Real-time Model-based Articulated Object Pose Detection and Tracking with Variable Rigidity Constraints

Karl Pauwels Leonardo Rubio Eduardo Ros
University of Granada, Spain
{kpauwels,lrubio,eros}@ugr.es

Abstract

A novel model-based approach is introduced for real-time detection and tracking of the pose of general articulated objects. A variety of dense motion and depth cues are integrated into a novel articulated Iterative Closest Point approach. The proposed method can independently track the six-degrees-of-freedom pose of over a hundred of rigid parts in real-time while, at the same time, imposing articulation constraints on the relative motion of different parts. We propose a novel rigidization framework for optimally handling unobservable parts during tracking. This involves rigidly attaching the minimal amount of unseen parts to the rest of the structure in order to most effectively use the currently available knowledge. We show how this framework can be used also for detection rather than tracking which allows for automatic system initialization and for incorporating pose estimates obtained from independent object part detectors. Improved performance over alternative solutions is demonstrated on real-world sequences.

1. Introduction

Much progress has been made recently with respect to real-time model-based pose estimation of rigid objects. In many situations, certain relations are known to exist between such objects (*e.g*. planar motion) or result from the way in which these objects are observed (*e.g*. through multiple fixed cameras). Articulated objects constitute an important class of objects that can be described as a set of rigid components whose relative motion satisfies a number of constraints. Although great progress is being made on certain articulated objects, such as hands or human bodies, less work focuses on more general articulated objects. These are important for robotic manipulation of dexterous objects or for understanding human manipulation of such objects. Exploiting the dynamic relations between separate objects or object parts can greatly facilitate detection and tracking of these objects in complex real-world scenarios.

1.1. Related Work

Although much research is concerned with how the relationships between different objects or object parts can be extracted from visual information [11], we only focus here on model-based methods that assume this information to be known.

Many discriminative [23] and generative [18] methods have been proposed that focus on particular articulated object classes such as hands or bodies. These methods can be difficult to extend to more general object types since they incorporate large amounts of prior information, often extracted in a learning stage, and actively impose it to *e.g*. reduce the search space of sampling-based methods. Certain assumptions (*e.g*. dynamical models or a single-actor hypothesis [12]) and/or stochastic optimization methods have also been used to apply sampling-based methods in more general scenarios. Another type of approaches instead operates on a differentiable formulation of the problem, which imposes limitations on the energy function, but scales more efficiently to high problem dimensionality [5]. Many depth-only articulated Iterative Closest Point (ICP) methods have also become widely-used [8, 21].

A distinction can be made between pose *detection* and pose *tracking*. Detection methods do not rely on temporal information but instead recover the pose from a single image frame. Tracking methods on the other hand refine an estimate (*e.g*. from the previous time) based on current time measurements. Currently most combined articulated pose detection and tracking methods are either discriminative or employ heuristic methods (*e.g*. hill climbing) to incorporate part detectors into particle-filter based methods [7].

Regarding articulated pose tracking, a flexible constraint framework was introduced by [6] and subsequently extended to more easily handle highly complex structures [3]. Not considered in those works are situations where parts become invisible, a situation frequently encountered with complex objects, due to occlusions, and with noisy input. Solving for such invisible parts effectively constitutes an inverse kinematics problem. Inverse kinematics have been

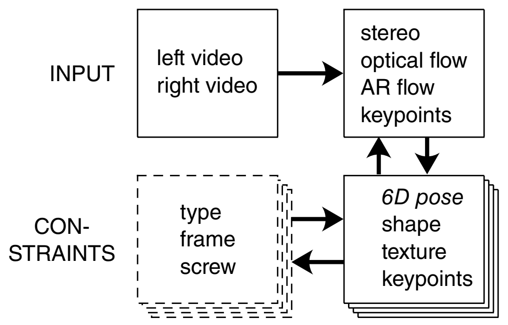

Figure 1. Method overview. The different visual cues (AR = Augmented Reality) are combined with the model components to estimate the pose of objects or object parts while satisfying a time-varying set of constraints (2.3.2) on relative velocities.

exploited previously to reduce the sampling manifold for particle filtering [22] by analytically solving sub-problems of the kinematic hierarchy based on separating active from inactive parameters. We introduce a novel automated approach here that solves such inverse kinematics problems under a continuously changing structure and apply it on top of a real-time multi-cue integrated detection and tracking system.

1.2. Main Contributions

We first extend the work of [6] by incorporating various motion and depth cues (as opposed to just edges) which effectively results in a novel articulated ICP approach. We next introduce a rigidization framework that enables handling arbitrary configurations of visible and invisible parts. We demonstrate that this involves solving inverse kinematics problems over a continuously changing object structure. We also show how this framework can be used for detection as well as tracking. This enables automatic initialization and/or recovery in case of tracking failures through the incorporation of object part detectors. Finally, we demonstrate that the entire framework obtains strong real-time results on a real-world dataset far exceeding the complexity of those used in [3, 6], in terms of speed, number and complexity of object parts, occlusions, and the critical need for a detection ability.

2. Proposed Method

A concise overview of the proposed method is shown in Fig. 1. The method extracts a variety of low-level visual cues from stereo video. These cues are combined with (a priori known) model information such as shape and appearance in order to update the six degrees-of-freedom (DOF) pose of multiple objects or object parts. A time-varying set of structural constraints are enforced between different model parts (e.g. hinges, slides) and/or between different models (e.g. to restrict motion to a common ground plane). The nature of these constraints is changed continuously, based on the current visibility of the scene, using a novel rigidization framework that rigidly attaches the minimal required number of unseen parts to the rest of the object. Finally, the model information (satisfying the active constraints) is continuously fed back to facilitate cue extraction.

2.1. Scene Representation

The modeled scene consists of multiple rigid objects or object parts, each represented by a 3D textured wireframe model. Self-occlusions and occlusions between different modeled objects are handled by rendering the scene using OpenGL. In addition, SIFT features are extracted from different viewpoints of each model and mapped onto the surface.

2.2. Visual Cues

We rely on highly efficient GPU libraries for dense optical flow, model-based dense stereo, and SIFT feature extraction [20, 24]. The model depth obtained in the current frame is used to initialize a coarse-to-fine stereo algorithm. This enables the efficient extraction of stereo in scenes with large disparity variability. In a similar way, the model texture at the current frame is used to compute Augmented Reality (AR) flow, which is the optical flow between a synthetically generated image based on the current tracking hypothesis, and the current image. This AR flow counters drift when included in tracking and also provides an indication of how well the hypothesized scene matches the observed [19]. It is extensively used as reliability measure in the remainder.

2.3. Pose Tracking and Detection

Similarly to [6], we use a redundant representation where the full pose of each rigid part is stored independently. Section 2.3.1 describes how a rigid pose update can be obtained from all the visual cues. In the presence of constraints, this update is not actually performed, but instead used to impose these constraints on the relative velocities between different rigid pose updates. This is explained in Section 2.3.2. It has been shown that pre-imposing the kinematic constraints during tracking or imposing the constraints after tracking (as done here) results in the same accuracy [3]. See [4] for a description and analysis of Drummond's method that is more extensive than the one provided in the next sections.

2.3.1 Rigid Pose Tracking

The critical aspect here that allows both real-time performance and a flexible constraint framework, is the ability to separate the composition of the normal equations from the constraint enforcement. This allows parallelizing the tracking of individual parts in a GPU-friendly manner. Initially the problem is thus considered as a set of independent 6DOF rigid pose estimations. Considering one object, the initial aim is to recover the rigid rotation and translation that best explains the dense visual cues and transforms each model point $\mathbf{m} = [m_x,\, m_y,\, m_z]^\top$ at time t into point $\mathbf{m}'$ at time $t+1$:

$$\mathbf{m}' = \mathbf{R}\,\mathbf{m} + \mathbf{t}\ , \tag{1}$$

with $\mathbf{R}$ the rotation matrix and $\mathbf{t} = [t_x,\, t_y,\, t_z]^\top$ the translation vector. The rotation matrix can be approximated:

$$\mathbf{m}' \approx (\mathbf{1} + [\boldsymbol{\omega}]_\times)\,\mathbf{m} + \mathbf{t}\ , \tag{2}$$

with $\boldsymbol{\omega} = [\omega_x,\, \omega_y,\, \omega_z]^\top$ the rotation axis and angle. Each stereo disparity measurement, d', at time $t+1$ is used to construct an approximation, $\mathbf{s}'$, to $\mathbf{m}'$:

$$\mathbf{s}' = \begin{bmatrix} s'_x \\ s'_y \\ s'_z \end{bmatrix} = \begin{bmatrix} x\, s'_z/f \\ y\, s'_z/f \\ -f\, b/d' \end{bmatrix}\ , \tag{3}$$

with pixel coordinates $\mathbf{x} = [x, y]^\top$ (with nodal point as origin), focal length f and b the baseline of the (rectified) stereo rig. Next, the model point $\mathbf{m}$ corresponding to $\mathbf{s}'$ is obtained using the efficient projective data association algorithm [1] that corresponds stereo measurements to model points that project to the same pixel. By projecting the error on $\mathbf{n} = [n_x,\, n_y,\, n_z]^\top$, the model normal vector in $\mathbf{m}$, a linearized version of the *point-to-plane* distance is obtained, expressing the distance from the reconstructed points to the plane tangent to the model [25]:

$$e_S(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \left(\Big[(\mathbf{1} + [\boldsymbol{\omega}]_\times)\,\mathbf{m}_i + \mathbf{t} - \mathbf{s}'_i \Big] \cdot \mathbf{n}_i \right)^2 . \tag{4}$$

This strictly shape-based error measure is linear in the parameters of interest $\mathbf{t}$ and $\boldsymbol{\omega}$.

Another linear, but strictly motion-based, error measure is derived from the differential motion equation from classical kinematics that expresses the 3D motion of a point, $\dot{\mathbf{m}}$, in terms of its 3D translational and rotational velocity:

$$\dot{\mathbf{m}} = \mathbf{t} + [\boldsymbol{\omega}]_\times \mathbf{m}\ . \tag{5}$$

This can be used to express the optical flow $\dot{\mathbf{x}} = [\dot{x}, \dot{y}]^\top$ as follows [15]:

$$\dot{x} = \frac{(f\, t_x - x\, t_z)}{m_z} - \frac{x\, y}{f}\,\omega_x + (f + \frac{x^2}{f})\,\omega_y - y\,\omega_z\ , \tag{6}$$

$$\dot{y} = \frac{(f\, t_y - y\, t_z)}{m_z} - (f + \frac{y^2}{f})\,\omega_x + \frac{x\, y}{f}\,\omega_y + x\,\omega_z\ , \tag{7}$$

which are linear in $\mathbf{t}$ and $\boldsymbol{\omega}$ provided the depth of the point is known. We obtain this depth m_z by rendering the model at the current pose estimate. Since we have two sources of pixel motion (the optical flow $\mathbf{o} = [o_x,\, o_y]^\top$ and AR flow $\mathbf{a} = [a_x,\, a_y]^\top$), we have two error functions:

$$e_O(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \|\dot{\mathbf{x}}_i - \mathbf{o}_i\|^2\ , \tag{8}$$

$$e_A(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \|\dot{\mathbf{x}}_i - \mathbf{a}_i\|^2\ . \tag{9}$$

Both the linearized point-to-plane distance in the stereo case and the differential motion constraint in the optical and AR flow case now provide linear constraints on the same rigid motion representation $(\mathbf{t}, \boldsymbol{\omega})$ and can thus be minimized jointly using the following rigid pose error function:

$$E_r(\mathbf{t}, \boldsymbol{\omega}) = e_S(\mathbf{t}, \boldsymbol{\omega}) + e_O(\mathbf{t}, \boldsymbol{\omega}) + e_A(\mathbf{t}, \boldsymbol{\omega})\ . \tag{10}$$

We can rewrite (10) as follows to expose the linearity:

$$E_r(\boldsymbol{\alpha}) = (\mathbf{F}\boldsymbol{\alpha} - \mathbf{d})^\top (\mathbf{F}\boldsymbol{\alpha} - \mathbf{d})\ , \tag{11}$$

where the rigid motion parameters are stacked into a screw vector $\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{t} \end{pmatrix}$ for convenience, and $\mathbf{F}$ and $\mathbf{d}$ are obtained by gathering the sensor data according to (4),(6),(7). This can be solved in the least-squares sense using the normal equations:

$$\mathbf{F}^\top \mathbf{F} \boldsymbol{\alpha} = \mathbf{F}^\top \mathbf{d}\ . \tag{12}$$

2.3.2 Articulated Pose Tracking

Drummond and Cipolla [6] pointed out that when a part's unconstrained velocity update $\boldsymbol{\alpha}$ is modified into $\boldsymbol{\beta}$, the sum-squared error changes according to $(\boldsymbol{\beta} - \boldsymbol{\alpha})^\top \mathbf{C} (\boldsymbol{\beta} - \boldsymbol{\alpha})$, with $\mathbf{C} = \mathbf{F}^\top \mathbf{F}$. They showed how equality constraints can be enforced between corresponding screw values after transforming them to the same coordinate frame. For each link between object parts p and q a coordinate frame $\mathbf{T}_{p,q}$ and a set of constraints $\mathbf{c}^k_{p,q}, k = 1 \ldots K_{p,q}$ are formulated. The coordinate frame is chosen to simplify the constraint formulation, and is typically aligned with the joint axis between the parts. The adjoint transformation [17]:

$$\mathcal{T} = \mathrm{Ad}(\mathbf{T}) = \mathrm{Ad}\left(\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \right) = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ 0 & \mathbf{R} \end{bmatrix}\ , \tag{13}$$

can then be used to transform both screws into a joint frame associated with the link and formulate each constraint as follows:

$$(\boldsymbol{\beta}_p - \boldsymbol{\beta}_q)^\top \mathcal{T}^\top_{p,q} \mathbf{c}^k_{p,q} = 0\ . \tag{14}$$

The nature and number of constraints $K_{p,q}$ between two parts determines either the joint type, full rigidity between the parts, or lack of constraints. Each $\mathbf{c}^k_{p,q}$ is a column 6-vector (typically taken from the identity matrix) that indicates which DOFs are constrained. They can all be stacked together as follows:

$$\mathcal{C}_{p,q} = \left[\mathbf{c}^1_{p,q}\mathbf{c}^2_{p,q}\dots\mathbf{c}^{K_{p,q}}_{p,q}\right] . \tag{15}$$

All constraints for the link can then be expressed jointly as:

$$(\boldsymbol{\beta}_p - \boldsymbol{\beta}_q)^\top \mathcal{T}^\top_{p,q}\mathcal{C}_{p,q} = \mathbf{0}_{K_{p,q}} . \tag{16}$$

This constrained optimization problem can be solved by introducing Lagrange multipliers for each link:

$$\boldsymbol{\lambda}_{p,q} = \left[\lambda^1_{p,q}\lambda^2_{p,q}\dots\lambda^{K_{p,q}}_{p,q}\right]^\top . \tag{17}$$

If we consider a simple three part chain consisting of grandparent part p, parent part q, and child part a, the following is obtained when differentiating the Lagrange system to $\boldsymbol{\beta}_q$:

$$2\mathbf{C}_q(\boldsymbol{\beta}_q - \boldsymbol{\alpha}_q) - \mathcal{T}^\top_{p,q}\mathcal{C}_{p,q}\boldsymbol{\lambda}_{p,q} + \mathcal{T}^\top_{q,a}\mathcal{C}_{q,a}\boldsymbol{\lambda}_{q,a} = \mathbf{0}_6 . \tag{18}$$

The first component is due to part q itself, the second due to the parent of q and the third due to the child of q. The second component is omitted at the root of the hierarchy, and more instances of the third component are added in case of multiple children.

Considering a simple two-part scenario with parent p and child q, the solution can be obtained directly from the following matrix equation:

$$\begin{bmatrix} 2\mathbf{C}_p & & \mathbf{A} \\ & 2\mathbf{C}_q & -\mathbf{A} \\ \mathbf{A}^\top & -\mathbf{A}^\top & \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_p \\ \boldsymbol{\beta}_q \\ \boldsymbol{\lambda}_{p,q} \end{bmatrix} = \begin{bmatrix} 2\mathbf{C}_p\boldsymbol{\alpha}_p \\ 2\mathbf{C}_q\boldsymbol{\alpha}_q \\ \mathbf{0}_{K_{p,q}} \end{bmatrix} , \tag{19}$$

where $\mathbf{A} = \mathcal{T}^\top_{p,q}\mathcal{C}_{p,q}$. The left-hand matrix can become large, but it is sparse and symmetric and efficient numerical procedures can be used to solve it, even for complicated articulated objects. Figure 2B shows an example of the sparsity pattern of this matrix for a 12-part articulated object with kinematic structure shown in Fig. 2A.

An iterative weighted least squares Tukey biweight M-estimator [16] is used to increase the robustness to outliers. Once the robust velocity updates are found, the articulated pose is updated by propagating the new joint angles forward through the kinematic chain. The entire procedure (rendering, segmentation, normal equations, ...) is then repeated as in other ICP algorithms in order to deal with the non-linearity of the problem. We typically use a total of three robust and three ICP iterations.

Unlike the scenarios considered in [3, 6] we are interested in cases where certain parts are unobservable, a situation that occurs frequently in real-world scenarios. This has two important implications. First of all the solution of the Lagrange multipliers and the $\boldsymbol{\beta}$ values cannot be separated as done in [3, 6] (this requires inverting the covariance matrix $\mathbf{C}$). Secondly the invisible parts need to be sufficiently constrained by the visible parts. This is explained next.

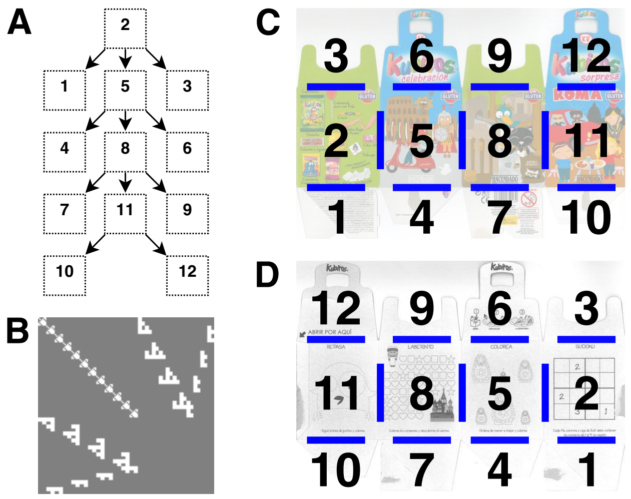

Figure 2. Kubito object. (A) Kinematic structure, (B) sparsity pattern of constraint matrix (19), (C) front and (D) back texture and articulation axes (blue lines). The numbers in (A), (C), and (D) correspond to the same part and are only shown for illustration.

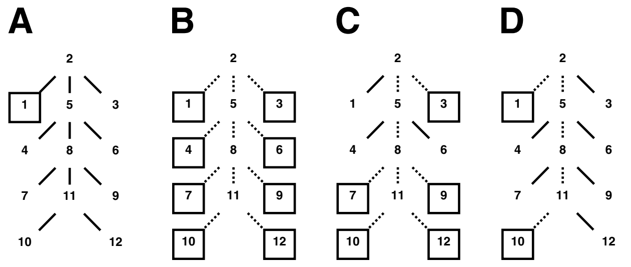

Figure 3. Visibility-based rigidization. Only the boxed object parts are visible. The minimal links that need to be made rigid are shown in solid lines. Links that can remain flexible are shown dotted. In (A) the entire structure needs to be made rigid, in (B) no links require rigidity, and (C,D) show intermediate non-trivial scenarios.

2.3.3 Rigidization Framework

When parts become invisible the constraint matrix may be singular. We propose to counter this by enforcing the minimal number of additional rigidity constraints on the kinematic structure required to obtain an invertible matrix. The intuition behind this is that in the absence of new observations, parts are assumed to remain in their last seen configuration. This is expected to facilitate tracking once they reappear.

This is effectively an inverse kinematics problem, however what is special in this case is that the structure is allowed to change from frame to frame, depending on the currently visible parts. Which links to fix is not straightforward and depends both on the kinematic hierarchy and on the orientation of the joints. Consider the examples in Fig. 3 that correspond to the *Kubito*-object of Fig. 2. In this figure, visible parts are boxed, links that need to be made

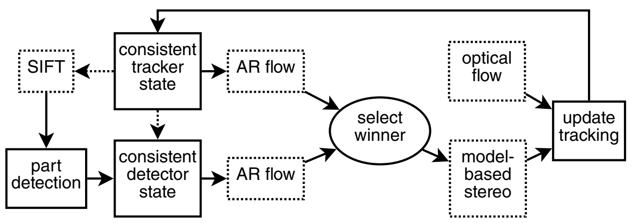

Figure 4. Combined tracking and detection. Dotted boxes refer to visual cues. The consistent tracker state determines which SIFT-based part detector is active according to (20).

rigid are shown solid, and links that can remain flexible are shown dotted. If only one part is visible (A) the entire structure needs to be made rigid, whereas certain partial visibility patterns do not require any rigidization (B). However, more complex and non-trivial situations can occur as well (C,D). In (D) for example the entire central structure is constrained by parts 1 and 10 due to the relative orientation of the joints. Note that we are always considering the incremental rather than the absolute pose.

We have performed a search procedure here to obtain the minimal rigidization pattern by evaluating the rank of the left matrix in (19) for different visibility configurations. The result is stored in a look-up table and allows us to instantly select the minimal number of links that need to be fixed depending on the current visibility. The object's joints were slightly bended during look-up table construction to avoid ambiguous configurations.

2.3.4 Incorporating Detection

So far we have only considered how to update the articulated pose on the basis of the previous frame estimate. This does not enable automatic initialization or recovery in case of occlusions. We next explain how we extend a RANSAC-based monocular perspective-n-point pose detector [2, 13] for rigid pose estimation, to the articulated case. An overview is shown in Fig. 4. The detector exhaustively matches image (2D) to model codebook (3D) SIFT keypoint descriptors considering one object (part) at a time in order to improve accuracy. Which object or object part to focus on is determined probabilistically and depends on the current tracking reliability of that part:

$$p(o) = \frac{1 - r(o)}{\sum_i \big(1 - r(i)\big)} \, , \tag{20}$$

with $r(o)$ the reliability, determined by the proportion of valid AR flow in the segment's region [19]. The regions are obtained by rendering the scene with custom OpenGL shaders that provide the part index for each pixel. This attention mechanism focuses the detector's limited resources on the least reliable part. As a result, in each frame, the detector provides a rigid pose hypothesis for a new (possibly previously unseen) part. To incorporate this novel part pose into the current articulated pose estimate a consistent update (one that satisfies the constraints) needs to be generated for the object as a whole. A consistent hypothesis can be obtained efficiently in simulation as follows. The model vertices (possibly subsampled for efficiency) are first configured in the current consistent tracker state (or in an initial position at the start of the algorithm or when tracking is lost). The update procedure from Section 2.3.2 and rigidization framework from Section 2.3.3 are then used to generate velocity updates that incorporate the newly detected part and satisfy the constraints. The already visible parts function both as initial and target position in this procedure. As a result a new consistent state is obtained that minimizes discrepancies from the previously visible and new parts. Since this uses model vertices rather than image data, the exact 3D velocities between current and target object parts are known at all times, and the optimization can be performed directly using (5) rather than (10) in (12). As a consequence this procedure is quite robust and can resolve many situations. Since it relies on linearizations internally, it does at times get stuck in local minima. In this case, the AR flow generated with this incorrect hypothesis signals its low quality and the detector/tracker winner selection will discard this solution. A better hypothesis will likely be generated at subsequent frames due to the probabilistic nature of the part selection (20).

As shown in Fig. 4 AR flow is computed for both the tracker and detector state in order to determine their reliability. The detector state is only chosen if it improves upon the tracker's reliability for that part and does not negatively affect the reliability of any other currently tracked part. A part is considered visible if its proportion valid AR flow exceeds 0.15. Once visible, it remains visible until sufficient evidence is available to actively remove it. In this way the rigidization is effectively used to continue tracking occluded parts. Occluded parts can then be picked up immediately when they re-appear, without requiring the detector.

3. Processing Times

The timings mentioned in this section were obtained using an NVIDIA Geforce GTX 590 and an Intel Core i7. The system relies heavily on the integration between OpenGL and NVIDIA's CUDA framework [14] in order to achieve real-time performance. The low-level cue extraction uses efficient coarse-to-fine algorithms to achieve frame rates around 100 Hz [19]. Initially, the normal equations can be composed independently and in parallel for each segment. The most time-consuming additional step required as compared to single object tracking is assigning the valid measurements to their respective segments (in accordance

Table 1. Multiple rigid object pose tracking frame rates (in Hz)

	# samples	
# parts	50,000	500,000
1	62	57
20	55	44
150	47	38

with the current pose estimates). For this, the segment indices need to be sorted. Since the number of segments is limited, an efficient radix sort can be used here [10]. The computation of the normal and weighted normal equations (for robust estimation) is carefully done, combining composition with compaction and subsequent reduction. An approximate median operation is used to obtain the scale of the residuals, required by the re-weighting procedure. Table 1 shows the achieved frame rates for different numbers of tracked parts as a function of number of data samples (optical flow, stereo, ...) used. The SIFT-based part detection runs independently on the GTX 590's second GPU. Its estimates are employed when available so that it does not slow down the tracker. In our current implementation it provides a pose hypothesis at 20 Hz. The most computationally demanding aspect of the constraint enforcement is solving (19). The left-hand matrix is however sparse and symmetric and can be solved efficiently using iterative methods in case the system needs to be scaled to very complex articulated objects. For the 12-part object considered here, we required 0.3 ms to solve the equation using the Eigen library [9] on a single CPU core. In the examples shown here this was performed 3 times in the detector simulations, and 9 times in tracking (3 internal robust iterations and 3 external ICP iterations). In total, the constraint enforcement increases computation times by 5 ms, which still enables frame rates exceeding 35 Hz on the above-mentioned hardware.

4. Comparative Evaluation

4.1. Real-world Sequences

We compare the proposed method to a number of alternatives using complex real-world stereo sequences. These sequences were recorded using low quality webcams under difficult illumination conditions and contain the *Kubito*-object depicted in Fig. 2 as target. The dataset is available on-line.[1] Additional sequences with different objects are shown in the supplemental material video. The *Kubito* object's pose is 17-dimensional (11 revolute joints, 3 translation, 3 rotation). Many parts of the object (3,4,9,10) are untextured and the entire backside (Fig. 2D) contains very little visual information. Figure 5 shows some examples of the three sequences considered. In the *wave* sequence the object is waved in front of the camera. In the *box* sequence the object is folded and unfolded into a box and rotated. The *various* sequence finally contains non-trivial configurations that challenge the detector.

Obtaining the ground-truth 17D articulated pose for these sequences is non-trivial and would involve multiple cameras and/or intrusive visual or magnetic markers. Instead we have used a manual labeling approach. The three sequences, each around 1,000 frames in length, were processed using five different algorithms (see Sect. 4.2). Each algorithm returns an articulated pose estimate together with the number of parts it considers reliable. To evaluate these estimates, we asked a naive human subject to count the number of correctly identified parts. A total of 100 frames were randomly selected from each sequence and the estimates obtained by all algorithms on these frames were shown to the subject in random order so that the subject could not identify the algorithm. The estimates were visualized both as regions marked on the image (as in Fig. 5) and as a 3D view rendered from a viewpoint different from the camera to facilitate judging the pose.

4.2. Alternative Methods

We compare the performance of the proposed **rigidization** method to four alternatives. The simplest, **SIFT**, considers only the pose estimates as returned by the detector. Contrary to its use in the real-time trackers, in this case the detector is run multiple times on each frame, once for each object part. Although far from real-time, in this way its performance provides an indication of the basic information available to the trackers for initialization and recovery. The second method, **multi-object**, is a real-time variant that detects and tracks the different object parts independently, without enforcing the articulation constraints. The third method, **depth-only**, retains all the components of the proposed method (constrained detection and tracking, and rigidization) but relies on depth-only (AR flow is only used to evaluate reliability). In this way the method is comparable to other depth-only articulated ICP algorithms [8, 21]. A final alternative considered, **pseudo-inverse**, uses the Moore-Penrose pseudoinverse rather than the rigidization framework to handle the singularity of (19) in the case of invisible parts. This involves a Singular Value Decomposition (SVD). The latter is not real-time capable but is included to further justify the rigidization approach.

4.3. Results

Some example results obtained by the five algorithms on the different sequences are shown in Fig. 5. The average number of correctly (True Positives, TP) and incorrectly (False Positives, FP) detected parts are summarized in Table 2. Note that this number is much lower than the total number of parts (12) but since the object undergoes com-

[1]`http://www.karlpauwels.com`

Table 2. Average number of correctly (TP) and incorrectly (FP) tracked parts (out of 12) for three different sequences

	wave		*box*		*various*	
	TP	FP	TP	FP	TP	FP
rigidization	4.30	0.05	2.19	0.28	2.56	0.14
pseudo-inverse	4.04	0.01	1.93	0.19	2.41	0.18
multi-object	4.15	2.02	2.11	2.94	2.05	3.19
SIFT	3.98	1.15	1.86	0.89	2.16	1.19
depth-only	0.66	1.36	0.25	1.06	0.46	0.67

plex foldings, only a small number of parts are visible in each frame. In addition, the untextured parts are not picked up by the detector.

The proposed rigidization framework outperforms all the approaches in terms of correctly detected parts and obtains similar performance to the *pseudo-inverse* method in terms of incorrectly detected parts. The *rigidization* framework outperforms the *pseudo-inverse* solution by maintaining previously gathered evidence when the object is occluded. The *multi-object* method can extend *SIFT*'s part detections over time, but can not resolve ambiguous parts (*e.g.* 6 and 12 in Fig. 2) and its reliability evaluation suffers from the small object parts' sizes and general lack of texture. It is also unable to maintain previously seen structure over time through rigidization. The *SIFT* results provide an indication of the quality of the information provided to the trackers, how well the trackers maintain it over time (the increased TP relative to *SIFT*), and how well the trackers evaluate their own reliability (the decreased FP relative to *SIFT*). We should point out that many of the pose estimates returned by *multi-object* and *SIFT* that were labeled correct by the human subject are clearly inferior to the ones obtained by the constrained methods, since the latter can minimize the error over the entire (consistent) structure. To quantify this, more precisely measured ground-truth data is required. Finally, the *depth-only* method performs very weakly with this type of object, since the shape information is insufficient to uniquely determine its pose. As can be seen in the top and bottom rows of Fig. 5E the estimates typically shift in a direction that is unconstrained by shape information. Additional results on these and other sequences are shown in the supplemental material video.

5. Discussion

As an alternative to the proposed rigidization framework, a stationary prior could be imposed in a filtering method. This however requires using the conventional kinematical approach in a *pre-* rather than *post*-imposed constraint method. This results in different normal equations in each segment depending on the depth in the kinematic chain, which is problematic for GPU implementation (different computations are required in each segment) and scaling (the number of parameters involved grows with chain depth). Another alternative explored here involves solving the singular system using for example the Moore-Penrose pseudoinverse, resulting in the smallest norm solution. This is undesirable for two reasons. First of all such techniques are much more computationally intense since they require a SVD or some other rank-revealing operation, and secondly, they lose the advantage of keeping the object in the last known configuration, as confirmed by the results in Table 2.

The approach used here also allows different types of (time-varying) constraints between relative object motion (such as planarity) and/or multi-camera configurations. Such scenarios will be considered in future work.

In conclusion, we have presented a novel real-time articulated pose detection and tracking method and have demonstrated how complex real-world situations can be successfully handled by continuously changing the object's rigidity structure. This has resulted in improved performance as compared to a variety of alternative solutions.

Acknowledgments

This work has been supported by grants from the European Commission (Marie Curie FP7-PEOPLE-2011-IEF-301144 and TOMSY FP7-270436). The GPU used for this research was donated by the NVIDIA Corporation.

References

[1] G. Blais and M. Levine. Registering multiview range data to create 3D computer objects. *IEEE PAMI*, 17(8):820–824, 1995.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] T. de Campos, B. Tordoff, and D. Murray. Recovering articulated pose: a comparison of two pre and postimposed constraint methods. *IEEE PAMI*, 28(1):163 –168, 2006.

[4] T. E. de Campos. *3D visual tracking of articulated objects and hands*. PhD thesis, Department of Engineering Science, University of Oxford, 2006.

[5] M. de La Gorce, D. Fleet, and N. Paragios. Model-based 3D hand pose estimation from monocular video. *IEEE PAMI*, 33(9):1793–1805, 2011.

[6] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE PAMI*, 24(7):932 –946, 2002.

[7] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *CVPR*, pages 755–762, 2010.

[8] D. Grest, J. Woetzel, and R. Koch. Nonlinear body pose estimation from depth images. Lecture Notes in Computer Science, pages 285–292. 2005.

[9] G. Guennebaud, B. Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010.

[10] J. Hoberock and N. Bell. Thrust: A parallel template library, 2010. Version 1.7.0.

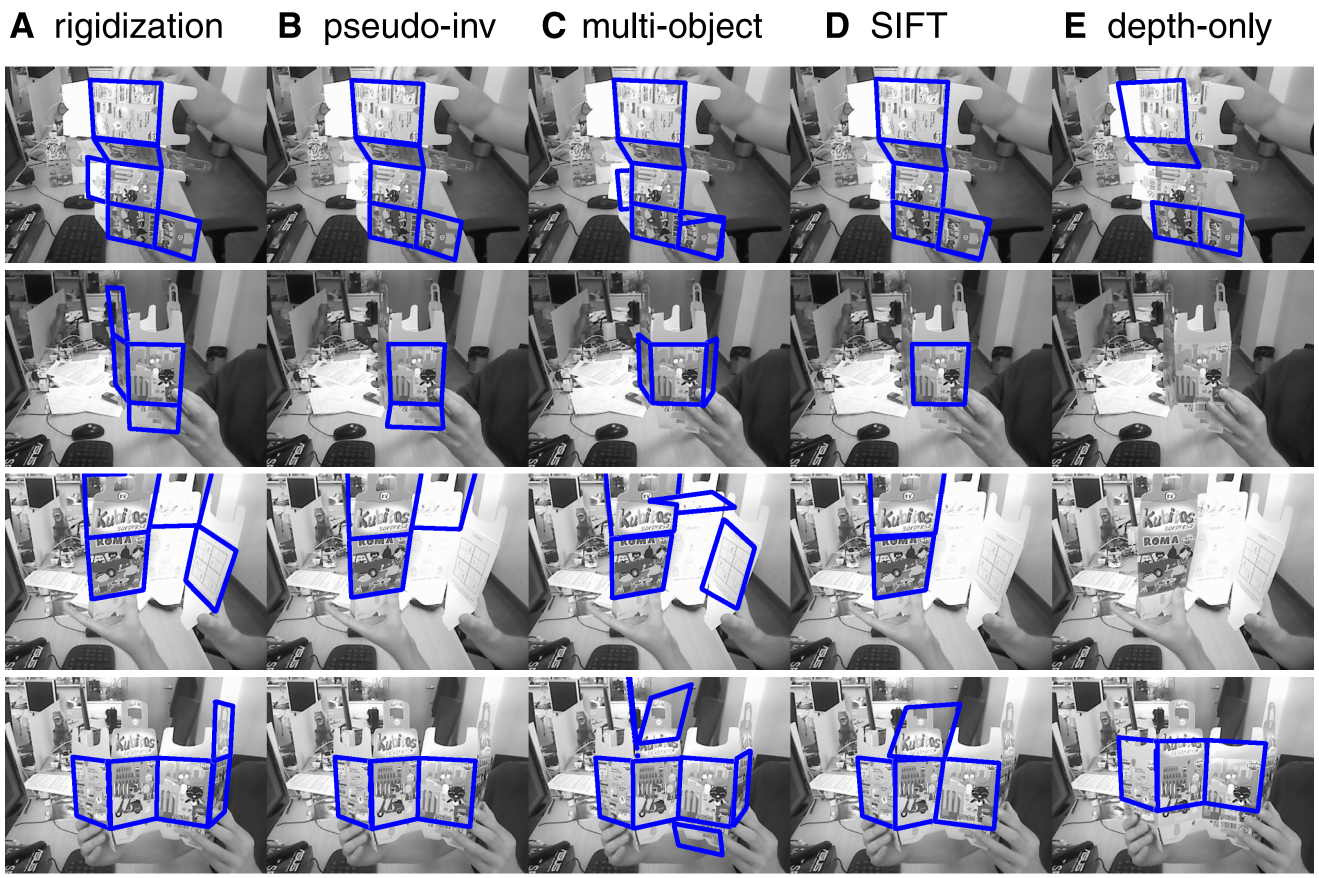

Figure 5. Articulated poses estimated (and considered reliable) by the five algorithms on example frames from the *wave* (top row), *box* (second row), and *various* (third and bottom rows) sequences.

[11] B. Jacquet, R. Angst, and M. Pollefeys. Articulated and restricted motion subspaces and their signatures. In *CVPR*, pages 1506–1513, 2013.

[12] N. Kyriazis and A. Argyros. Physically plausible 3D scene tracking: The single actor hypothesis. In *CVPR*, pages 9–16, 2013.

[13] V. Lepetit and P. Fua. Monocular model-based 3D tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision*, 1:1–89, 2005.

[14] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. NVIDIA Tesla: A unified graphics and computing architecture. *IEEE Micro*, 28(2):39–55, 2008.

[15] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *P. Roy. Soc. B-Biol. Sci.*, 208:385–397, 1980.

[16] F. Mosteller and J. Tukey. *Data analysis and regression: A second course in statistics*. Addison-Wesley Reading, Mass., 1977.

[17] R. M. Murray and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[18] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *ICCV*, pages 2088–2095, 2011.

[19] K. Pauwels, L. Rubio, J. Díaz Alonso, and E. Ros. Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *CVPR*, pages 2347–2354, 2013.

[20] K. Pauwels, M. Tomasi, J. Díaz, E. Ros, and M. Van Hulle. A comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features. *IEEE Transactions on Computers*, 61(7):999–1012, 2012.

[21] S. Pellegrini, K. Schindler, and D. Nardi. A generalisation of the ICP algorithm for articulated bodies. In *BMVC*, 2008.

[22] G. Pons-Moll, A. Baak, J. Gall, L. Leal-Taixe, M. Muller, H. Seidel, and B. Rosenhahn. Outdoor human motion capture using inverse kinematics and von Mises-Fisher sampling. In *ICCV*, pages 1243–1250, 2011.

[23] 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 *CVPR*, pages 1297–1304, June 2011.

[24] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[25] C. Yang and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.