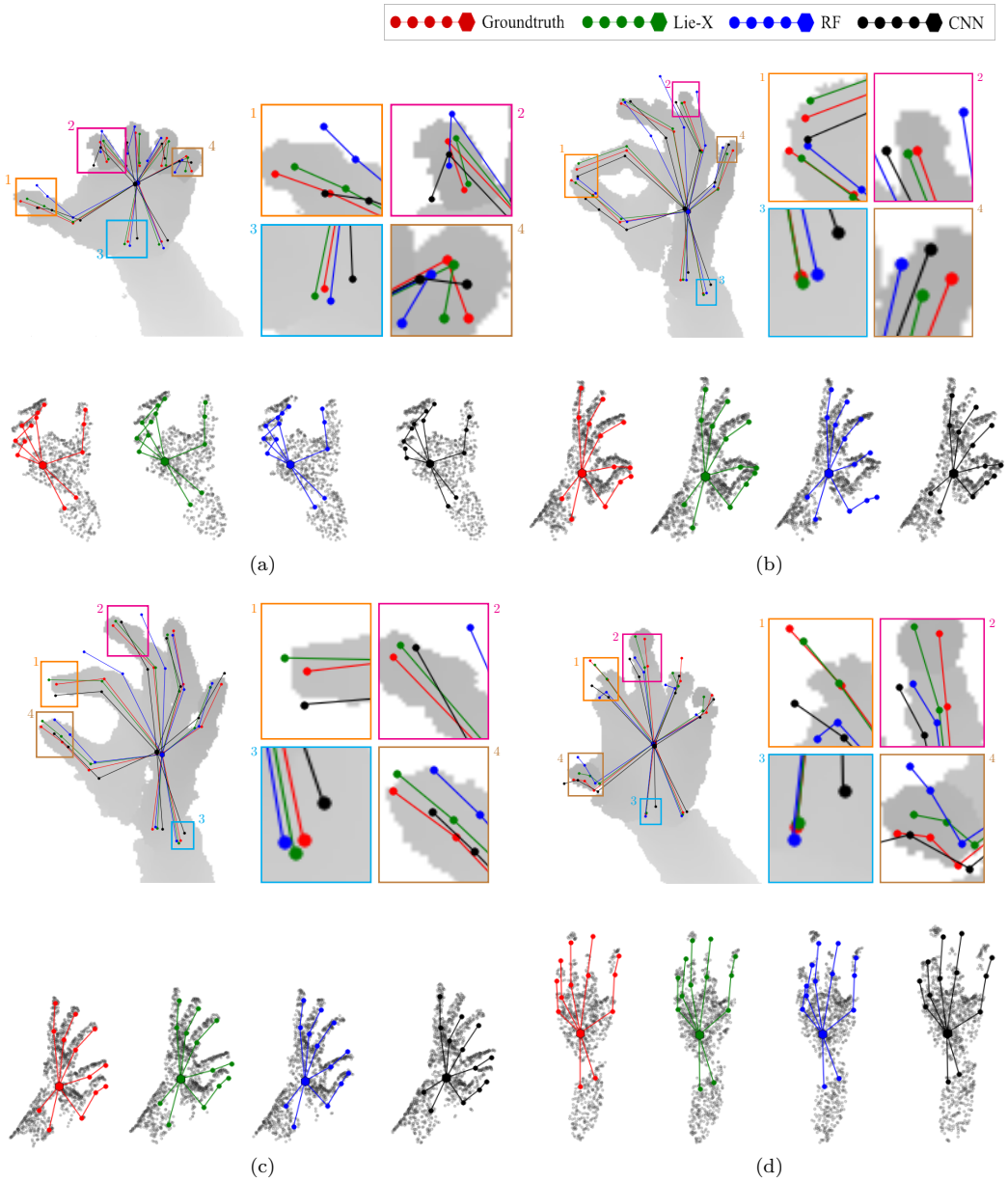


Fig. 10: Visual comparison of hand examples. Here pose estimates of RF, CNN, as well as our *Lie-X* approach are compared together with respective human-annotated ground-truths. In each of the four panels, top row displays the full top-view together with four zoom-in visual examinations. Meanwhile, the bottom row displays side-views of the respective methods. Best viewed in color.

Sensitivity Analysis of the Internal Parameters As our approach contains a number of internal parameters, it is of interest to systematically investigate the influence of these parameters w.r.t. the final performance of our system. Here we consider five influential parameters, which are the number of initial poses K_t, the number of rounds C, the number and depth of trees in our first type of regressors (i.e. the local regressors $\left\{\mathcal{R}_{\mathrm{j}}^{(c)}\right\}$), as well as the number of trees used in our learned metric component $\mathcal{R}_{\mathrm{m}}$. Fig. 7 displays the performance of *Lie-X* with respect to each of these five parameters row-by-row. Meanwhile each of the three columns presents the respective results for fish, mouse, and hand. Each of the fifteen panels in this five by three matrix is obtained by varying one parameter of interest while keeping the other parameters fixed to default values. In general, our system behaves in a rather stable manner w.r.t. the change of internal parameters over a wide range of values. Moreover, in each of the panels, a red colored dot is placed to indicate the specific parameter value em-

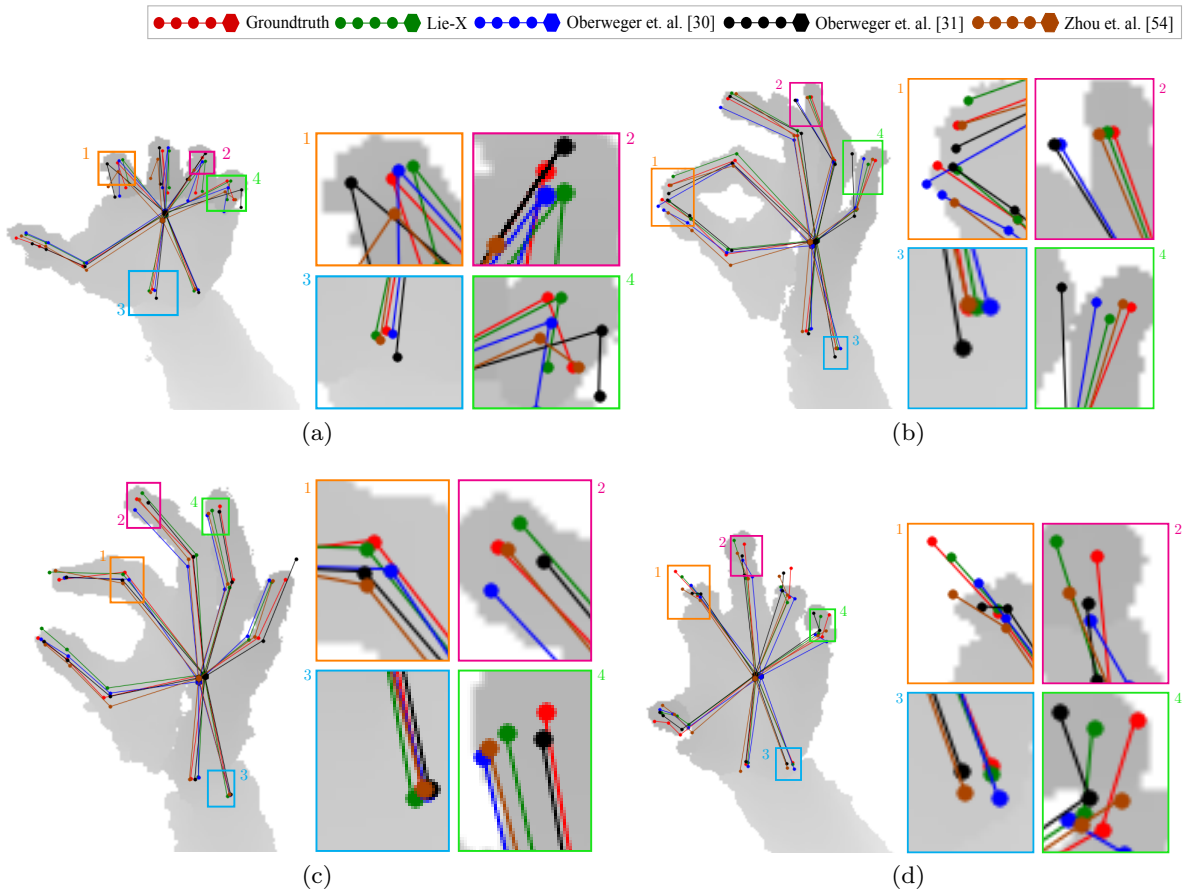


Fig. 11: Visual comparison of *Lie-X* as well as the state-of-the-art methods on the same four input hand images presented in Fig. 10. In each of the panels, the corresponding example is presented with four zoom-in visual examinations. Best viewed in color.

Comparison methods	uses GPU	frames per second (FPS)
Tompson et. al. [49]	✓	30
Oberweger et. al. [34]	✓	**5,000**
Oberweger et. al. [35]	✓	400
Zhou et. al. [58]	✓	125
Lie-X	×	123

Table 3: Runtime speed comparison with state-of-the-art methods for hand pose pose estimation task. Note our *Lie-X* results are obtained using *CPU only*, while the rest methods all utilize GPUs.

pirically employed by our approach in this paper. It is worth noting that the choice of these internal parameter values represents a compromise between performance and efficiency.

Comparison with Baselines and the State-of-the-art Methods To evaluate the performance of the proposed approach, a series of experiments are conducted on the aforementioned datasets for fish, mouse, and hand pose estimation tasks. Table 1 presents a comparison of *Lie-X* w.r.t. the two non-trivial baseline methods (i.e. RF and CNN) on fish and mouse pose estimation tasks. Overall, our approach clearly outperforms the others by a significant margin, while CNN achieves better results over RF. Moreover, the error distributions of these comparison methods are also presented in Fig. 8(a-b), where our approach clearly outperforms the baselines most of the time. The superior performance of our *Lie-X* approach is also demonstrated in Fig. 9, which provides visual comparisons of pose estimation results for six fish and six mouse examples, respectively, over the three competing methods. From these visual examples, it is observed that the estimated poses from our *Lie-X* approach tend to be more faithfully aligned with the ground-truth when compared against the two baseline methods.

Our approach is also validated on the NYU hand depth benchmark, as is displayed in Table 2. Overall, our CNN baseline result is on par with the standard deep learning results of e.g. [34] that also utilizes

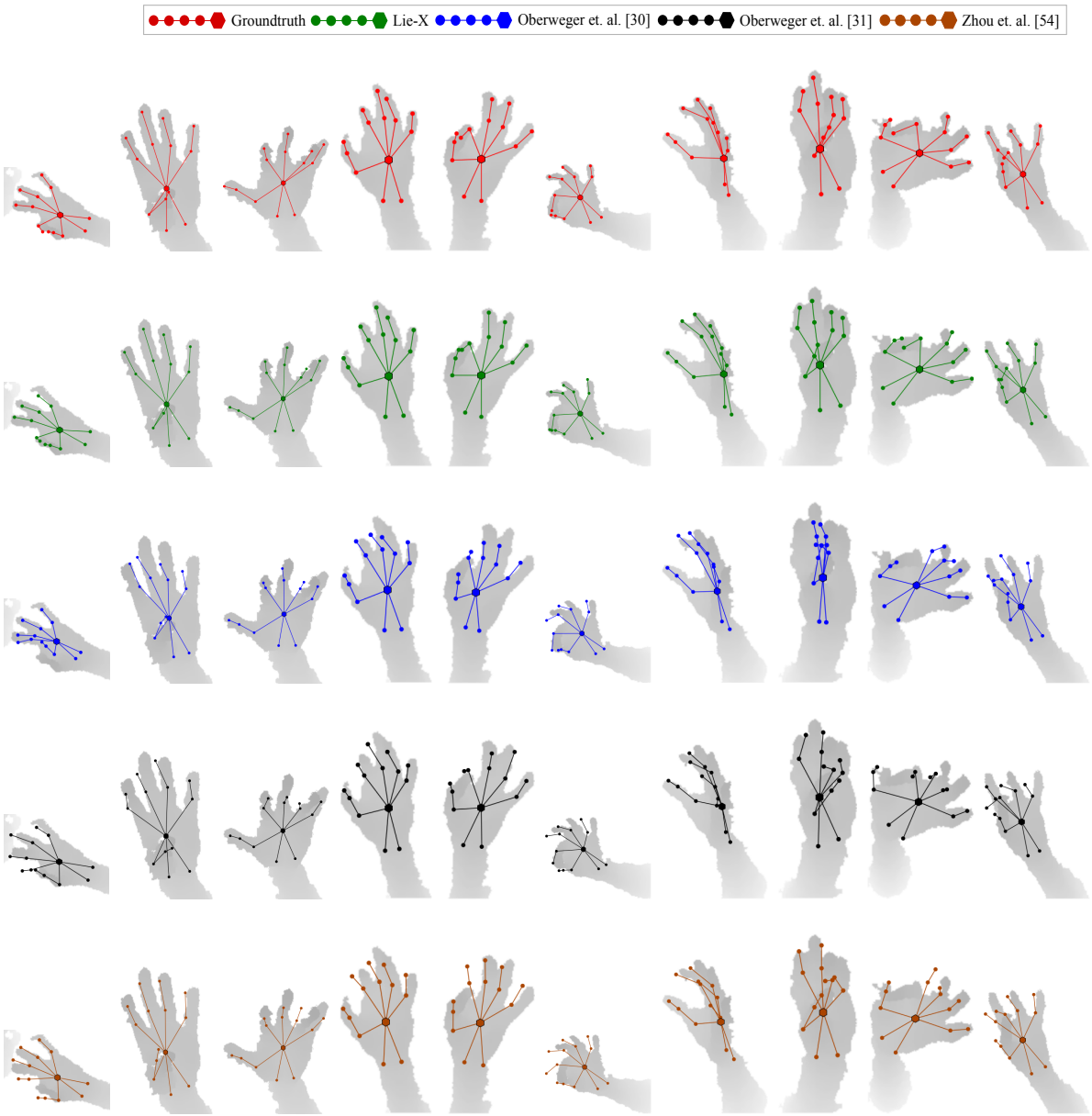


Fig. 12: Visual comparison of *Lie-X* results and the state-of-the-art methods on ten additional hand examples. Each column presents an example, while each row displays results from a particular competing method. Best viewed in color.

a AlexNet-like CNN. This helps to establish that our baselines are consistent in terms of performance with what has been reported in the literature, which are also used as pose estimation baselines on fish and mouse objects. Moreover, the results of the state-of-the-art methods are also directly compared here, including Tompson et. al. [49], Oberweger et. al. [35], and Zhou et. al. [58]. It is worth pointing out that the test error rate of our approach is 14.51 mm in terms of average joint error. This is by far the best result on hand pose estimation task to our knowledge, which improves over the best state-of-the-art result of 16.50 mm of [35] by almost 2 mm. More detailed quantitative information is revealed through the error distributions of comparison methods in Fig. 8(c), where our approach clearly outperforms the baselines and the state-of-the-art methods. Similarly, visual comparison results are provided in Fig. 10, Fig. 11, and Fig. 10, where our approach is again shown to clearly outperform other methods. More specifically, Fig. 10 and Fig. 11 present the visual results of all competing methods on the same four exemplar hand images. Due to the access limit, we are only

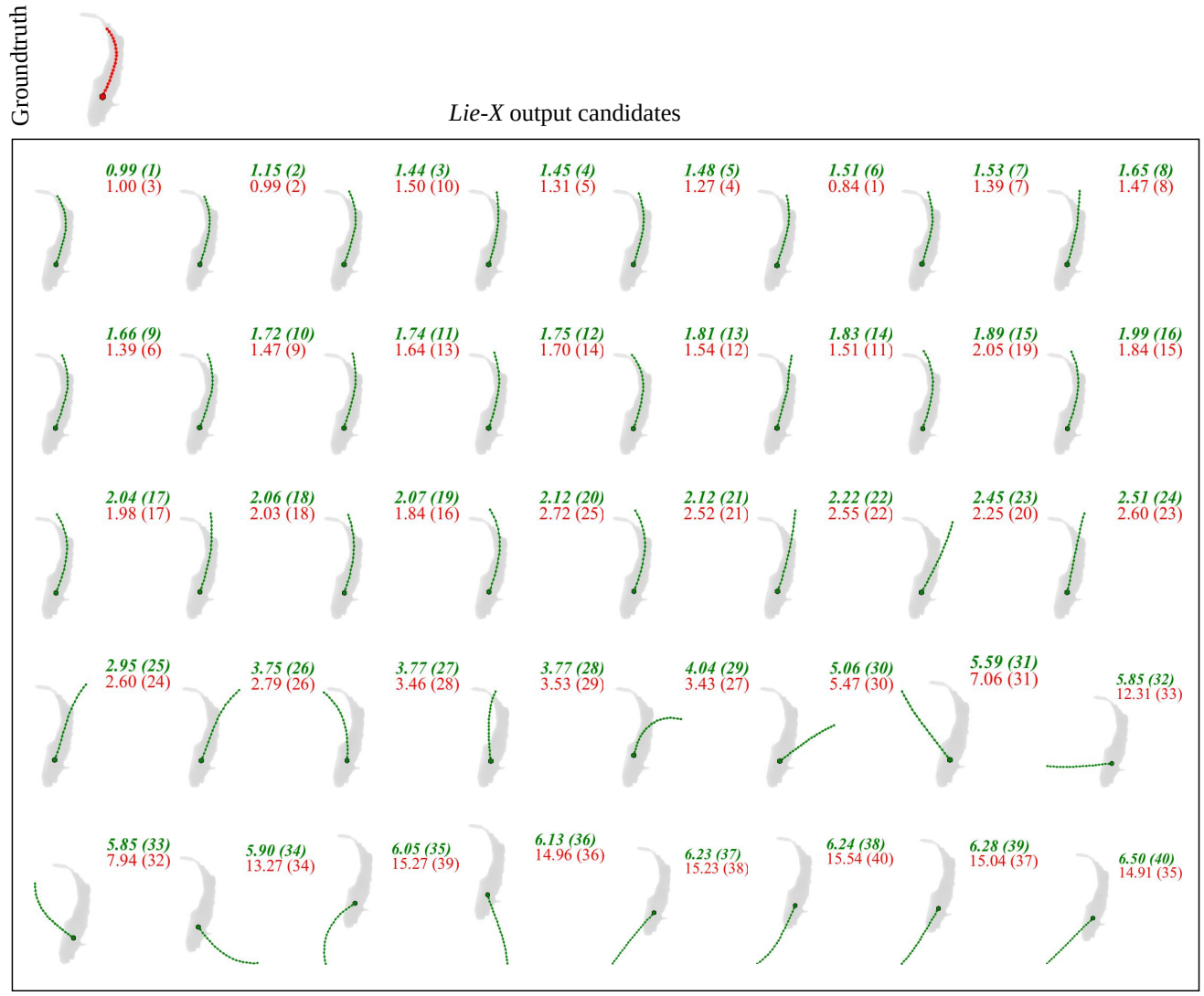


Fig. 13: An example of fish pose estimation that visually illustrates the inner-working of the learned internal metric in our approach applied onto the $K_t = 40$ pose candidates of the input depth image. Here green colored numbers correspond to the scores (lower the better here) and ranking results obtained by applying the learned metric, red colored numbers denote the corresponding actual evaluation scores and ranking results by engaging the empirical evaluation metric of average joint error when we have access to the ground-truth annotations. See text for details.

able to present the side-view results on our approach and the baseline methods of RF and CNN. Fig. 10 provides additional visual results comparing our approach to state-of-the-arts on ten more hand images.

To reveal the inner working of our approach, in Fig. 13, Fig. 14, and Fig. 15, a visual example is respectively provided for pose estimation of fish, mouse, and hand. It is evident that a diverse set of pose candidates are obtained that covers distinct pose location, orientation, and sizes. This is made possible due to the adoption of multiple initial poses. Moreover, prediction scores and associated orders from our learned metric module in general closely resembles that of the empirical evaluation metric. In addition, Fig. 16 presents several intermediate pose estimation results from different joints and rounds on an exemplar mouse image, when executing the pose estimation pipeline as illustrated in Fig. 3. It is observed that each step of the iterative process usually helps in converging toward the final pose estimation.

With vs. Without Multiple Initial Poses As presented in Table 1 for fish and mouse objects and Table 2 for hand objects, empirically we observe that the presence of multiple initial poses always improves the pose estimation performance. As presented in Figs. 13, 14,

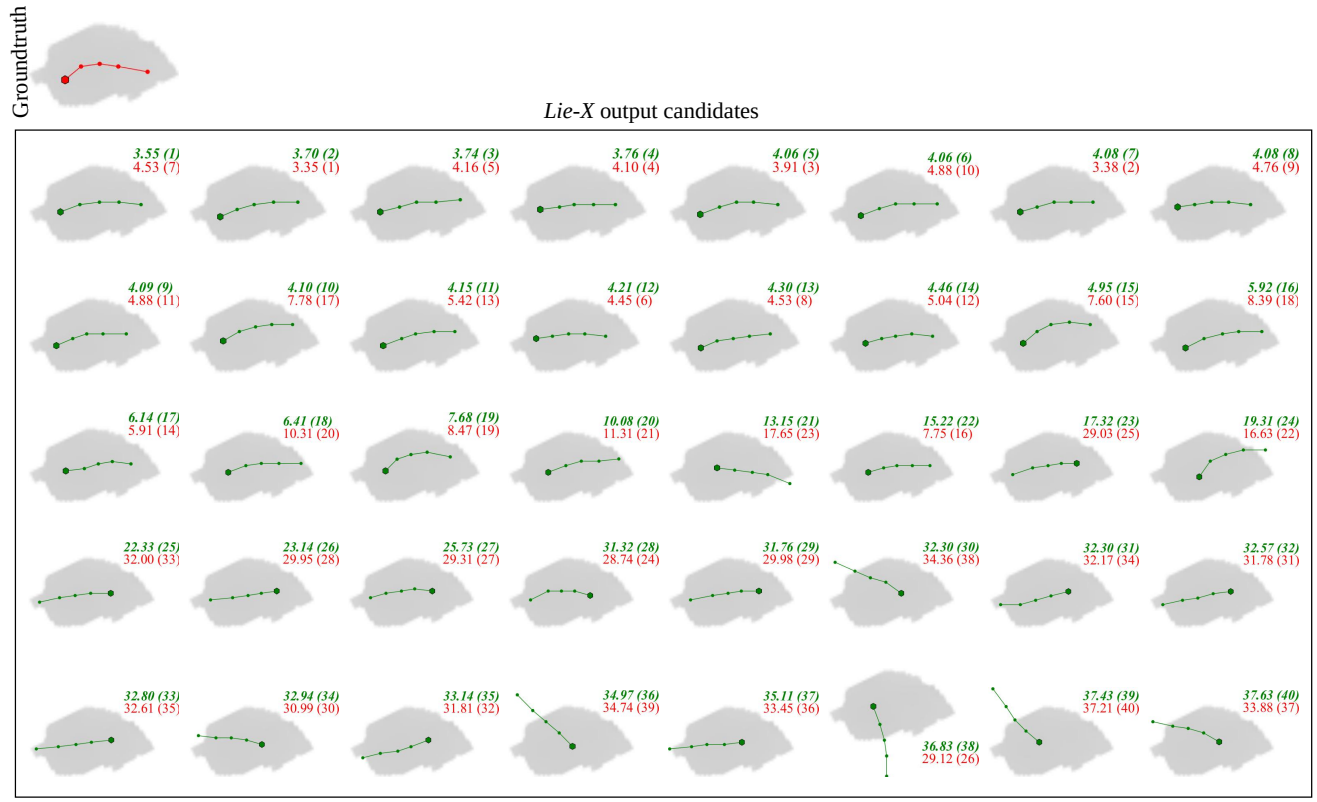


Fig. 14: An example of mouse pose estimation that visually illustrates the inner-working of the learned internal metric in our approach applied onto the $K_t = 40$ pose candidates. Here green colored numbers refer to the scores (lower the better here) and ranking results obtained by applying the learned metric, red colored numbers are the corresponding actual evaluation scores and ranking results by engaging the empirical evaluation metric of average joint error when we have access to the ground-truth annotations. See text for details.

and 15, execution of our pose estimation process, starting from multiple distinct initial poses, results in unique pose estimates, each of which can be regarded as a locally optimal result. This is due to the highly non-convex nature of our problem, a well-known fact for systems of rigid-bodies in general. These visual examples also demonstrate the importance of having multiple initial poses to avoid getting trapped into local optimal points that are far from the ground-truth point.

With vs. Without the Learned Metric To examine the usefulness of our learned internal metric, a special variant of our approach without this component is engaged here, which is also referred to as *Lie-X* w/o learned metric. Provided with multiple output pose candidates, this variant differs from our full-fledged approach by averaging over them for each of the joints in the 3D Euclidean space, instead of scoring them with the learned metric to pick up the best estimate. Empirical experiments such as those presented in Table 1 for fish and mouse objects and Table 2 for hand objects suggest a noticeable performance degradation when without the learned metric. Clearly the learned internal metric does facilitate in selecting from a global viewpoint the final estimate, which is obtained from the pool of locally optimal candidates using multiple initial poses. It has also been demonstrates in Figs. 13, 14, and 15 that in our context a max operation (i.e. with the learned metric) may well outperform an average operation (i.e. w/o learned metric). In particular, visually our learned internal metric is capable of producing predicted error scores that are nicely aligned with the *true* average joint error when we have access to the ground-truth.

Computational Efficiency All experiments discussed in this paper are performed on a desktop PC with an Intel i7-960 CPU and with 24Gb memory. At this moment, our CPU implementation achieves an average run-time speed of 83 FPS, 267 FPS, and 123 FPS for fish, mouse, and hand tasks, respectively. Table 3 summarizes the run-time speed comparisons with state-of-the-art hand pose estimators on the NYU hand dataset [49]. Our result of 123 FPS is obtained with only CPU access, nevertheless it is still comparable with most of these

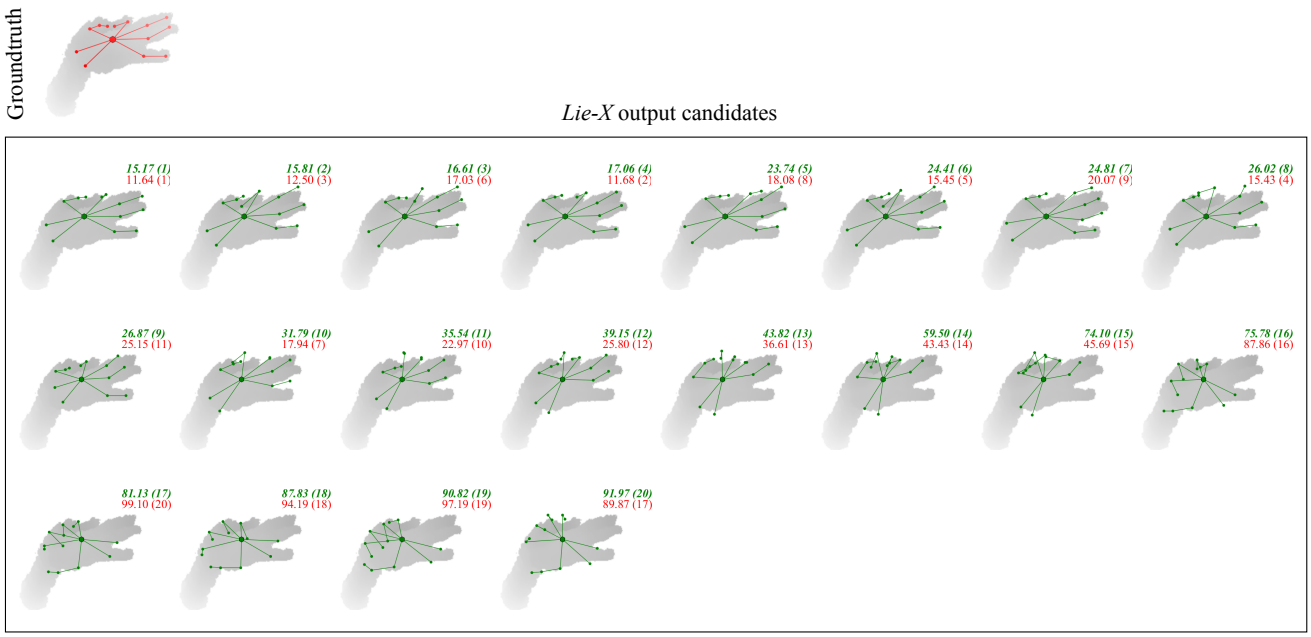


Fig. 15: An example of hand pose estimation that visually illustrates the inner-working of the learned internal metric in our approach applied onto the $K_t = 20$ pose candidates. Here green colored numbers present the scores (lower the better here) and ranking results obtained by applying the learned metric, red colored numbers denote the corresponding actual evaluation scores and ranking results by engaging the empirical evaluation metric of average joint error when we have access to the ground-truth annotations. See text for details.

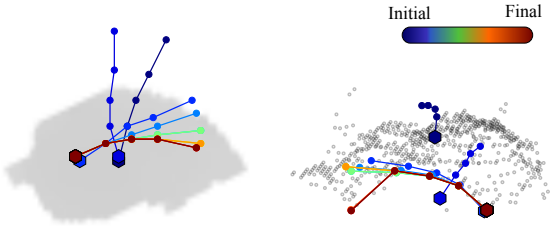


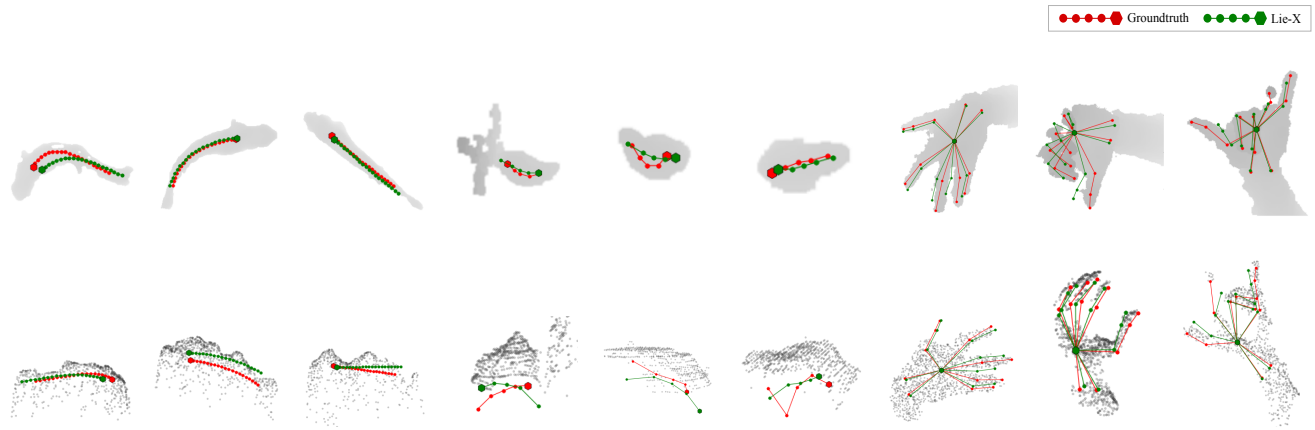
Fig. 16: Illustrating the convergence process on the same mouse example presented in Fig. 14. It starts from an initial pose candidate to the final pose estimation result, which is the top-left one among the list of all 40 output candidates.

recent methods which are based on GPUs. Note the empirical runtime speed of our approach could be further improved by exploiting the computing power of modern GPUs. Meanwhile, an exceptionally high speed method is developed in Oberweger et. al. [34], which is made possible by the usage of very shallow neural nets. This however comes with degraded performance as shown in Table 2, with a significant average joint error increase of 5.41 mm when compared to our approach.

Common Pose Estimation Errors of Our Approach Although our *Lie-X* approach performs relatively well in practical pose estimation settings, inevitably it does make mistakes in practical situations. These common errors include the following ones: orientation flips, displacement along the z-direction and sub-optimal shape fitting. A visual illustration of these common errors made by our approach is provided in Fig. 17. As can be observed, usually our Lie-Fish results are best aligned with the ground-truths. The mistakes of Lie-Mouse are more noticeable. Meanwhile the visual displacements of our Lie-Hand results from the ground-truths are most significant. This is to be expected, as the corresponding complexity levels of the three tasks varies from being relatively simple (i.e. kinematic chains) to complex (i.e. kinematic trees).

5.3 Tracking of Mouse

Our *Lie-X* approach is also examined on the tracking task using the mouse tracking dataset described beforehand. Compared with our single frame based pose estimation of Alg. 1, it is of interest to examine on how much we can gain from our tracking algorithm of Alg. 3, when temporal information is available. Empirically our mouse tracker is shown to produce an improved per-



(a) Fish (b) Mouse (c) Hand

Fig. 17: Visual examples of common pose estimation errors made by our *Lie-X* approach. These errors include orientation flips, displacement along the z-direction and sub-optimal shape fits. Each of the columns presents an exemplar depth image of fish, mouse, and hand, respectively, while the first and second rows display its top and side views.

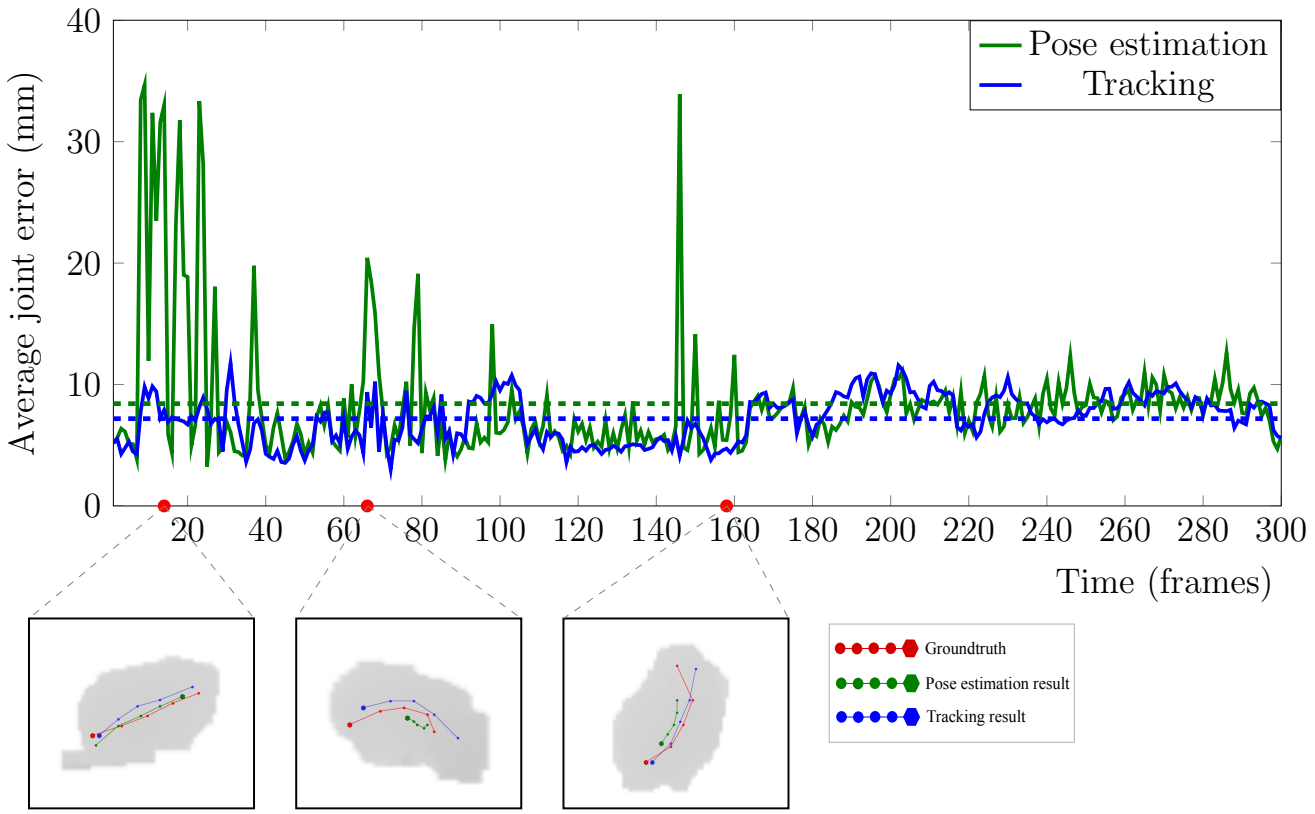
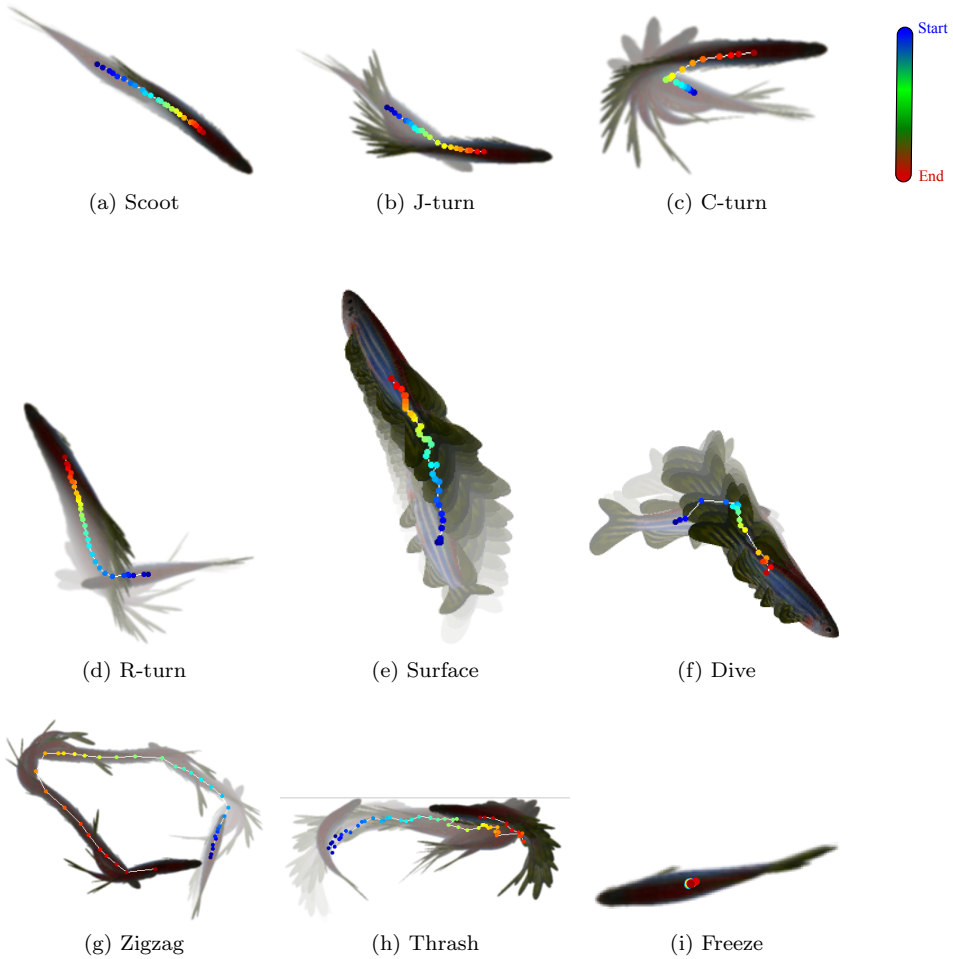


Fig. 18: Pose estimation vs. tracking: An comparison of the average joint error on frames of a mouse test sequence when employing our pose estimation (Alg. 1) vs. tracking (Alg. 3) modules. The horizontal dotted lines in green and blue colors are the respective mean errors of pose estimation and tracking results. Visual comparisons at various time frames are presented in the bottom row.

formance of 7.19 mm from the 8.42 mm results from our pose estimator on single frames. This can also be observed from the bottom row of Fig. 18 where visual comparisons are provides at several time frames. By ex-



(a) Scoot (b) J-turn (c) C-turn

(d) R-turn (e) Surface (f) Dive

(g) Zigzag (h) Thrash (i) Freeze

Fig. 19: Key frames from the nine distinct fish action categories considered in our experiments. The colored dots display the trajectory of the fish motions, where blue and red mark the start and end of the action respectively. Note for a better illustration of the distinct fish action categories, (e) and (f) present a side view of the *surface* and *dive* actions, while a top view is adopted for the rest action types.

ploiting temporal information, the results of our tracker are shown to produce less dramatic mistakes comparing to that of pose estimation. It is again evidenced quantitatively in Fig. 18, which presents a frame-by-frame average joint error comparison of tracking vs. pose estimation on a test sequence. Clearly there exists a number of very noisy predictions of pose estimation. In comparison our tracking results are in general much less noisy. Overall, the tracking results outperforms post estimation with a noticeable gap of at least 1 mm. Note the tracking results in some frames are slightly inferior to that of the pose estimation counterpart, which we attribute to the utilization of the averaging operations in our tracker.

5.4 Action Recognition of fish

To demonstrate the application of our approach on action recognition tasks, in what follows we conduct experiments on the aforementioned fish action dataset. In addition to the proposed tangent vector (i.e. Lie algebras) based feature representation, as comparison we also consider a joint based feature representation. Here the main body of the feature representation follows exactly as in the tangent vector representation, including e.g. the adoption of a temporal pyramid of $\{4, 2, 1\}$, with the only change as follows: Instead of tangent vectors, the corresponding the 3D joint positions are employed. This finally leads to an 888-dim feature vector representation.

Fig. 19 displays our fish dataset that contains nine unique action categories. The standard evaluation met-

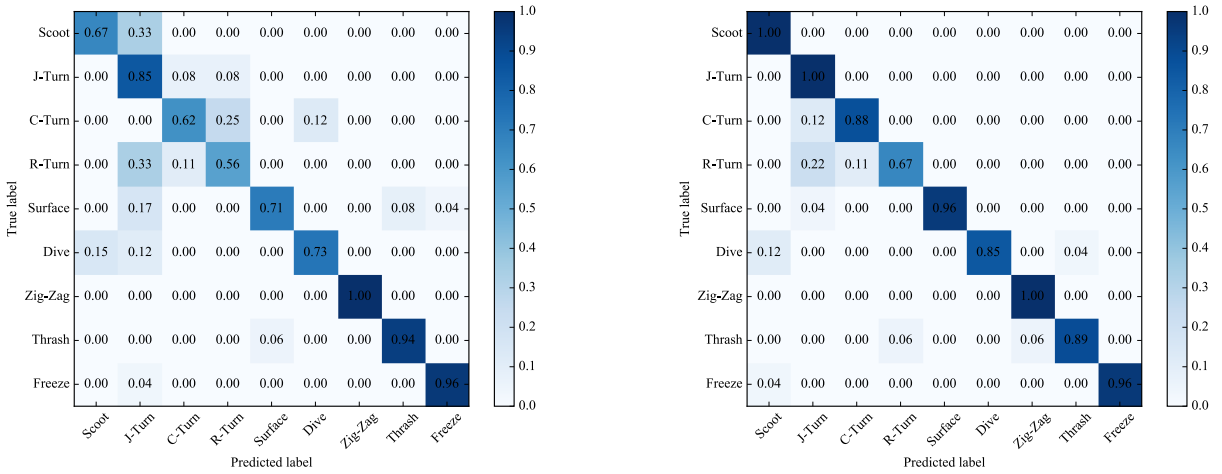


Fig. 20: Action recognition confusion matrices on our fish action dataset. (a) is for joint position based features, while (b) is for tangent vector based features. Their overall performance in terms of average classification accuracy for (a) and (b) is 79.19% and 91.33%, respectively.

ric of average classification accuracy are considered in this context. Empirically the comparison method that utilizing joint position features achieves a performance of 79.19%, which is significantly outperformed by our approach based on tangent vector features with the average accuracy of 91.33%. Fig. 20 provides further information of category-wise errors in the form of the confusion matrices. It is observed that the joint based method tends to confuse among the subset of actions of *scoot*, *J-turn*, *c-turn*, and *r-turn*, which are indeed more challenging to be separated due to their inherent similarities. Nonetheless, the performance on this subset is dramatically improved in our approach with tangent vector based features. We hypothesize that by following the natural tangent vector representation, our approach gains the discriminative power to separate these otherwise troublesome action categories.

6 Conclusion and Future Work

A unified Lie group approach is proposed for the related key problems of pose estimation, tracking, and action recognition of diverse articulated objects from depth images. Empirically our approach is evaluated on human hand, fish and mouse datasets with very competitive performance. For future work, we plan to work with more diverse articulated objects such as human full body and wild animals, as well as their interactions with other articulated objects and background objects.

Acknowledgment

The project is partially supported by A*STAR JCO grants 1431AFG120 and 15302FG149. Mouse and fish images are acquired with the help of Zoe Bichler, James Stewart, Suresh Jesuthasan, and Adam Claridge-Chang. Zilong Wang helps with the annotation of mouse data, while Wei Gao and Ashwin Nanjappa help with implementing the mouse baseline method.

References

1. Agarwal, A., Triggs, B.: Recovering 3D human pose from monocular images. IEEE Trans. PAMI **28**(1) (2006)
2. Ali, K., Fleuret, F., Hasler, D., Fua, P.: Joint pose estimator and feature learning for object detection. In: ICCV (2009)
3. Altafini, C.: Nonlinear Control in Year 2000, chap. The De Casteljau algorithm on SE(3), pp. 1–12. Springer-Verlag (2000)
4. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: CVPR (2008)
5. Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M., Pfau, D., Schaul, T., Shillingford, B., de Freitas, N.: Learning to learn by gradient descent by gradient descent. Arxiv pp. 1–50 (2016)
6. Arnol'd, V.I.: Mathematical methods of classical mechanics. Springer (2013)
7. Ballan, L., Taneja, A., Gall, J., Gool, L.V., Pollefeys, M.: Motion capture of hands in action using discriminative salient points. In: ECCV (2012)
8. Barsoum, E.: Articulated hand pose estimation review. Arxiv pp. 1–50 (2016)

9. Bookstein, F.: The study of shape transformation after D'Arcy Thompson. Mathematical Biosciences **34**(3–4), 177–219 (1977)
10. Bourdev, L., Malik, J.: Poselets: Body part detectors trained using 3D human pose annotations. In: ICCV (2009)
11. Branson, K., Belongie, S.: Tracking multiple mouse contours (without too many samples). In: CVPR (2005)
12. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
13. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: ICML (2005)
14. Chen, L., Wei, H., Ferryman, J.: A survey on model based approaches for 2D and 3D visual human pose recovery. PRL **34**(15), 1995–2006 (2013)
15. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: IEEE Workshop on PETS (2005)
16. Dollar, P., Welinder, P., Perona, P.: Cascaded pose regression. In: CVPR (2010)
17. Felzenszwalb, P., Huttenlocher, D.: Pictorial structures for object recognition. International Journal of Computer Vision **61**(1), 55–79 (2005)
18. Fleuret, F., Geman, D.: Stationary features and cat detection. JMLR **9**, 2549–2578 (2008)
19. Gall, J., Yao, A., Razavi, N., van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. IEEE Trans. PAMI **33**(11), 2188–2202 (2011)
20. Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., Navab, N.: Dominant orientation templates for real-time detection of textureless objects. In: CVPR (2010)
21. Hough, P.: Machine analysis of bubble chamber pictures. In: Proc. Int. Conf. High Energy Accelerators and Instrumentation (1959)
22. Hsu, E.P.: Stochastic Analysis on Manifolds. AMS Press (2002)
23. Huang, C., Allain, B., Franco, J., Navab, N., Boyer, E.: Volumetric 3D tracking by detection. In: CVPR (2016)
24. Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. International journal of computer vision **29**(1), 5–28 (1998)
25. Kalueff, A., Gebhardt, M., Stewart, A., Cachat, J., Brimmer, M., Chawla, J., Craddock, C., Kyzar, E., Roth, A., Landsman, S., Gaikwad, S., Robinson, K., Baatrup, E., Tierney, K., Shamchuk, A., Norton, W., Miller, N., Nicolson, T., Braubach, O., Gilman, C., Pittman, J., Rosemberg, D., Gerlai, R., Echevarria, D., Lamb, E., Neuhauss, S., Weng, W., Bally-Cuif, L., Schneider, H.: Towards a comprehensive catalog of zebrafish behavior 1.0 and beyond. Zebrafish **10**(1), 70–86 (2013)
26. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
27. Lee, J.: Introduction to Smooth Manifolds. Springer (2003)
28. Leibe, B., Leonardis, A., Schiele, B.: Combined object categorization and segmentation with an implicit shape model. In: In ECCV workshop on statistical learning in computer vision, pp. 17–32 (2004)
29. Mahasseni, B., Todorovic, S.: Regularizing long short term memory with 3D human-skeleton sequences for action recognition. In: CVPR (2016)
30. Manton, J.: A primer on stochastic differential geometry for signal processing. IEEE J. Sel. Topics Signal Processing **7**(4), 681–99 (2013)
31. Mikic, I., Trivedi, M.M., Hunter, E., Cosman, P.C.: Human body model acquisition and tracking using voxel data. International Journal of Computer Vision **53**(3), 199–223 (2003)
32. Murray, R., Sastry, S., Li, Z.: A Mathematical Introduction to Robotic Manipulation. CRC Press (1994)
33. Nie, X., Xiong, C., Zhu, S.: Joint action recognition and pose estimation from video. In: CVPR (2015)
34. Oberweger, M., Wohlhart, P., Lepetit, V.: Hands deep in deep learning for hand pose estimation. In: Computer Vision Winter Workshop (2015)
35. Oberweger, M., Wohlhart, P., Lepetit, V.: Training a feedback loop for hand pose estimation. In: ICCV (2015)
36. Oikonomidis, N., Argyros, A.: Efficient model-based 3D tracking of hand articulations using Kinect. In: BMVC (2011)
37. Perez-Sala, X., Escalera, S., Angulo, C., Gonzalez, J.: survey of human motion analysis using depth imagery. Sensors **14**, 4189–4210 (2014)
38. Poppe, R.: Vision-based human motion analysis: An overview. Comput. Vis. Image Underst. **108**(1–2), 4–18 (2007)
39. Procesi, C.: Lie groups: An approach through invariants and representations. Springer (2007)
40. Qian, C., Sun, X., Wei, Y., Tang, X., Sun, J.: Realtime and robust hand tracking from depth. In: CVPR (2014)
41. Rahmani, H., Mian, A.: 3D action recognition from novel viewpoints. In: CVPR (2016)
42. Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., Blake, A.: Efficient human pose estimation from single depth images. IEEE TPAMI **35**(12), 2821–40 (2013)
43. Sinha, A., Choi, C., Ramani, K.: Deephand: Robust hand pose estimation by completing a matrix imputed with deep features. In: CVPR (2016)
44. Srivastava, A., Turaga, P., Kurtek, S.: On advances in differential-geometric approaches for 2D and 3D shape analyses and activity recognition. Image Vision Comput. **30**(6-7), 398–416 (2012)
45. Sun, X., Wei, Y., Liang, S., Tang, X., Sun, J.: Cascaded hand pose regression. In: CVPR (2015)
46. Tan, D., Cashman, T., Taylor, J., Fitzgibbon, A., Tarlow, D., Khamis, S., Izadi, S., Shotton, J.: Fits like a glove: Rapid and reliable hand shape personalization. In: CVPR (2016)
47. Tang, D., Taylor, J., Kohli, P., Keskin, C., Kim, T., Shotton, J.: Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In: ICCV (2015)
48. Tompson, J., Jain, A., LeCun, Y., Bregler, C.: Joint training of a convolutional network and a graphical model for human pose estimation. In: NIPS (2014)
49. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. SIGGRAPH (2014)
50. Tuzel, O., Porikli, F., Meer, P.: Learning on Lie groups for invariant detection and tracking. In: CVPR (2008)
51. Vemulapalli, R., Arrate, F., Chellappa, R.: human action recognition by representing 3D skeletons as points in a Lie group. In: CVPR (2014)
52. Vemulapalli, R., Chellappa, R.: Rolling rotations for recognizing human actions from 3D skeletal data. In: CVPR (2016)
53. Wiltschko, A., Johnson, M., Iurilli, G., Peterson, R., Katon, J., Pashkovski, S., Abraira, V., Adams, R., Datta, S.: Mapping sub-second structure in mouse behavior. Neuron **88**(6), 1121–35 (2015)
54. Xiong, X., la Torre, F.D.: Supervised descent method and its applications to face alignment. In: CVPR (2013)
55. Xu, C., Cheng, L.: Efficient hand pose estimation from a single depth image. In: ICCV (2013)

56. Xu, C., Nanjappa, A., Zhang, X., Cheng, L.: Estimate hand poses efficiently from single depth images. International Journal of Computer Vision pp. 1–25 (2015)
57. Yang, Y., Ramanan, D.: Articulated pose estimation with flexible mixtures-of-parts. In: CVPR (2011)
58. Zhou, X., Wan, Q., Zhang, W., Xue, X., Wei, Y.: Model-based deep hand pose estimation. In: IJCAI (2016)