DART: Dense Articulated Real-Time Tracking

Tanner Schmidt, Richard Newcombe, Dieter Fox
University of Washington
Computer Science & Engineering
Seattle, Washington

{tws10, newcombe, fox}@cs.washington.edu

***Abstract*—This paper introduces DART, a general framework for tracking articulated objects composed of rigid bodies connected through a kinematic tree. DART covers a broad set of objects encountered in indoor environments, including furniture and tools, and human and robot bodies, hands and manipulators. To achieve efficient and robust tracking, DART extends the signed distance function representation to articulated objects and takes full advantage of highly parallel GPU algorithms for data association and pose optimization. We demonstrate the capabilities of DART on different types of objects that have each required dedicated tracking techniques in the past.**

I. Introduction

The ability to accurately track the pose of objects in real time is of fundamental importance to many areas of robotics. Applications range from navigation to planning, manipulation and human-robot interaction, all of which have received the attention of researchers working within a state-space model-based paradigm within both computer vision and robotics. The class of objects that can be described as collections of rigid bodies chained together through a kinematic tree is quite broad, including furniture, tools, human bodies, human hands, and robot manipulators. Tracking articulated bodies from a single viewpoint and without instrumenting the object of interest still presents a significant challenge where the single viewpoint and occlusions, including self-occlusion, limit the amount of information available for pose estimation. Noisy sensor data and approximate object models pose additional problems. Finally, the objects being tracked can be highly dynamic and have many degrees of freedom, making real-time tracking difficult.

Early articulated model-based tracking techniques relied on tracking 2D features such as image edges on a CPU [8, 5]. Recently introduced depth cameras along with highly parallel algorithms optimized for modern GPUs have enabled new algorithms for tracking complex 3D objects in real time. Examples include KinectFusion and related efforts for 3D mapping [23, 16, 34], human body pose tracking [29, 35, 15], articulated hand tracking [24, 19, 26]. These approaches were developed for specific application domains and have not been demonstrated or tested on multiple tracking applications. The application-specific nature of these approaches enables their authors to show excellent performance by taking advantage of domain-specific features and constraints, but it also prevents them from serving as general tools for tracking arbitrary articulated objects. Techniques have also been developed to track fully non-rigid deformations of an underlying surface template for both specific [21] and general [14, 20, 28] object cases. However, the full generality of these models comes at the cost of increased model complexity, and for many objects that are well modelled as piecewise rigid bodies, such overparameterized output obscures the utility of tracking the articulated body state directly.

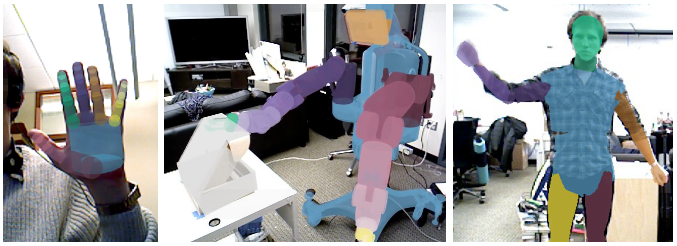


Fig. 1: Articulated objects successfully tracked at frame-rate using DART, without object-specific tuning of the algorithm: a human hand, Rethink Robotics Baxter robot opening a box, and a full human body.

In this paper we present DART, a general framework for tracking articulated models with formally defined kinematic and geometric structure using standard depth sensors. DART represents objects via an articulated version of the signed distance function (SDF), which has been used to achieve very robust and efficient results for online 3D mapping [23, 16, 34] and for tracking rigid objects in six degrees of freedom [10, 27, 22, 4, 3]. Our tracking framework uses a local gradient based approach to find the pose of the model which best explains the data points observed in a depth frame along with a prior based on previous data. The objective function is trivially parallelizable, and is optimized on a GPU, which allows us to use all available data to achieve accurate tracking results in real-time, even with models having as many as 48 degrees of freedom. To use DART in a new domain, one need only supply a model file specifying the relative locations of all the frames of reference, the possible articulations of the frames, and the geometry attached to each frame. Given this model, the object can be tracked with no changes to the code. These models can be designed manually, generated from a currently available volumetric fusion method, or derived from CAD or robot description models.

The main contributions of this paper are (1) a general framework for highly efficient tracking of articulated objects, (2) a generalization of signed distance functions (SDF) to articulated objects, (3) a symmetric formulation of point set registration that incorporates negative information into the energy function, and (4) a demonstration of the framework tracking a variety of different models.

This paper is organized as follows. After discussing related work, we will introduce the DART framework in Section III. Experimental results are described in Section IV followed by a discussion of the limitations and possible extensions to DART in Section V. We conclude in Section VI.

II. Related Work

Existing approaches to articulated model tracking tend to fall into one of three categories. *Gradient-based methods*, the category to which our approach belongs, involve the minimization of an objective function which is characterized by having partial derivatives with respect to the pose vector. These approaches are therefore able to take advantage of well-studied optimization techniques that make use of derivative information to incrementally improve an estimate of the optimum. Implicit surfaces and gradient descent on ICP-like error terms have proven successful in tracking rigid bodies, as by Newcombe et al. [23], Sturm et al. [30], and Henry et al. [16]. Implicit surfaces have also been used for model-based tracking of articulated models; Dewaele et al. [6] used an implicitly defined model and an expectation maximization approach to track human hands. Grest et al. [13] presented early work on human body tracking from depth, but subsampled the points and still did not reach real-time. Schulman et al. [28] track fully deformable models with RGB-D cameras. They use the aid of a state of the art physics simulator to reason about configurations of deformable objects, but require foreground/background segmentation as a preprocessing step and only report results using either distinctively colored objects or multiple cameras. They furthermore rely on downsampling the input data in order to achieve tracking results at 10 Hz. In contrast, our framework introduces articulated and symmetric signed distance functions to solve the data association problem robustly and efficiently, enabling real-time tracking using all input data points. Another approach similar to ours is that of Ganapathi et al. [12], which uses range data to track articulated motion of human bodies by augmenting traditional ICP with free space information. Our work differs by incorporating information about observed free space directly into the energy function, rather than using constraint projection. We also demonstrate the applicability of our approach to a wider set of application domains and demonstrate how the trivially parallelizable optimization can take advantage of GPU acceleration. Finally, Ye and Yang [35] perform a gradient based optimization over a rigged mesh model for human body tracking, but again rely on subsampling of both the model and observation point sets, and do not take free space information into account.

Sampling-based methods, on the other hand, rely on objective functions that are cheap to evaluate, but have partial derivatives which are either too expensive to compute or which do not give reliable information about the direction of the true minimum. For example, many articulated model tracking approaches in this category utilize some form of silhouette information, necessitating the use of indicator functions which are discontinuous and therefore not globally differentiable. Such methods thus often rely on inference techniques which are less thoroughly studied and perhaps less theoretically justified, such as particle swarm optimization (PSO) as employed by Oikonomidis et al. [24], Kyriazis and Argyros [19].

Finally, *discriminative or appearance-based methods* utilize training sets and machine learning techniques to learn a direct map from the features extracted from input images to specific articulated part labels and poses. Discriminative approaches have proven extremely successful for the problem of human pose estimation [29, 31], and similar approaches have also been applied to hand tracking [17]. While these methods can be highly efficient and robust, they require a lot of data and time to train recognition models for each new object. However, these approaches could also be seen as complementary to ours, as they are able to provide pose estimates with little or no prior. A combination of discriminative and generative approaches has been applied to articulated tracking, e.g. by Ballan et al. [1], who used fingernail detectors to help constrain a gradient descent optimization, by Ganapathi et al. [11], who used a gradient-based optimizer in conjunction with body part recognition for human body tracking, and by Qian et al. [26], who combine PSO, a standard ICP formulation, and a hand-designed finger detector to track human hands, thus straddling all three categories.

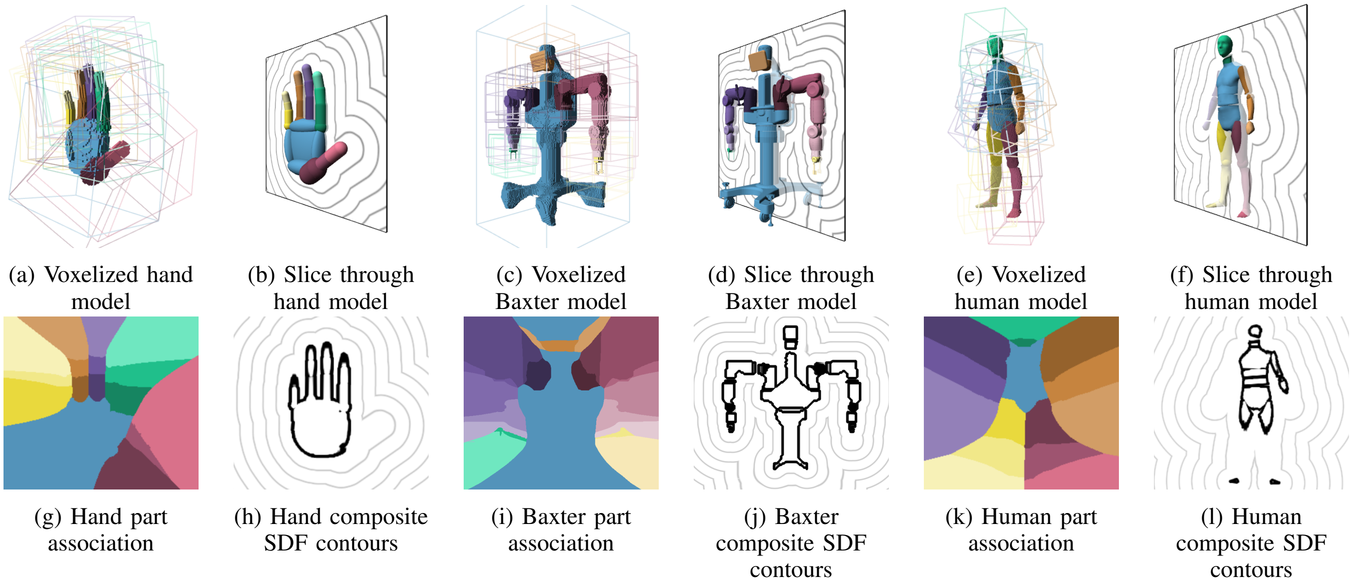
While many of these existing techniques achieve excellent results in their application domain, such as hand/object tracking [24, 19, 26], human pose tracking [29, 31, 11, 12, 35], or deformable object tracking [28], none of them have been demonstrated to work in several domains. In contrast, our DART framework relies on a highly efficient representation and optimization to operate under very general conditions, thereby enabling it to track a wide variety of objects.

III. DART: Dense Articulated Real-Time Tracking

We now present our tracking framework. While DART is able to track multiple articulated objects moving in a scene, we present the framework for a single object only, in order to avoid overly complicated notation. The extension to multiple objects is straightforward.

A. Model Representation

The models we track are defined as a set of rigid bodies connected to each other in a kinematic tree. By convention, we designate the root of the tree as frame 0. The root frame is related to the camera frame of reference by the rigid body transformation matrix $T_{0,c} = [R_{0,c} | t_{0,c}] \in \mathbb{SE}_3$. Every other frame in the kinematic tree is attached to some other frame which is designated as its parent. The frame i (for $i \neq 0$) is defined relative to its parent via the transform $T_{parent(i),i}(\theta) \in \mathbb{SE}_3$, which may be a function of any subset of joint parameters θ (which collectively describe the articulated pose of the entire model) together with any fixed transformation defined relatively between the frames. The



(a) Voxelized hand model

(b) Slice through hand model

(c) Voxelized Baxter model

(d) Slice through Baxter model

(e) Voxelized human model

(f) Slice through human model

(g) Hand part association

(h) Hand composite SDF contours

(i) Baxter part association

(j) Baxter composite SDF contours

(k) Human part association

(l) Human composite SDF contours

Fig. 2: Representation of articulated models in DART. A model specification is given in the form of a kinematic tree of joints connecting rigid body geometric parts or segments. Each rigid part is voxelized, and a signed distance function is computed in the local coordinate frame. (a),(c), and (e) show the bounding boxes of the rigid parts (each color is a separate part) as well as the voxels which lie within the geometry surface. Given a specific model state, the global SDF for a point in the global frame is found by taking the minimal SDF at that point over all overlapping piecewise SDF functions. (b),(d), and (f) show a planar slice through the models with the isocontours of the SDF superimposed on the plane. (g),(i), and (k) show the colors representing the rigid parts from which the SDF values were generated for each pixel on the plane, and (h),(j), and (l) again show the isocontours of the SDF, with the zero level set shown in bold.

transformation from frame i to any other frame j is then defined recursively using:

$$T_{j,i}(\theta) = T_{j,parent(i)}(\theta) T_{parent(i),i}(\theta) \ , \tag{1}$$

such that the transform between any two frames is given by composing the transforms in a chain indexed using the $parent()$ function. Note that we assume a tree structure such that there is a unique path between any two frames.

Rigidly attached to each frame may also be some geometry. Geometry attached to frame k is defined implicitly by a signed distance function $\text{SDF}^k(\mathbf{x})$, which takes on negative values inside the geometry, positive values outside, and has a value of zero at the surface interface. While the SDF is in principle defined for all $\mathbf{x} \in \mathbb{R}^3$, discretizations of signed distance functions are, in practice, sampled within finite bounds derived from application-specific requirements and practical computational and memory restrictions, as seen in figure (2a). In our experiments, these functions are either generated from an analytical function describing primitive shapes such as cylinders and ellipsoids, or by voxelizing closed polygonal meshes followed by computing a sampled Euclidean SDF using an efficient 3D distance transform algorithm [9]. Alternatively, the SDF functions could easily be generated from real-world objects for which no model is readily available using an efficient 3D scanning method [23] [30] [16].

B. *DART_Tracker*

To track an articulated object over time, DART uses an extended Kalman filter (EKF) style approach [32], where the measurement update is achieved via optimization rather than linearization.

```
**DART_Tracker**($\mu^{t-1}, \Sigma^{t-1}, c^t, D^t$):
    $\bar{\mu}^t = g(c^t, \mu^{t-1})$
    $\bar{\Sigma}^t = G^t \ \Sigma^{t-1} \ G^{tT} + R^t$
    $\mu^t = \arg\min_\theta \left(-\log(p(D^t|\theta)) + (\theta - \bar{\mu}^t)(\bar{\Sigma}^t)^{-1}(\theta - \bar{\mu}^t)\right)$
    $\Sigma^t = H(\mu^t)^{-1} + \hat{\Sigma}$
    return $\mu^t, \Sigma^t$
```

TABLE I: The DART tracking algorithm.

The DART_Tracker estimates the mean, μ^t, and covariance, Σ^t, of the posterior over the state θ using the algorithm shown in Table I. The inputs to the algorithm are the state mean and covariance from the previous time step along with the most recent control information, c^t, and a depth map, D^t. Depending on the application, c^t may provide information about the motion of the tracked object, e.g. via a robot manipulator's internal sensors. Just like in the standard EKF, the predictive mean, $\bar{\mu}^t$, and covariance, $\bar{\Sigma}^t$, are computed via linearization of a motion model, $g(c^t, \mu^{t-1})$, where G^t and R^t are the Jacobian and the additive noise covariance, respectively. While a well-designed motion model can significantly improve the performance of a tracker, object-specific motion models are not the focus of this paper and we employ a constant position model in our current implementation.

The crucial component of DART_Tracker is the correction of the estimate based on the observed depth map: Rather than computing the Kalman gain to update the state estimate, the algorithm performs an optimization that determines the object state that minimizes the negative log likelihood of the observed depth map, while directly incorporating the predicted mean and covariance via a squared penalty term (Step 3). The optimization also provides the covariance of the estimate via the inverse Hessian matrix $H(\mu^t)^{-1}$ computed at the solution. However, the Hessian approximation tends to underestimate the uncertainty, since it assumes independence between all pixels in a depth map and does not consider possible errors in the data association. To deal with this problem in practice, our current algorithm adds a constant noise term, $\tilde{\Sigma}$, as shown in Step 4.

In the remainder of this section, we will describe how to efficiently perform the optimization required for Steps 3 and 4 of the DART_Tracker. We will focus on minimizing the negative of the measurement log likelihood $p(D^t|\theta)$. Given this result, incorporation of the predictive mean and covariance is then achieved simply by factoring the quadratic term $(\theta - \bar{\mu}^t)(\bar{\Sigma}^t)^{-1}(\theta - \bar{\mu}^t)$ into the optimization.

C. Measurement Model and Data Association

We work with an input device capable of producing a depth map $D(\mathbf{u})$, which defines for each pixel $\mathbf{u} = (u, v) \in \Omega$ a depth value $d \in \mathbb{R}$, where $\Omega \subset \mathbb{R}^2$ is the image plane. For each pixel $\mathbf{u}$ in an observed depth map, we back-project the value to its corresponding camera frame point $\mathbf{x_u} \in \mathbb{R}^3$:

$$\mathbf{x_u} = D\left(\mathbf{u}\right)\mathbf{K}^{-1}\left(u, v, 1\right)^{\top} \quad , \tag{2}$$

where $\mathbf{K}$ defines the depth camera intrinsic calibration matrix with known focal length and principle point parameters.

Specifically, our measurement model assumes that depth frames are generated from some distribution $p(D|\theta;\mathcal{M})$, where $\mathcal{M}$ is the given geometry and kinematic structure of our model, $\theta \in \mathbb{R}^N$ is the vector describing some particular pose of the model, and N is the number of degrees of freedom in the model. For the remainder of this section we will omit $\mathcal{M}$ for clarity, as the model is assumed to be constant.

We make the assumption that the probability of generating the depth value at each pixel $\mathbf{u}$ of the depth map is independent of the value at all other pixels given the pose vector, allowing us to express the likelihood as follows:

$$p(D|\theta) = \prod_{\mathbf{u}\in\Omega} p\left(D(\mathbf{u})|\theta\right) \ . \tag{3}$$

In theory, our depth observation at each pixel is the true depth $\hat{D}\left(\mathbf{u}\right)$ along the pixel ray, corrupted by Gaussian noise. This defines the per-pixel depth measurement likelihood as

$$p(D(\mathbf{u})|\hat{D}(\mathbf{u})) \propto \exp(-(e(\mathbf{u}))^2/\sigma^2) \ , \tag{4}$$

where $e(\mathbf{u}) = D(\mathbf{u}) - \hat{D}(\mathbf{u})$ is the projective signed distance error between the noisy observation and true surface point. In practice, we treat observed points as if they are generated by the exterior surface of the model with a Gaussian noise distribution in three dimensions rather than one dimensional noise in depth. Instead of the projective error described above, with this simplification our error is found simply by looking up the back-projected point at pixel $\mathbf{u}$ in the model signed distance function, $\text{SDF}_{\text{mod}}\left(\mathbf{x};\theta\right) : \mathbb{R}^3 \mapsto \mathbb{R}$, which defines for all of $\mathbb{R}^3$ the distance to the surface of the model when articulated according to pose parameters θ. We replace the projective signed distance error $e(\mathbf{u})$ with the true signed distance error and the observation likelihood for the articulated model becomes:

$$p(D|\theta) \propto \prod_{\mathbf{u}\in\Omega} \exp(-\text{SDF}_{\text{mod}}\left(\mathbf{x_u};\theta\right)^2/\sigma^2)) \ . \tag{5}$$

The MLE estimate of θ is then given by minimizing the negative log of (5):

$$\hat{\theta} = \arg\min_{\theta} \sum_{\mathbf{u}\in\Omega} ||\text{SDF}_{\text{mod}}\left(\mathbf{x_u};\theta\right)||^2 \ . \tag{6}$$

The observation model introduced here is the articulated body counterpart to the rigid body tracking approach introduced in [10]. Fitzgibbon originally demonstrated the efficiency with which a non-linear gradient descent style optimization could be used to solve a rigid body tracking version of (6). The resulting tracker replaced the need in traditional ICP and its variants to define an explicit data association step between the observed and model surfaces, instead utilizing the implicit data association obtained through a precomputed distance transform of the model as described above.

While only a single precomputed SDF suffices for rigid body tracking, the transition to the articulated model case results in a continuous space of signed distance functions, as a function of the state. This means that the global function $\text{SDF}_{\text{mod}}\left(\mathbf{x};\theta\right)$ can no longer be strictly precomputed. Recomputing a full global SDF at every iteration can be computationally expensive due to the numbers of articulated geometries in use.

Instead, we solve this problem by approximating the global signed distance function, $\text{SDF}_{\text{mod}}\left(\mathbf{x};\theta\right)$, by precomputing local signed distance functions, SDF^k, for each frame of reference k to which geometry is rigidly attached. We can then compute a signed distance value in the global frame by a composition of these local functions discussed below. In Figure (2) we illustrate a selection of global SDF compositions. For each model with a given state estimate, the labelled articulated parts are shown with a 2D slice through the resulting 3D SDF together with the associated closest part for each point in that plane of the volume.

Given a global point $\mathbf{x_u}$, defined in the camera frame of reference c, we obtain the associated frame k^* closest to $\mathbf{x_u}$ by iterating over the local SDF values:

$$k^* \quad = \quad \arg\min_{k\in\mathcal{M}} \text{abs}\left(\text{SDF}^k(T_{k,c}(\theta)\mathbf{x_u})\right) \tag{7}$$

resulting in the piecewise-defined signed distance function as:

$$e(\theta, \mathbf{u}) \quad = \quad \min_{k\in\mathcal{M}} \text{abs}\left(\text{SDF}^k(T_{k,c}(\theta)\mathbf{x_u})\right) \ . \tag{8}$$

Figures (3a) - (3e) illustrate the data association, SDF error, and gradients induced by the current observation in the composite model SDF.

Inserting (8) into (6), we arrive at the approximated articulated tracking energy,

$$\hat{\theta} = \arg\min_{\theta} \sum_{\mathbf{u}\in\Omega} e(\theta, \mathbf{u})^2 \ . \tag{9}$$

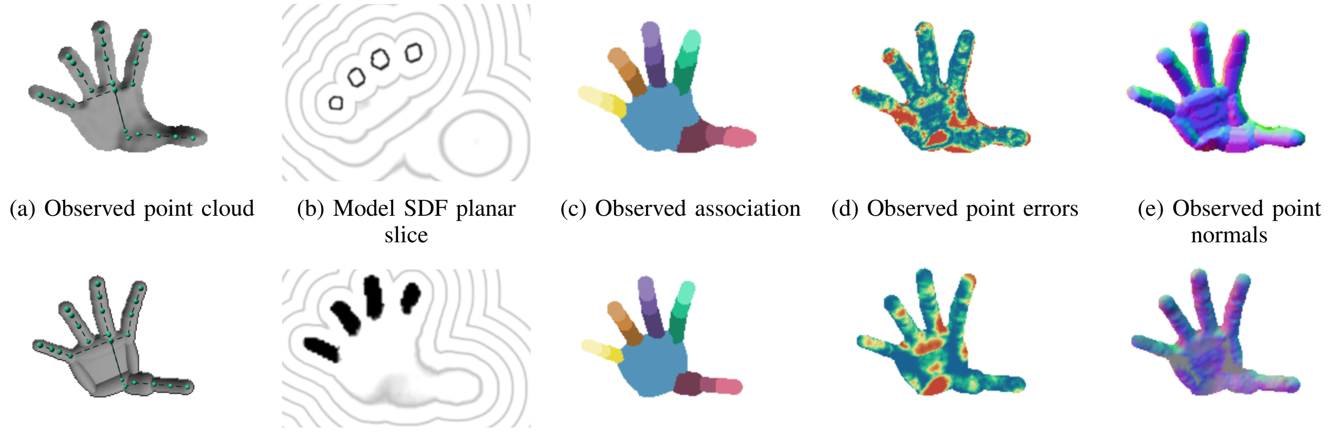

(a) Observed point cloud
(b) Model SDF planar slice
(c) Observed association
(d) Observed point errors
(e) Observed point normals
(f) Predicted point cloud
(g) Observation SDF planar slice
(h) Prediction association
(i) Predicted point errors
(j) Predicted point normals

Fig. 3: (a) and (f) show the observed and predicted point clouds for a given depth frame, respectively, overlaid with the model skeleton. (b) shows a planar slice of the model SDF, and (g) shows a slice of the observation SDF on the same plane (black indicates the zero level set). Each observed point in (a) is looked up in the model SDF (b) to determine the rigid part data association (c), error (d), and SDF gradients (e) which are used to compute derivatives of the energy function. Likewise, each predicted point in (f) is looked up in the observation SDF (g) to determine rigid part data association (h), error (i), and gradients (j).

D. *Optimization*

Given our energy function and some initial guess for θ, we perform a second order Taylor series expansion of (9) and apply the Gauss-Newton approximation to the Hessian term leading to the construction of the standard normal equations:

$$\Delta\theta = -\sum_{\mathbf{u}\in\Omega}\left(J(\mathbf{u},\theta)^\top J(\mathbf{u},\theta)\right)^{-1} e(\mathbf{u},\theta)J(\mathbf{u},\theta) \ , \tag{10}$$

where $J(\mathbf{u},\theta)=\frac{\partial}{\partial\Delta\theta}e(\mathbf{u},\theta\oplus\Delta\theta)$, and $\oplus$ defines the operator that composes the parameter update vector $\Delta\theta$, obtained by solving the normal equations. The updated state is therefore:

$$\theta \leftarrow \theta\oplus\Delta\theta \ . \tag{11}$$

We have used the composition operator rather than a simple addition so that we can handle non-linearities, as will soon become clear.

We will now discuss the computation of the Jacobian terms. In each iteration, we compute the implicit data association $k^*(\mathbf{u})$ and error $e(\mathbf{u},\theta)$ as given by equations (7) and (8). The computation of $J(\mathbf{u},\theta)$ breaks down nicely via the chain rule:

$$\begin{aligned}\frac{\partial}{\partial\Delta\theta}\mathrm{SDF}^{k^*}(T_{k^*,c}(\theta\oplus\Delta\theta)\mathbf{x_u}) &= \\ \nabla\mathrm{SDF}^{k^*}(T_{k^*,c}(\theta)\mathbf{x_u})&\frac{\partial}{\partial\Delta\theta}T_{k^*,c}(\theta\oplus\Delta\theta)\mathbf{x_u} \ ,\end{aligned} \tag{12}$$

where we simply need to compute a $3\times N$ matrix which describes the motion of point $\mathbf{x_u}$ in $\mathbb{R}^3$ in the frame of reference of k^* caused by incremental pose update $\Delta\theta$, then compute the dot product with the local SDF gradient at the current point location to get the $1\times N$ Jacobian contribution from pixel $\mathbf{u}$.

Given our model parameterization, the transform from the global frame c to k^* follows some chain of the form $T_{k^*,c}=T_{k^*,parent(k^*)}...T_{0,c}$. By convention the first 6 parameters describe the 6 degrees of freedom of the global transformation $T_{0,c}$. These are parameterized using the Lie algebra $\mathfrak{se}_3$, which has been shown to be an effective representation for gradient-based visual tracking techniques [8, 7]. This parameterization results in a matrix of the following form:

$$\frac{\partial}{\partial\Delta\theta}T_{k^*,c}(\theta\oplus\Delta\theta)\mathbf{x} = \left(\begin{array}{cccccc|ccc} 1 & 0 & 0 & 0 & \mathbf{x}_3 & -\mathbf{x}_2 & & & \\ 0 & 1 & 0 & -\mathbf{x}_3 & 0 & \mathbf{x}_1 & \frac{\partial\mathbf{x}}{\partial\Delta\theta_7} & \cdots & \frac{\partial\mathbf{x}}{\partial\Delta\theta_N} \\ 0 & 0 & 1 & \mathbf{x}_2 & -\mathbf{x}_1 & 0 & & & \end{array}\right) \tag{13}$$

where the first 3×6 block is the linearization of $\mathbb{SE}_3$ around the identity transform (because we are solving for an incremental update in (10), and the remainder of the matrix describes the motion with respect to individual joint parameters. To compute $\frac{\partial}{\partial\Delta\theta_i}T_{k^*,c}\mathbf{x_u}$ for the remaining columns, we apply the product rule and, because each element of the pose vector can affect at most one transform, we get at most one nonzero term of the form:

$$\frac{\partial}{\partial\Delta\theta_i}T_{k^*,c}\mathbf{x_u} = T_{k^*,parent(j)}\left(\frac{\partial}{\partial\Delta\theta_i}T_{parent(j),j}\right)T_{j,c}\mathbf{x_u} \tag{14}$$

where j is the frame of reference whose transformation (relative to its parent) depends on θ_i. If degree of freedom i is a rotational joint with some axis $\mathbf{z}$ in the frame of reference $parent(j)$, then this becomes:

$$\frac{\partial}{\partial\Delta\theta_i}T_{k^*,c}\mathbf{x_u} = T_{k^*,parent(j)}\left(\mathbf{z}\times T_{parent(j),c}\mathbf{x_u}\right) \tag{15}$$

where $T_{k^*,parent(j)}$ will apply only the rotational component of the transform from $parent(j)$ to k^*, as the result of the cross product will be homogenized as a vector. If i is instead a prismatic joint along some axis $\mathbf{z}$, also in the frame of reference of $parent(j)$, then we simply have:

$$\frac{\partial}{\partial \Delta\theta_i} T_{k^*,c}\mathbf{x_u} = T_{k^*,parent(j)}\mathbf{z} \tag{16}$$

where, again, only the rotational component of $T_{k^*,parent(j)}$ will be applied to vector $\mathbf{z}$. We note that the above two situations only apply when the position of frame k^* *depends* on parameter θ_i, which is to say that θ_i affects some transform between frame k^* and the root; if, on the other hand, θ_i appears below k^* or in another branch of the kinematic tree, then column i will be the zero vector.

Once we have computed the incremental update we must perform the composition of equation (11). As stated previously, $\theta_{1:6}$ describes the global 6DOF transformation between the camera and the model via the exponential map from se_3 to $\mathbb{SE}_3$. Given that we've solved for a *relative* transformation about our current estimate, we update this parameter block by composing the transformations and taking the log map back to se_3. The composition of the remainder of the parameter vector can be done with simple vector addition:

$$\theta_{7:N} \leftarrow \theta_{7:N} + \Delta\theta_{7:N} \ . \tag{17}$$

Crucially, our dense, fully-articulated model tracking algorithm described here trivially parallelizes onto modern GPGPU hardware.

E. Free Space Constraints and the Observation SDF

Traditional ICP-based approaches neglect some information available in depth observations; namely, each pixel in a depth map provides information not only about where surfaces exist in the world, but also where there are none. Any (non-corrupted) observation of a nonzero depth indicates that there is nothing between that observed point and the camera, up to some noise threshold and barring sensor error. The usefulness of this 'negative' information was noted by Ganapathi et al. [12], who used constraint projection to avoid gross free space violation.

We compute a signed distance function representation of each observation, $\mathrm{SDF}_{\mathrm{obs}}(\mathbf{x};D)$, and add a simple complementary term to the model SDF energy described in the previous section, which we call the observation SDF term. We are then able to take negative information into account directly in the optimization, rather than relying on constraint projection. The result is a nicely symmetric objective function, where observed points incur an error in the predicted model SDF, and predicted points incur an error in the observation SDF. A model surface which is predicted to lie in front of or to the side of the observation will induce an error in this complementary term which can be reduced by moving the surface out of known free space. In addition to the elegant symmetry, we find the addition results in superior tracking robustness, as can be seen in figure (4).

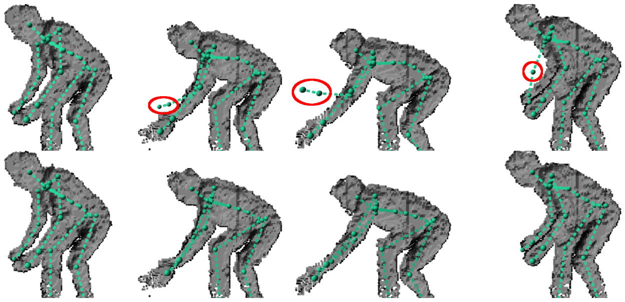


Fig. 4: Stills from human body tracking, demonstrating anecdotally the benefits of using free space information. Above, the asymmetric formulation loses track of the occluded arm, as indicated by the skeleton protruding into free space (grossly mispredicted joints are circled in red). Below, the symmetric formulation maintains a reasonable estimate through occlusion using negative information and remains in the basin of convergence when the arm reappears in the depth map. This figure is best viewed digitally.

Specifically, we first instantiate a volume of fixed size such that it entirely encloses our model in the current frame. We compute the pixel on which each voxel projects and instantiate its value to zero if it has depth greater than the observation, and to a large positive value otherwise. We then compute the 3D distance transform to obtain $\mathrm{SDF}_{\mathrm{obs}}(\mathbf{x};D)$. An illustration of the observation SDF and resulting error term is given in figures (3f) - (3j).

We generate in each iteration predicted points $\tilde{\mathbf{x}}_\mathbf{u}(\theta)$ using a standard OpenGL rendering pipeline and a shader which also produces a map of data association labels $\tilde{K}$ providing for each predicted point the rigid body from which it was generated. Then, we can augment our probability distribution in equation (3):

$$p(D|\theta) = \prod_{\mathbf{u}\in\Omega} p(D(\mathbf{u})|\theta)p(\tilde{\mathbf{x}}_\mathbf{u}(\theta)|D) \ , \tag{18}$$

and then define the likelihood function for the predicted vertex maps as:

$$p(\tilde{\mathbf{x}}_\mathbf{u}(\theta)|D) \propto \exp(-\mathrm{SDF}_{\mathrm{obs}}(\tilde{\mathbf{x}}_\mathbf{u}(\theta);D)^2) \ . \tag{19}$$

Finally, we arrive at a new energy minimization of the form:

$$\hat{\theta} = \arg\min_{\theta} \sum_{\mathbf{u}\in\Omega} \mathrm{SDF}_{\mathrm{mod}}(\mathbf{x_u};\theta)^2 + \lambda \sum_{\mathbf{u}\in\Omega} \mathrm{SDF}_{\mathrm{obs}}(\tilde{\mathbf{x}}_\mathbf{u}(\theta);D)^2 \tag{20}$$

which expresses both the positive and negative information available in the observation, and balances the two terms. The balancing factor λ has a probabilistic interpretation, in that it represents the ratio of the variance between the two measurement models, and given that the variance on each term is in units of distance in $\mathbb{R}^3$, it has a geometric interpretation as well. Note that the cost of computing $\mathrm{SDF}_{\mathrm{obs}}(x;D)$ is independent of the number of degrees of freedom and must be done only once per frame. The error induced in the observation by the current model rendering is computed by direct tri-linear interpolation of the SDF, as with the model SDF. For the derivatives, we note that for each predicted point $\tilde{\mathbf{x}}_\mathbf{u}$ there is some point $\tilde{\mathbf{x}}'_\mathbf{u}$ in frame of reference $\tilde{k}$ such that:

$$\tilde{\mathbf{x}}_\mathbf{u}(\theta) = T_{c,\tilde{k}}(\theta)\tilde{\mathbf{x}}'_\mathbf{u} \ . \tag{21}$$

We thus compute $\tilde{\mathbf{x}}'_\mathbf{u}$ for each point in the predicted point cloud and compute $J(\mathbf{u},\theta) = \frac{\partial}{\partial\Delta\theta}\mathrm{SDF}_{\mathrm{obs}}\left(T_{c,\tilde{k}}(\theta\oplus\Delta\theta)\tilde{\mathbf{x}}'_\mathbf{u};D\right)$, which is done exactly as in equations (12) and (13), excepting that the transform is inverted.

Subject	1	2	3	4	5	6
FORTH	35.4	19.8	27.3	26.3	16.6	46.2
ICP-PSO	9.3	24.1	14.4	13.4	11.0	20.0
DART asym.	32.0	34.4	47.4	21.3	19.1	35.6
DART symm.	14.1	12.0	24.7	14.4	12.6	26.8

TABLE II: Comparison with state-of-the-art hand tracking approaches on the dataset of [26]. Results are the average distance between predicted and ground truth markers for the wrist and five fingers, in mm. FORTH is the approach from Oikonomidis et al. [25]. Results for both FORTH and ICP-PSO are as reported and implemented by Qian et al. [26].

IV. Experiments

As the key focus of DART is the ability to generalize across a wide variety of models, we evaluate its performance in multiple domains. We give quantitative results for markerless tracking from depth datasets in the human body and human hand domains, and demonstrate a robotics application that relies on accurate tracking of a robot manipulator. There are some variations in parameter values to account for differences in model size and sensor types, but the code used to track all models is the same. Results are given for both the symmetric and asymmetric formulations of DART, which refer to the presence or absence of the free space constraints of section III-E, respectively. All results were computed at a frame rate at or above 30 FPS on an Nvidia GeForce GTX 680 with minimal CPU requirements.

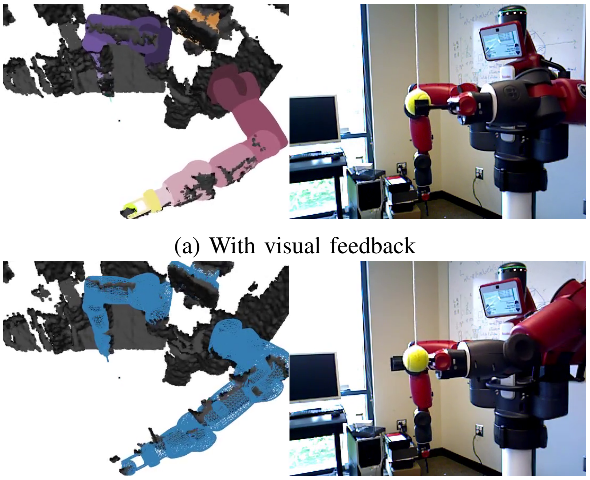
A. Human Body Tracking

We demonstrate the performance of DART on the EVAL dataset presented by Ganapathi et al. [12], and compare our results with the results from that work as well as the very recent results of Ye and Yang [35]. The human body models used are based on the meshes of [15], and had 48 degrees of freedom. The dataset consists of a sequence of point clouds with ground truth joint markers derived from a motion capture system. The error metric used is the percentage of joints predicted within 10cm of the given ground truth across all 24 sequences in the dataset. The results presented in figure (6) show that DART is competitive with state-of-the-art human body tracking approaches. Here, we would like to highlight that while some of the remaining error is due to inaccurate pose estimation, a significant portion is also due to noise in the ground truth markers themselves.

We are able to improve on the results of Ganapathi et al., even without using negative information, most likely because our precomputation of the model SDF allows us to use complex mesh models rather than being restricted to geometric primitive approximations. The gap between DART and the work of Ye and Yang on this dataset is likely explained by their further model improvements achieved by automated shape estimation, as well as their use of a smooth mesh deformation model as opposed to the segmentation of meshes into rigid parts required by DART.

B. Human Hand Tracking

We also demonstrate results on the recently published dataset of Qian et al. [26], comparing our work against their ICP-PSO algorithm as well as the approach of Oikonomidis et al. [25]. The human hand model is a collection of manually-defined ellipsoids, cylinders, and spheres, and has 26 degrees of freedom. The results are shown in Table II. We compare only against the reported results that don't make use of the hand-designed finger detector, as this is hand-specific. Again, DART is competitive with state-of-the-art approaches. Furthermore, this dataset clearly demonstrates the benefit of free space information incorporated via the observation SDF in the symmetric version of DART. Here, the use of such information is very important for accurately tracking a model that moves as quickly and with as much self-occlusion as the human hand. It is unclear how well the work of Ye and Yang, which does not take negative information into account at all, would perform in this domain.


(a) With visual feedback

(b) Without visual feedback

Fig. 5: Visual servoing results. In (a), the arm position is continuously estimated, allowing for a successful grasp. In (b), the position of the ball is estimated initially, but no visual feedback is used during motion. Though Baxter's end effector is reported to be at the initially-estimated ball position (shown in blue overlay), it is in fact significantly off, resulting in a grasping failure.

The gap between ICP-PSO and DART might be explained by the fact that the particle-based nature of PSO makes it more able to escape the sort of local minima that DART can fall into. Furthermore, the dataset was manually labelled using the model applied in ICP-PSO, which may have introduced a bias in the given marker positions. It is also unclear how well the ICP-PSO approach would generalize to human body tracking, which would require many more than 48 spheres and would likely degrade performance below real time. It is also highly unlikely that a sphere model could be used to track a robot with the level of accuracy achievable with our mesh model, as illustrated in the following experiment.

C. Visual Servoing

The datasets currently available for markerless tracking from a single RGB-D camera do not give a full picture of the level of accuracy achievable with a dense data term and a model of essentially arbitrary precision. In order to demonstrate its applicability to a practical robotics problem,

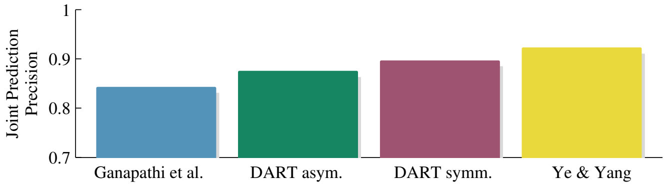


Fig. 6: Comparison with state-of-the-art human body tracking approaches on the EVAL dataset of [12]. Results given are the percentage of joints which are predicted within $10cm$ of the given ground truth positions.

we implemented a very simple closed-loop controller for visual servoing with the Rethink Robotics Baxter robot. We hung a tennis ball at 10 different locations within the reach of Baxter's arms, and the task was simply to grab the ball. First, we estimated the location of ball relative to Baxter by manually initializing the position of a Baxter and sphere model and refining the estimates with DART. We then used inverse kinematics to find joint angles required to grab a ball at the resulting relative offset and moved the arm accordingly. As a result of some combination of errors in the joint angles reported by Baxter (although the arm had just been calibrated) and depth skew in the reported depth values, the manipulator frequently went to an incorrect location and only succeeded in grasping the ball in 3 out of the 10 positions.

Next, we used DART to provide visual feedback to guide the manipulator towards the target. At fixed time intervals, the offset between the end effector position and the observed ball tracked by DART was used to refine the estimate of the ball position in Baxter's frame of reference and to update the inverse kinematics solution. When using real-time visual feedback, the ball was successfully grasped in all 10 locations. Figure (5) shows stills of an attempted grasp with and without visual feedback.

V. Limitations and Possible Extensions

DART currently employs a very simple approach to initialize the tracker and recover from failure. A more effective approach to these problems might be achieved by borrowing ideas from discriminative tracking techniques [29, 31, 17], where one would first learn to detect certain parts or features of an object, then use the detection to initialize the pose estimate within the basin of convergence.

DART also does not attempt to model physical constraints such as contacts. While our objective function does incentivize non-intersection, it does not explicitly penalize self-intersection of the model. This might be problematic for tracking actuated bodies interacting or manipulating other tracked objects, but could be solved by adding such constraints to the system, similar to the approach taken by Schulman et al. [28] for deformable objects.

While we conservatively ignored color information so as to be insensitive to lighting conditions, future work could add this information to strengthen tracking under some assumptions about the lighting. This could be useful for rejecting incorrect data association or the segmentation of background pixels lacking depth information to enhance free space constraints.

Since our focus has been on a maximally general and efficient framework, the motion model currently implemented in DART is a rather simple constant position model. However, DART could readily incorporate joint encoder information when tracking robots or better motion models generated by physics-based engines such as Bullet [2] or MuJoCo [33]. Instead of the current Kalman filter style model of the DART_Tracker, one could further increase robustness via a switching motion model implemented through Rao-Blackwellized particle filtering [32, 18], where multiple possible motions are sampled and an optimization is run for each particle. Obviously, one would have to carefully trade off the increase in computational cost against the increase in robustness. Finally, learning synergistic motion models that enable reasoning about redundancy in the state spaces of interacting objects, hands and body parts, could improve tracking performance of both the observed and occluded parts of the articulated models.

VI. Conclusion

This paper introduced DART, a general framework for tracking articulated models with formally defined kinematic and geometric structure. DART represents objects via a symmetric version of signed distance functions, extended to articulated objects. Our framework uses a local gradient based optimization to find the pose of the model which best explains the data points observed in a depth camera frame along with a prior based on previous data. Because of its representation, the energy function underlying DART is trivially parallelizable, and is optimized on a GPU, which allows us to use dense data to achieve accurate tracking results in real-time.

We demonstrate successful real-time tracking in scenarios with as many as 48 degrees of freedom. Each specific application of DART only requires the specification of a model: there are no underlying algorithmic changes required for the different objects. Though DART does not provide estimates that are demonstrably better than all existing methods in their specialized domains, our work is novel in its demonstration of competitive results across multiple domains.

We believe that the robotics community can greatly benefit from a general tool such as DART, enabling researchers to take advantage of accurate fine grained information about the state of articulated objects, and at real-time update rates. As a general base to articulated model tracking, there are a multitude of ways in which DART could be extended for any particular tracking scenario. Promising directions for future work include improved motion models, incorporation of physics-based reasoning, discriminative techniques for (re)-initialization of the tracker, and automatic learning of both articulated model geometry and motion constraints.

Acknowledgments

This work was funded in part by the Intel Science and Technology Center for Pervasive Computing (ISTC-PC) and by ONR grant N00014-13-1-0720.

References

[1] L. Ballan, A. Taneja, J. Gall, L. J. V. Gool, and M. Pollefeys. Motion Capture of Hands in Action Using Discriminative Salient Points. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *ECCV (6)*, volume 7577 of *Lecture Notes in Computer Science*, pages 640–653. Springer, 2012. ISBN 978-3-642-33782-6. 2

[2] Bullet Real-Time Physics Simulation. http://bulletphysics.org/wordpress/. 8

[3] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013. 1

[4] D. Canelhas, T. Stoyanov, and A. Lilienthal. SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3671–3676, Nov 2013. 1

[5] A. Comport, E. Marchand, and F. Chaumette. Kinematic sets for real-time robust articulated object tracking. *Image and Vision Computing*, 25(3):374–391, 2007. 1

[6] G. Dewaele, F. Devernay, and R. Horaud. Hand Motion from 3D Point Trajectories and a Smooth Surface Model. In T. Pajdla and J. Matas, editors, *ECCV (1)*, volume 3021 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2004. ISBN 3-540-21984-6. 2

[7] T. Drummond and R. Cipolla. Visual tracking and control using Lie algebras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999. 5

[8] T. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, pages 315–320 vol.2, 2001. 1, 5

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004. 3

[10] A. W. Fitzgibbon. Robust Registration of 2D and 3D Point Sets. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2001. 1, 4

[11] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 755–762. IEEE, 2010. 2

[12] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. In *Computer Vision–ECCV 2012*, pages 738–751. Springer, 2012. 2, 6, 7, 8

[13] D. Grest, J. Woetzel, and R. Koch. Nonlinear body pose estimation from depth images. In *Pattern Recognition*, pages 285–292. Springer, 2005. 2

[14] D. Hähnel, S. Thrun, and W. Burgard. An Extension of the ICP Algorithm for Modeling Nonrigid Objects with Mobile Robots. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI. 1

[15] T. Helten, A. Baak, G. Bharaj, M. Müller, H. Seidel, and C. Theobalt. Personalization and Evaluation of a Real-time Depth-based Full Body Tracker. In *Proceedings of the 3rd joint 3DIM/3DPVT Conference (3DV)*, 2013. To appear. 1, 7

[16] P. Henry, D. Fox, A. Bhowmik, and R. Mongia. Patch Volumes: Segmentation-based Consistent Mapping with RGB-D Cameras. In *International Conference on 3D Vision (3DV)*, 2013. 1, 2, 3

[17] C. Keskin, F. Kiraç, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *ICCV Workshops*, pages 1228–1234. IEEE, 2011. ISBN 978-1-4673-0062-9. 2, 8

[18] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276, 2004. 8

[19] N. Kyriazis and A. Argyros. Physically plausible 3D scene tracking: The single actor hypothesis. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1, 2

[20] H. Li, R. W. Sumner, and M. Pauly. Global Correspondence Optimization for Non-Rigid Registration of Depth Scans. *Computer Graphics Forum (Proc. SGP'08)*, 27(5), July 2008. 1

[21] H. Li, J. Yu, Y. Ye, and C. Bregler. Realtime Facial Animation with On-the-fly Correctives. *ACM Transactions on Graphics*, 32(4), July 2013. 1

[22] R. A. Newcombe. *Dense Visual SLAM. PhD thesis*. PhD thesis, Imperial College London, June 2014. 1

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 1, 2, 3

[24] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *BMVC*. BMVA, 2011. doi: 10.5244/C.25.101. 1, 2

[25] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *BMVC*, pages 1–11, 2011. 7

[26] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and Robust Hand Tracking from Depth. 2014. 1, 2, 7

[27] C. Ren and I. Reid. A Unified Energy Minimization Framework for Model Fitting in Depth. In *Computer Vision ECCV 2012. Workshops and Demonstrations*, volume 7584 of *Lecture Notes in Computer Science*, pages 72–82. Springer Berlin Heidelberg, 2012. 1

[28] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking Deformable Objects with Point Clouds. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2013. 1, 2, 8

[29] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 1, 2, 8

[30] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. CopyMe3D: Scanning and Printing Persons in 3D. In *Pattern Recognition*, pages 405–414. Springer, 2013. 2, 3

[31] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 103–110. IEEE, 2012. 2, 8

[32] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, September 2005. ISBN 0-262-20162-3. 3, 8

[33] E. Todorov. Analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2014. 8

[34] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*, 2012. 1

[35] M. Ye and R. Yang. Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1, 2, 7