Project Acronym:	GRASP
Project Type:	IP
Project Title:	Emergence of Cognitive Grasping through Introspection, Emulation and Surprise
Contract Number:	215821
Starting Date:	01-03-2008
Ending Date:	28-02-2012

Deliverable Number:	D25
Deliverable Title :	Extracting constraints on the detection, tracking and representation of a human hand by observing its interaction with known geometrical structures
Type:	PU
Authors	A.A. Argyros, N. Kyriazis, I. Oikonomidis
Contributing Partners	FORTH

Contractual Date of Delivery to the EC:	28-02-2012
Actual Date of Delivery to the EC:	28-02-2012

Contents

1 Executive summary **5**

A Attached papers **7**

Chapter 1

Executive summary

Deliverable D25 presents the fourth year developments within WP1 - "Learning to Observe Human Grasping and Consequences of Grasping". According to the Technical Annex, deliverable D25 presents the activities in the context of Tasks 1.3, 1.4 and 1.5:

- [**Task 1.3**] Observing humans: Definition and development of a system that detects and tracks humans and their movements in particular. Activities in this task will focus on the important problem of acquiring real 3D motion of the arms while the human is interacting with objects. The tracking should be successful also in cases when the robot does not have a frontal view of the human.
- [**Task 1.4**] Observing human grasping: Definition and development of a computational method that detects, tracks and represents human hands in action. The derived representation includes aspects and features in the full 4D spatiotemporal space (3D space and time dimensions). The aim is to extract from a sequence of stereoscopic hand observations, the information that is necessary and sufficient for subsequent (WP2) parsing and interpretation of observed hand activities that, in turn, support future repeats by a robotic hand. Activities within this task will address important subproblems such as figure-ground segmentation (environmental modelling, motion/colour based segmentation, coarse object categorisation) tracking humans/hands in 2D/3D (feature selection, hand models, representation of prior knowledge of motion models, prediction and search strategies), etc. [**Task 1.4**] Observing consequences of grasping: The observation of human grasping is considered in the context defined by the interaction of a human hand with its ecological niche. The aim is to derive valuable constraints on the observation of a hand through its interaction with known environmental structures and to extract properties of environmental structures as a consequence of the knowledge regarding the activities in which a hand is engaged.

The work in this deliverable relates to the following third year Milestone:

- [**Milestone 11**] Integration and evaluation of scenarios on multiple experimental platforms, demonstration of cognitive capabilities of robots.

Still, the WP1 work carried out during the 4th year is highly relevant to other project milestones:

- [**Milestone 2**] Definition of initial ontology based on human studies; acquisition (perception and formalisation) of knowledge through hand-environment interaction.
- [**Milestone 4**] Analysis of action-specific visuo-spatial processing, vocabulary of human actions/interactions for perception of task relations and affordances.
- [**Milestone 6**] Integration and evaluation of human hand and body tracking on active robot heads, demonstration of a grasping cycle on the experimental platforms.
- [**Milestone 7**] Observing consequences of grasping; vocabulary of robot action/interactions and definition of a hierarchical structure of features.

The progress in WP1 is presented in the below summarized scientific publications, attached to this deliverable.

- In Attachment A, we first observe that due to occlusions, the estimation of the full pose of a human hand interacting with an object is much more challenging than pose recovery of a hand observed in isolation. In this work we formulate an optimization problem whose solution is the 26-DOF hand pose together with the pose and model parameters of the manipulated object. Optimization seeks for the joint hand-object model that (a) best explains the incompleteness of observations resulting from occlusions due to hand-object interaction and (b) is physically plausible in the sense that the hand does not share the same physical space with the object. The proposed method is the first that solves efficiently the continuous, full-DOF, joint hand-object tracking problem based solely on camera input. Additionally, it is the first to demonstrate how hand-object interaction can be exploited as a context that facilitates hand pose estimation, instead of being considered as a complicating factor. Extensive quantitative and qualitative experiments with simulated and real world image sequences as well as a comparative evaluation with a state-of-the-art method for pose estimation of isolated hands, support the above findings.

- In Attachment B, we present a novel solution to the problem of recovering and tracking the 3D position, orientation and full articulation of a human hand from markerless visual observations obtained by a Kinect sensor. We treat this as an optimization problem, seeking for the hand model parameters that minimize the discrepancy between the 3D structure and appearance of hypothesized instances of a hand model and actual hand observations. This optimization problem is effectively solved using a variant of Particle Swarm Optimization (PSO). The proposed method does not require special markers and/or a complex image acquisition setup. Being model based, it provides continuous solutions to the problem of tracking hand articulations. Extensive experiments with a prototype GPU-based implementation of the proposed method demonstrate that accurate and robust 3D tracking of hand articulations can be achieved in near real-time (12Hz).

- In Attachment C, we propose a method that relies on markerless visual observations to track the full articulation of two hands that interact with each-other in a complex, unconstrained manner. We formulate this as an optimization problem whose 54-dimensional parameter space represents all possible configurations of two hands, each represented as a kinematic structure with 26 Degrees of Freedom (DoFs). To solve this problem, we employ Particle Swarm Optimization (PSO), an evolutionary, stochastic optimization method with the objective of finding the two-hands configuration that best explains the RGB-D observations provided by a Kinect sensor. To the best of our knowledge, the proposed method is the first to attempt and achieve the articulated motion tracking of two strongly interacting hands. Extensive quantitative and qualitative experiments with simulated and real world image sequences demonstrate that an accurate and efficient solution of this problem is indeed feasible.

- In Attachment D, we start by observing that a dynamic scene and, therefore, its visual observations are invariably determined by the laws of physics. We demonstrate an illustrative case where physical explanation, as a vision prior, is not a commodity but a necessity. By considering the problem of ball motion estimation we show how physics-based simulation in conjunction with visual processes can lead to the reduction of the visual input required to infer physical attributes of the observed world. Even further, we show that the proposed methodology manages to reveal certain physical attributes of the observed scene that are difficult or even impossible to extract by other means. A series of experiments on synthetic data as well as experiments with image sequences of an actual ball, support the validity of the proposed approach. The use of generic tools and the top-down nature of the proposed approach make it general enough to be a likely candidate for handling even more complex problems in larger contexts.

- In Attachment E, We present a generic computational framework that exploits GPU processing to cope with the significant computational requirements of a class of model-based vision problems. We study the structure of this class of problems and map the involved processes to contemporary GPU architectures. The proposed framework has been validated through its application to various instances of the problem of model-based 3D hand tracking. We show that through the exploitation of this framework near real-time performance is achieved in problems that are prohibitively expensive to solve on CPU-only architectures. Additional experiments performed in various GPU architectures demonstrate the scalability of the approach and the distribution of the execution time among the involved processes.

- Finally, Attachments F and G, present two posters that have been submitted to the COGSYS'2012 conference and which summarize our work on model based tracking of hand articulations.

Appendix A

Attached papers

[A] 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 Proceedings of the 13th IEEE International Conference on Computer Vision (oral presentation), ICCV2011, Barcelona, Spain, Nov. 6-13, 2011.

[B] I. Oikonomidis, N. Kyriazis and A.A. Argyros, Efficient model-based 3D tracking of hand articulations using Kinect, in Proceedings of the 22nd British Machine Vision Conference, BMVC2011, University of Dundee, UK, Aug. 29-Sep. 1, 2011.

[C] I. Oikonomidis, N. Kyriazis and A.A. Argyros, Tracking the Articulated Motion of Two Strongly Interacting Hands, submitted to the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2012, under review).

[D] N. Kyriazis, I. Oikonomidis and A.A. Argyros, Binding vision to physics based simulation: The case study of a bouncing ball, in Proceedings of the 22nd British Machine Vision Conference, BMVC2011, University of Dundee, UK, Aug. 29-Sep. 1, 2011.

[E] N. Kyriazis, I. Oikonomidis, A.A. Argyros, A GPU-powered computational framework for efficient 3D model-based vision, Technical Report TR420, Jul. 2011, ICS-FORTH, 2011.

[F] I. Oikonomidis, N. Kyriazis and A.A. Argyros, "Efficient model-based tracking of the articulated motion of hands", poster submission at COGSYS'2012, under review.

[G] N. Kyriazis, I. Oikonomidis, A.A. Argyros, A GPU-powered computational framework for efficient 3D model-based vision, poster submission at COGSYS'2012, under review.

Full DOF Tracking of a Hand Interacting with an Object by Modeling Occlusions and Physical Constraints

Iason Oikonomidis, Nikolaos Kyriazis, Antonis A. Argyros
Institute of Computer Science - FORTH and Comp. Science Department, Univ. of Crete
Heraklion, Crete, Greece
{oikonom,kyriazis,argyros}@ics.forth.gr

Abstract

Due to occlusions, the estimation of the full pose of a human hand interacting with an object is much more challenging than pose recovery of a hand observed in isolation. In this work we formulate an optimization problem whose solution is the 26-DOF hand pose together with the pose and model parameters of the manipulated object. Optimization seeks for the joint hand-object model that (a) best explains the incompleteness of observations resulting from occlusions due to hand-object interaction and (b) is physically plausible in the sense that the hand does not share the same physical space with the object. The proposed method is the first that solves efficiently the continuous, full-DOF, joint hand-object tracking problem based solely on markerless multicamera input. Additionally, it is the first to demonstrate how hand-object interaction can be exploited as a context that facilitates hand pose estimation, instead of being considered as a complicating factor. Extensive quantitative and qualitative experiments with simulated and real world image sequences as well as a comparative evaluation with a state-of-the-art method for pose estimation of isolated hands, support the above findings.

1. Introduction

The estimation of the full pose of hands from markerless visual observations is a problem whose solution is of fundamental importance in numerous applications including but not limited to the visual perception of grasping and manipulation, sign language understanding, human computer interaction, etc. It is also known that a number of cascading issues such as the dimensionality of the problem, the incomplete and/or ambiguous observations due to scene clutter and occlusions as well as the requirement for accurate estimates in real time, hinder its effective solution.

Full DOF hand pose recovery during hand-object interaction is a much more interesting problem but also more difficult due to the induced hand-object occlusions. A common approach is to treat the object as a "distractor" that nevertheless leaves some partial evidence which is enough for hand pose estimation. On the contrary, our goal is to see the manipulated object as a source of useful constraints.

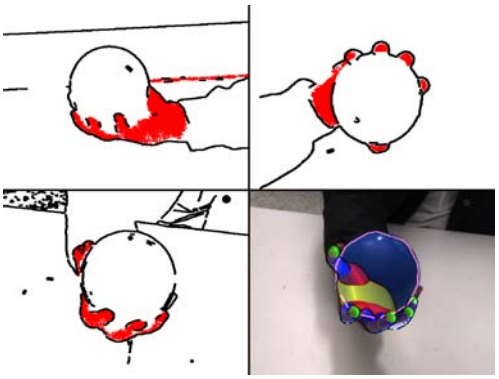


Figure 1. Top row, and bottom left: Three views of a hand grasping an object. Skin regions appear in red and edges in black. The hand is partially occluded by the object in all views. The incomplete skin and edge maps of the hand facilitate the generation of a hypothesis for a hand manipulating a compact sphere. At the same time, given this hypothesis, the 3D pose of the hand can be estimated more accurately. Bottom right: the output of the proposed approach superimposed in one of the frames.

In this work, it is assumed that hand-object interaction is observed by a multicamera system. In each of the acquired views, edge and skin color maps form 2D cues of the presence of a hand. Depending on the viewpoint, the presence of an object occludes the performing hand (Fig. 1). This incomplete observation of the hand provides important evidence on the type and pose of the manipulated object. Conversely, attributing missing hand observations (skin color, edges) to the presence of a manipulated object permits a more accurate estimation of the pose of the partially observed hand. Another source of useful constraints stems from the properties of the natural world, i.e., the fact that the hand and the object cannot share the same physical

2011 IEEE International Conference on Computer Vision
978-1-4577-1102-2/11/$26.00

space. Thus, the 3D shape and pose of the object provides important information on the articulation of the hand and vice versa. The tight coupling between "what the hand tells about the object" and "what the object tells about the hand", suggests that we should identify *simultaneously* the hand configuration and the object 3D model and pose that best explain the observed scene holistically. In this spirit, we formulate an optimization problem that takes into account the constraints mentioned above. Thus, the solution to this problem is a joint hand-object model that besides being compatible to the available visual observations, is also physically plausible.

1.1. Related work

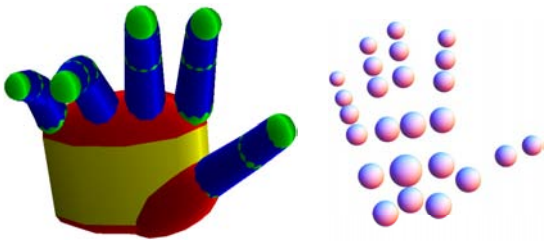
The recovery of the full 3D configuration of articulated objects such as humans and hands presents a lot of challenges. Several approaches have been proposed that address various aspects of the problem such as its dimensionality, the incomplete and/or ambiguous observations due to scene clutter, its computational requirements, etc. Moeslund et al. [15] provide a review of research to the general problem of visual human motion capture and analysis. A review that is specific to the problem of human hand motion estimation is provided in [7]. Related methods are categorized as partial or full pose estimation methods, depending on the level of detail they provide regarding the observed hand.

Another categorization identifies appearance-based and model-based methods. Appearance-based methods estimate hand configurations by establishing a direct mapping of image features to the hand configuration space [3, 22, 28, 20]. Model-based approaches employ a 2D or a 3D hand model [19, 25, 26, 6, 16]. In the case of 3D hand models, the hand pose is estimated by matching the projection of the model to the observed image features. The task is then formulated as a search problem in a high dimensional configuration space, which typically induces a high computational cost. In our work presented in [17], we proposed an efficient (15 fps) method for tracking the articulation of a hand, which depends on visual input provided by the Kinect sensor [14]. A common characteristic of all the methods mentioned above is that they consider human hands in isolation. Thus, in the context of hand-object interaction, their accuracy in hand pose estimation is compromised due to the induced hand-object occlusions that affect drastically the completeness of hand observations.

Given the significant role of context in human visual recognition [18], several researchers have attempted to exploit contextual constraints in solving computer vision problems. Closely related to our problem, a few recent works [12, 8, 29, 21] consider context for classifying human-object interaction activities. The related methods can be classified based on whether they refer to the human body or hand and also on whether they provide a detailed 3D model of the actor (human body or hand) and the object. Thus, [8, 29] study the full human body while in interaction with objects. From these, only [29] provides detailed information on human body pose. For the same problem, Gupta et al. [9] and Sigal et al. [23] propose solutions to handling self-occlusions but not occlusions with other objects. Kjellstrom et al. [12] consider hand-object interactions but only for classifying them, without providing a detailed hand and object model. The work of Hamer et al. [10] also addresses the problem of hand-object interaction but depends on the use of a structured light range sensor and does not model the manipulated object. Finally, Romero et al. [21] propose a method for estimating the pose of a hand interacting with an object which is appearance-based. A method that exploits context to provide a detailed 3D model for both hands and objects is missing from the current literature. The proposed method is trying to fill this gap.

Towards this direction, in this work we extend the approach in [16] by considering jointly the hand and the manipulated object. In [16], a generative, multiview method for 3D hand pose recovery is presented. In each of the acquired views, reference features are computed based on skin color and edges. A 26-DOF 3D hand model is adopted. For a given hand configuration, skin and edge feature maps are rendered and compared directly to the respective observations. The discrepancy of a given 3D hand pose to the observations is quantified by an objective function that is minimized through Particle Swarm Optimization (PSO). The whole approach is implemented efficiently on a GPU. In the proposed approach, we do not only seek for the optimal hand model that explains the available hand observations but rather the joint hand-object model that best explains both the available hand/object observations and the occlusions. Additionally, the objective function penalizes hand-object penetration, seeking for a physically plausible solution. It is demonstrated that the aforementioned constraints are very useful towards an accurate solution to this more complex and interesting problem.

The proposed approach is the first model-based method that solves efficiently, the continuous, full-DOF, joint hand-object tracking problem based on markerless camera input. Additionally, it is the first to demonstrate that hand-object interaction is not necessarily a complicating factor towards estimating the configuration of a hand but a context that can be exploited effectively towards a more accurate solution. As an additional result, the method provides a parametric 3D model of the manipulated object together with its 3D position and orientation. This is achieved by exploiting the hand-object occlusions and despite the fact that only a parametric representation of the object's 3D shape is known that is lacking an explicit appearance model. The approach explores an infinite configuration space. Thus, its accuracy is not limited by the size and content of a database of hand



(a) (b)

Figure 2. Graphical illustration of the employed 26-DOF 3D hand model, consisting of 37 geometric primitives (a) and the 25 spheres constituting the hand's collision model (b).

configurations, as e.g. in [20]. The above are supported by qualitative and quantitative experiments with both simulated and real world image sequences as well as by a comparative evaluation with the method in [16].

2. Hand-object pose estimation (*HOPE*)

The problem of joint hand-object pose estimation is formulated as a multidimensional optimization problem. In the following, we present in detail the basic building blocks of the proposed method for joint Hand-Object Pose Estimation (*HOPE*), with emphasis on the employed observation model, joint hand-object 3D model, hypothesis evaluation mechanism and optimization method.

2.1. Computed visual cues

The proposed method operates on sequences of synchronized views acquired by intrinsically and extrinsically calibrated cameras. A set of images acquired from these cameras at the same moment in time is called a *multi-frame*. If $M_i = \{I_1, I_2, \ldots\}$ is a multiframe of a sequence $S = \{M_1, M_2, \ldots\}$, I_j denotes the image from the j-th camera/view at the i-th time step. For each image I of a multiframe M, an edge map $o_e(I)$ is computed through Canny edge detection [4] and a skin color map $o_s(I)$ is computed using the method presented in [2]. As a convention, 1 indicates presence and 0 indicates absence of skin or edges in the respective maps. For each edge map $o_e(I)$, a distance transform $o_d(I)$ is computed. Maps $\{o_s(I), o_d(I)\}$ constitute the observation cues for image I.

2.2. Joint hand-object model

The proposed approach employs a model $m = (h, o)$ that represents jointly a hand h and the manipulated object o. The hand model h consists of a palm and five fingers. The palm is modeled as an elliptic cylinder and two ellipsoids for caps. Each finger consists of three cones and four spheres, except for the thumb that consists of two cones, an ellipsoid and three spheres (Fig. 2, (a)). 25 additional spheres constitute the hand's collision model and are used for checking for hand-hand and hand-object interpenetration (Fig. 2, (b) and Sec. 2.3). The resulting 62 3D geometric primitives of the hand model are different parameterizations of an ellipsoid and a truncated cylinder. The assembly of appropriate homogeneous transformations of these two geometric primitives yields a hand model similar to that of [25].

The kinematics of each finger is modeled using four parameters encoding angles, two for the base of the finger and two for the remaining joints. Ranges of parameter values are determined based on anatomical studies [1]. The position of a fixed point on the palm defines the global position of the hand. The global orientation is parameterized using the redundant representation of quaternions. This parameterization results in a 26-DOF model encoded in a vector of 27 parameters.

For representing an object, in principle, any parametric model o can be used. The representation of common handheld objects such as cuboids, ellipsoids and cylinders requires 3, 3 and 2 intrinsic shape parameters, respectively. More complex parametric shape models like superquadrics require as many as 6 parameters. Regardless of the intrinsic shape parameterization, 7 additional parameters are required, 3 for 3D position and 4 for a quaternion-based representation of 3D orientation. In this work, we provide experimental results with ellipsoids, cuboids and cylinders. Nevertheless, there is no inherent limitation that prevents the method from being able to handle more complex object models, provided that this does not increase the dimensionality of the problem prohibitively. Interestingly, a complex, known 3D object represented as a mesh has less DOFs compared to our parametric object models (6 DOFs, 3D pose).

2.3. Evaluation of hand-object model hypotheses

Given a joint parametric hand-object model $m = (h, o)$, the goal is to estimate the parameters that give rise to the hand-object configuration that (a) is most compatible to the image features present in multiframe M (Sec. 2.1) and (b) is physically plausible in the sense that two different rigid bodies cannot share the same physical space (interpenetration constraints). To achieve this, an objective function $O(m, M)$ is defined as:

$$O(m, M) = \sum_{I \in M} D(I, m) + \lambda_k W(m). \qquad (1)$$

In Eq.(1), the first term quantifies the discrepancies of a given hand-object model m to the actual camera-based observations, while the second term quantifies the penetration depth between the hand and the object, but also among hand parts (fingers, palm, etc). λ_k is a weighting factor experimentally set to $\lambda_k = 0.1$.

To compute $D(I, m)$, we first compute comparable image features from each hypothesized hand-object model.

More specifically, an edge map $r_e(m)$ and a skin color map $r_s(m)$ can be generated my means of rendering. The reference implementation of the rendering process is similar to that of [25]. The implicit assumption made at this point is that an object cannot contain skin-colored pixels. Thus, the hand component h of m contributes to the skin color map $r_s(m)$ by setting visible hand pixels to 1, while the object component o of m contributes to the skin color map $r_s(m)$ by setting map pixels to 0. Experimental results have verified that the presence of a moderate number of skin-colored pixels on the object's surface does not affect the accuracy of the method. $D(I,m)$ is then defined as:

$$D(I,m) = 1 - \frac{2\sum o_s(I) \wedge r_s(m)}{(\sum o_s(I) \wedge r_s(m)) + (\sum o_s(I) \vee r_s(m))} + \lambda \frac{\sum o_d(I) \cdot r_e(m)}{\sum r_e(m) + \epsilon}, \quad (2)$$

where $o_s(I), o_d(I)$ are defined in Sec. 2.1 and ϵ is a small quantity to prevent division by zero. The 1st row of Eq.(2) models the discrepancies between the skin-colored pixels of the model and the observations. Sums are computed over entire feature maps. In contrast to [16], this part of the objective function is normalized to the interval $[0..1]$. The 2nd row models the discrepancies between the rendered edge maps and the observed edge maps. This is achieved by summing the values of the distance-transformed observation edge map that concur with the edges of the rendered model. λ is a constant normalization factor that was set to 0.02 in all experiments.

The role of function $W(m)$ in Eq.(1) is to penalize (a) hand configurations where hand parts intersect each other (self-penetration) and (b) hand-object configurations where the hand h intersects the object o (interpenetration). Let $P(p_i, p_j)$ be the minimum magnitude 3D translation that is required so that the volume of intersection of geometric primitives p_i and p_j becomes equal to 0. This is effectively computed using the *Open Dynamics Engine* (ODE) [24]. Let also S_h be the primitives of the hand's collision model, as shown in Fig.2(b). The self-penetration P_{hh} of a given hand configuration is defined as $P_{hh} = \max_{i \in S_h, j \in S_h, i \neq j}\{P(i,j)\}$. The interpenetration P_{ho} is similarly defined as $P_{ho} = \max_{i \in S_h}\{P(i,o)\}$. Then, $W(m)$ is defined as $W(m) = \max\{P_{hh}, P_{ho}\}$. Thus, both self- and inter- penetrations are treated in a uniform manner. Additionally, self-penetration is treated more systematically compared to [16] where only certain abduction-adduction angles between adjacent fingers were penalized.

2.4. Optimization

The minimization of the objective function of Eq.(1) is achieved through PSO. Introduced by Kennedy et al. [11], PSO achieves optimization through a policy which emulates the "social interaction" of a population of atoms (particles) that evolves in a number of generations. A population is essentially a set of particles that lie in the parameter space of the objective function to be optimized.

Following the notation introduced in [27], every particle holds its current position (current candidate solution, set of parameters) in a vector x_t and its current velocity in a vector v_t. The ith particle stores in vector p_i the position which corresponds to the best evaluation of its objective function up to the current generation t. All particles of the swarm become aware of the current global optimum p_g, the best position encountered across all particles of the swarm. In every generation t, the velocity of each particle is updated according to $v_t = K(v_{t-1} + c_1 r_1 (p_i - x_{t-1}) + c_2 r_2 (p_g - x_{t-1}))$ and its position according to $x_t = x_{t-1} + v_t$. In the above equations, K is a constant constriction factor [5], c_1 is called the cognitive component, c_2 is termed the social component and r_1, r_2 are random samples of a uniform distribution in $[0..1]$. Finally, $c_1 + c_2 > 4$ must hold [5]. In all performed experiments the values $c_1 = 2.8$, $c_2 = 1.3$ and $K = 2/|2 - \psi - \sqrt{\psi^2 - 4\psi}|$ with $\psi = c_1 + c_2$ were used.

The search space is a multidimensional cuboid. The particle positions are initialized randomly and the particle velocities are set to zero. If, during the position update, a velocity component forces the particle to move outside the search space, this component is zeroed and the particle does not perform any move at the corresponding dimension. The final outcome of the PSO is p_*, the particle with the best score across all generations.

The search space of *HOPE* is the joint hand-object model parameter space m. Given a hand model represented by 27 parameters and an object model represented by d parameters, the search space has $(27 + d)$ dimensions. The objective function to be minimized is $O(m, M)$ and the population is a set of hypothesized 3D hand-object configurations. The outcome of PSO $p_* = m_* = \text{argmin}_m(O(m, M))$ represents the best guess of the algorithm for the joint hand-object model parameters m given the multiframe M.

In the first multiframe, tracking is manually initialized by placing the hand roughly at a predefined position and pose. We have verified experimentally that model estimation is successful with discrepancies of up to $10cm$ in position and up to $20deg$ in global hand pose. Finger joints are correctly estimated if the above constraints are met. To cope with the tracking of the hand-object configuration in time, temporal continuity is exploited. The solution for multiframe M_{t-1} is used to bootstrap the initial population for the optimization problem of M_t. The first member of the population m_{ref} for M_t is the solution for M_{t-1}; The rest of the population consists of perturbations of this solution. The optimization for multiframe M_t is executed for a fixed amount of generations/iterations. After all generations have evolved, the best hypothesis m_* is dubbed as the solution for time step t.

3. Experimental Evaluation

The proposed method has been validated extensively based on both synthetic and real-world sequences of multiframes. First, we demonstrate the accuracy and the computational performance of the proposed (*HOPE*) method on a synthetically rendered data set where hands perform different grasps on a variety of objects (Sec. 3.1). We also compare the performance of *HOPE* to that of the method in [16], hereafter abbreviated as *PEHI* (Pose Estimation of Hands in Isolation). A final experiment with synthetic data involves the application of *HOPE* to a data set showing hands in isolation. The goal of this experiment is to show that *HOPE* can also estimate the pose of hands in isolation effectively, as a special case.

Besides the synthetic data, we also provide qualitative evidence on how the *HOPE* and *PEHI* perform on real sequences of multiframes (Sec. 3.2). Although ground truth information is not available, these indicative results confirm the superiority of *HOPE* over *PEHI* which is in accordance with the experimental results over synthetic data.

3.1. Experiments on synthetic data

Experiments with synthetically produced sequences of multiframes were performed to enable the assessment of the proposed method based on ground truth data. To that end, we simulated different grasps of three different objects (an ellipsoid, a cylinder, and a box) performed by the employed hand model (Sec. 2.2). The interaction of the hand with each of these three objects was observed by 8 virtual cameras surrounding the scene. This resulted in three sequences consisting of 116 multiframes of 8 frames, each. The required cue maps (edges, skin color) were synthesized through rendering (Sec. 2.2).

For the quantitative evaluation of the method, an error metric quantifying the discrepancy between a true hand pose and an estimated hand pose is required. This was computed as follows. The five fingertips as well as the center of the palm were selected as reference points. For each such reference point, the Euclidean distance between its estimated position and its ground truth position is first calculated. These distances are averaged across all multiframes of each sequence, and all sequences. This results in a single error value $\mathcal{D}$ for the whole dataset.

Figures 3(a) and (c) illustrate the estimated error $\mathcal{D}$ of the *HOPE* method as a function of the PSO parameters. In Fig. 3(a), $\mathcal{D}$ is plotted as a function of the number of PSO generations and particles per generation, for multiframes consisting of 2 views. The cameras providing these two views are placed opposite to each other. $\mathcal{D}$ takes values between $22mm$ and $55mm$. It can be verified that for more than 30 generations and more than 32 particles/generation the error in 3D hand pose recovery for *HOPE* does not vary considerably and it is in the order of $25mm$.

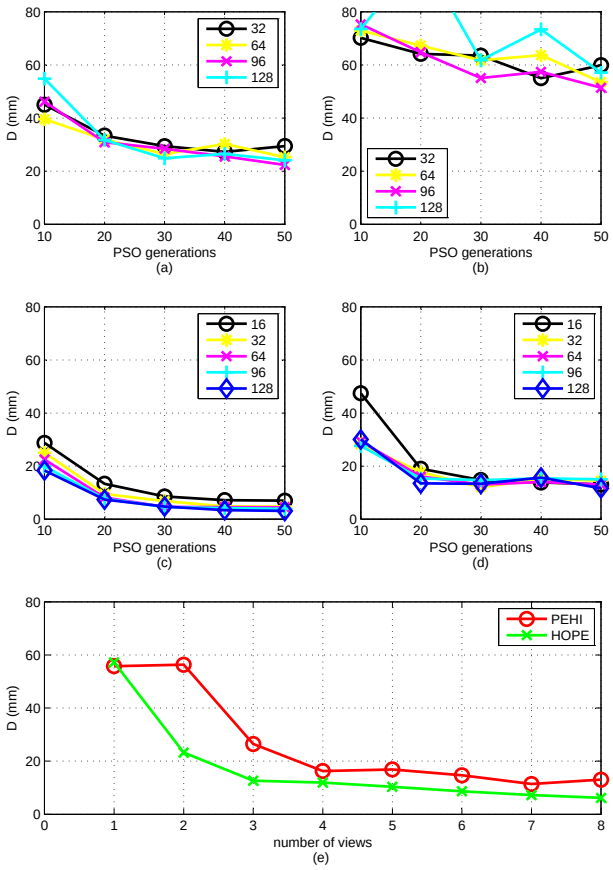


Figure 3. Mean error $\mathcal{D}$ for hand pose estimation (in mm) for *HOPE* (left) and *PEHI* (right) for different PSO parameters and number of views. (a),(b): Varying PSO particles and generations for 2 views. (c),(d): Same as (a),(b) for 8 views. (e): Increasing number of views, 40 generations, 64 particles/generation.

Figure 3(b) is analogous to that of Fig. 3(a) for the *PEHI* algorithm. In this case, the mean error $\mathcal{D}$ does not decrease monotonically as a function of particles. This is attributed to the incomplete/occluded hand observations that undermine the convergence of *PEHI*. $\mathcal{D}$ now ranges between $51mm$ and $101mm$. It can be verified that for more than 30 generations and more than 32 particles/generation the error in 3D hand pose recovery for *PEHI* is in the order of $55mm$. Thus, the error of *PEHI* is on average more than twice the error of *HOPE*.

Figures 3(c) and (d) are analogous to Figs. 3(a) and (b), except the fact that each multiframe now consists of 8 rather than 2 views. $\mathcal{D}$ takes values between $3mm$ and $29mm$ for *HOPE* and between $12mm$ and $47mm$ for *PEHI*. For more that 30 PSO generations and more than 32 particles per generation the error of *PEHI* is still more than twice the error of *HOPE*. Interestingly, whatever *HOPE* achieves with 16 particles and 20 generations is equal or better to what *PEHI* achieves with any of the tested particles/generations combi-

Table 1. Estimated/actual parameters for the object models in the experiments with synthetic data.

Object	Estimated/Actual parameters (in mm)
Cylinder	Radius: 54/55, Height: 127/128
Ellipsoid	X: 54/55, Y: 83/85, Z: 126/128
Box	X: 77/77, Y 128/129, Z: 155/156

nations.

In order to better assess the behavior of the method with respect to the number of available views, additional experiments with a varying number of views were conducted. Figure 3(e) shows the behavior of *HOPE* and *PEHI* as a function of the size of a multiframe. For the experiments with less than 8 views, these were selected empirically to be as complementary as possible. PSO involved 64 particles running for 40 generations. The obtained results demonstrate that modeling the occluder and the physical constraints is more beneficial than adding an extra camera. As an example, exploiting these constraints with two cameras is still better than with three cameras and the hand alone. In fact, whatever *HOPE* achieves with three views is already better to what *PEHI* achieves with as many as eight.

Overall, the experiments in Fig. 3 show a consistent and significant superiority of *HOPE* over *PEHI* which is dominant in the case of a limited number of available views. This is important because it allows for practical joint hand-pose estimation by a multicamera system with a few cameras that is associated with less costs, complexity and requirements for computational resources.

Besides its superiority in hand pose estimation, *HOPE* also estimates the model parameters of the manipulated object. The average positional error of object detection across all sequences of multiframes in the experiments of Fig. 3 is $3mm$ (Euclidean distance between true and estimated positions) and the average orientation error is $2\,\mathrm{deg}$. Table 1 shows the actual and estimated object parameters. The later are averaged for all the multiframes of the sequence that depicts the corresponding object. It can be verified that for all types of objects, the estimated model parameters are very close to the ground truth.

The runtime[1] of a GPU-powered implementation of *HOPE* [13] for runs of 40 PSO generations and 64 particles per generation is 0.31sec for a single-view multiframe and 2.19sec for an 8-view multiframe. An online version of the system employing 4 cameras, operates at 2 fps.

In multiframes of sizes larger than 2, *PEHI* is approximately 20% faster than *HOPE*. This overhead is attributed to the computation of the $W(m)$ component of the objective function. Since this is a fixed overhead that is independent

[1]Experiments run on the computational infrastructure presented in Sec.3.2.

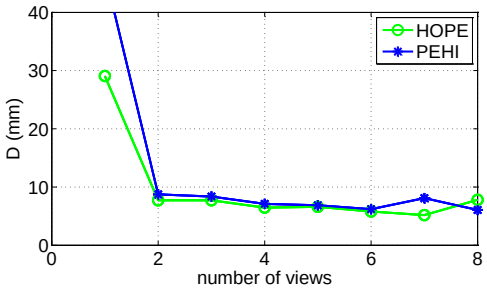


Figure 4. Performance of *HOPE* and *PEHI* on a synthetic sequence of multiframes that shows hands in isolation. 64 PSO particles and 40 generations have been used in both cases.

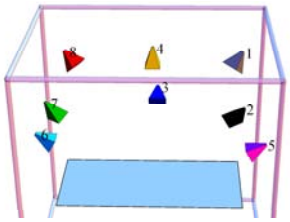


Figure 5. Camera setup for the experiments with real data.

of the multiframe size, the relative difference in computational performance decreases with the number of views.

Finally, we applied both *HOPE* and *PEHI* to a synthetic image sequence (400 multiframes, 8 frames/multiframe) showing non-rigid motion of hands in isolation. Figure 4 plots the mean error $\mathcal{D}$ as a function of the number of the employed views. For both algorithms, 40 PSO generations and 64 particles per generation were used. For *HOPE*, a cylindrical object has been hypothesized. The result shows that the performance of the two algorithms is comparable, a fact that indicates the capability of *HOPE* to track hands observed in isolation. Expectedly, *HOPE* estimated the presence of very small objects (size in the order of a few mms).

3.2. Experiments on real image data

Real-world image sequences were acquired using a multicamera system (Fig. 5) installed around a $2 \times 1m^2$ bench and consisting of 8 synchronized and calibrated *Flea2* PointGrey cameras. Each camera has a maximum framerate of 30 fps, at 1280×960 image resolution. However, the core processing is performed on 256×256 windows centered around the previous multiframe solution. The workstation for image acquisition and processing is equipped with a quad-core Intel i7 920 CPU, 6 GBs RAM and a 1581 $GFlops$ Nvidia GTX 580 GPU with 1.5 GBs RAM.

Three sequences of multiframes have been acquired, each showing a hand grasping and manipulating a spherical (301 multiframes), a cylindrical (261 multiframes), and a box (251 multiframes) object. Figure 6(a) provides sample results obtained by applying *HOPE* (top row) and *PEHI*

Table 2. Estimated/actual parameters for the object models in the experiments of Fig. 6.

Object	Estimated/actual parameters (in mm)
Cylinder	Radius: 51/53, Height: 121/131
Ellipsoid	X: 128/116, Y: 128/116, Z: 122/116
Box	X: 66/67, Y: 158/150, Z: 84/93

Table 3. The mean value of the objective function of *HOPE* and its standard deviation when optimization searches for cylinders, ellipsoids and cuboids for a sequence showing an ellipsoid (sphere).

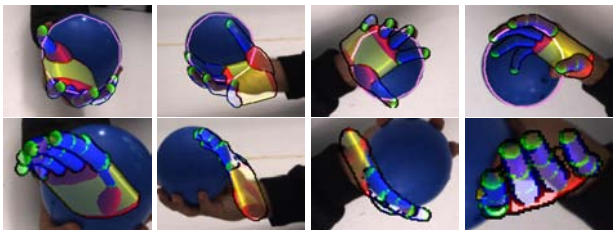
	Cylinder	Ellipsoid	Cuboid
Mean value	3.02	2.65	3.95
Stdev.	0.68	0.57	1.17

(bottom row) to a specific multiframe of the sphere sequence. Since the hand is mostly occluded by the sphere in all views, *HOPE* estimates the hand configuration correctly while *PEHI* fails completely. Similar results were obtained in the case of the cylinder sequence which shows a hand grasping and turning a cylindrical object up-side down. Fig. 6(b) shows four frames acquired from the same camera in different moments in time. *HOPE* tracks the configuration of the hand throughout the whole sequence whereas *PEHI* looses track of the hand as soon as the later becomes severely occluded by the object. Figure 6(c) shows a similar result for the box sequence. Additionally, in Table 2, we compare the object shape parameters estimated by *HOPE* to the actual, physically measured ones, computed by averaging estimations for all multiframes of a given sequence. The standard deviation of these estimations is in the order of a few millimeters. It can be verified that the error in object shape estimation is satisfactory.

For *HOPE*, we also run a simple classification experiment. Although shape classification is not the focus of this work, it provides an indirect indication of the accuracy of the optimization process. For the sphere sequence (Fig. 6(a) and (b)), we ran *HOPE* assuming a cuboid, an ellipsoid and a cylinder. Table 3 shows the mean value and the standard deviation of the objective function of *HOPE* in all multiframes of the sequence. As it can be verified, the hypothesis of an ellipsoid better explains the observed scene. In fact, 98.67% of the multiframes were better explained by the ellipsoid, 1.33% by the cylinder and none by the cuboid.

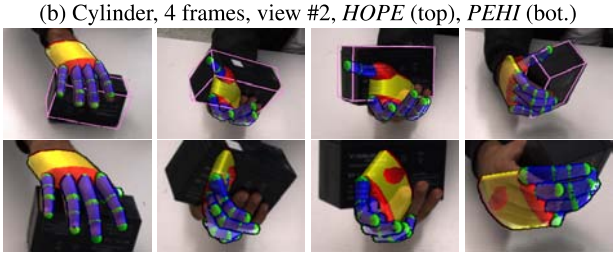
Finally, Fig.7, shows sample snapshots from the results obtained on a sequence of a hand performing fine manipulation of an elongated cuboid. Visual inspection confirms that the accuracy of *HOPE* is quite satisfactory, despite the complex and challenging hand-object interaction. Sample videos out of these experiments are provided as supplemental material to this submission and are also available online[2].

[2]http://www.youtube.com/watch?v=N3ffgj1bBGw



(a) Sphere, frame #103, 4 views, *HOPE* (top), *PEHI* (bot.)

(b) Cylinder, 4 frames, view #2, *HOPE* (top), *PEHI* (bot.)



(c) Box, 4 frames, view #1, *HOPE* (top), *PEHI* (bot.)

Figure 6. Sample frames from the results obtained by *HOPE* and *PEHI* in real-world experiments. For *HOPE* the projection of the estimated 3D object model is shown in pink color.

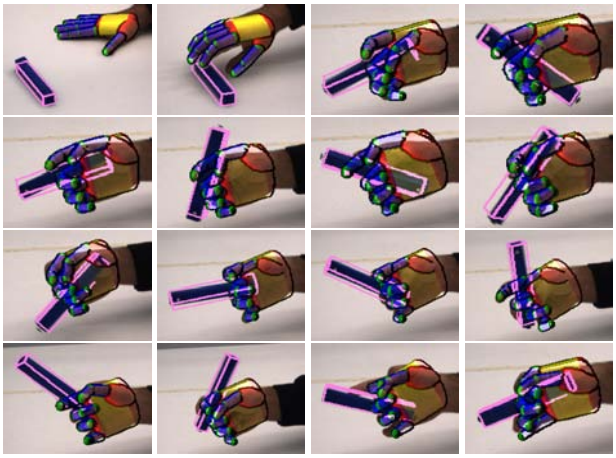


Figure 7. Snapshots from an experiment where a hand performs a complex manipulation of an elongated cuboid.

4. Discussion and conclusions

In a hand-object interaction scenario, the observation of hands provides information that is important to the understanding of the object's state and vice versa. In this paper,

we demonstrated that by considering jointly the hand and the object and by modeling occlusions and physical constraints it is possible to better understand aspects of both. More specifically, the optimization over the parameters of a joint hand-object 3D model results in full hand pose estimation that is performed more accurately compared to methods that consider the hand in isolation. On top of that, a parametric expression of the manipulated object is also computed. PSO is proved very competent in handling the complex, multidimensional and multimodal objective function of this problem. Results from extensive experiments on simulated data demonstrated the potential of the method against ground truth, but also comparatively to the results of a state-of-the-art hand pose estimation method that considers hands in isolation. Experiments in real world sequences exhibit that the proposed method performs well in challenging cases of complex hand articulation and hand-object interaction. Ongoing research investigates the potential of *HOPE* in supporting the interpretation of the semantics of human grasping and manipulation activities.

Acknowledgments

This work was partially supported by the IST-FP7-IP-215821 project GRASP. The contribution of Konstantinos Tzevanidis and Pashalis Padeleris, members of CVRL/FORTH, is gratefully acknowledged.

References

[1] I. Albrecht, J. Haber, and H. Seidel. Construction and animation of anatomically based human hand models. In *2003 ACM SIGGRAPH/Eurographics symposium on Computer Animation*. Eurographics Association, 2003. 3

[2] A. Argyros and M. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *ECCV*, 2004. 3

[3] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR*, volume 2, page 432, Los Alamitos, CA, USA, 2003. IEEE Computer Society. 2

[4] J. Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, 1986. 3

[5] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002. 4

[6] M. de la Gorce, N. Paragios, and D. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *CVPR*, pages 1–8, 2008. 2

[7] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *CVIU*, 108(1-2):52–73, Oct. 2007. 2

[8] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *PAMI*, 31:1775–1789, 2009. 2

[9] A. Gupta, A. Mittal, and L. Davis. Constraint integration for efficient multiview pose estimation with self-occlusions. *PAMI*, 30(3):493 –506, Mar 2008. 2

[10] H. Hamer, K. Schindler, E. Koller-Meier, and L. V. Gool. Tracking a hand manipulating an object. In *ICCV*, Oct 2009. 2

[11] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001. 4

[12] H. Kjellstrom, J. Romero, D. Martinez, and D. Kragic. Simultaneous visual recognition of manipulation actions and manipulated objects. In *ECCV*, 2008. 2

[13] N. Kyriazis, I. Oikonomidis, and A. Argyros. A gpu-powered computational framework for efficient 3d model-based vision. Technical Report 420, FORTH, July 2011. 6

[14] Microsoft Corp. Redmond WA. Kinect for Xbox 360. 2

[15] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *CVIU*, 104(2-3):90–126, Dec 2006. 2

[16] I. Oikonomidis, N. Kyriazis, and A. Argyros. Markerless and efficient 26-dof hand pose recovery. In *ACCV*, 2010. 2, 3, 4, 5

[17] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, Aug 2011. 2

[18] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520 – 527, 2007. 2

[19] J. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *ICCV*, page 612, Los Alamitos, CA, USA, 1995. IEEE Computer Society. 2

[20] J. Romero, H. Kjellstrom, and D. Kragic. Monocular real-time 3d articulated hand pose estimation. *IEEE-RAS Int'l Conf. on Humanoid Robots*, Dec 2009. 2, 3

[21] J. Romero, H. Kjellström, and D. Kragic. Hands in action: Real-time 3d reconstruction of hands in interaction with objects. In *ICRA*, 2010. 2

[22] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3d hand pose reconstruction using specialized mappings. In *ICCV*, 2001. 2

[23] L. Sigal and M. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *CVPR*, volume 2, pages 2041 – 2048, 2006. 2

[24] R. Smith. Open dynamics engine, http://www.ode.org/, 2006. 4

[25] B. Stenger, P. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. *CVPR*, pages II–310–II–315, 2001. 2, 3, 4

[26] E. Sudderth, M. Mandel, W. Freeman, and A. Willsky. Visual hand tracking using nonparametric belief propagation. In *CVPR Wkshp on Generative Model-based Vision*, 2004. 2

[27] B. White and M. Shaw. Automatically tuning background subtraction parameters using particle swarm optimization. In *IEEE ICME*, 2007. 4

[28] Y. Wu and T. S. Huang. View-independent recognition of hand postures. In *CVPR*, pages 88–94, 2000. 2

[29] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, Jun 2010. 2

Efficient Model-based 3D Tracking of Hand Articulations using Kinect

Iason Oikonomidis[1,2]
http://www.ics.forth.gr/~oikonom
Nikolaos Kyriazis[1,2]
http://www.ics.forth.gr/~kyriazis
Antonis A. Argyros[1,2]
http://www.ics.forth.gr/~argyros

[1] Computational Vision and Robotics Lab., Institute of Computer Science, FORTH
[2] Computer Science Department, University of Crete, Greece

Abstract

We present a novel solution to the problem of recovering and tracking the 3D position, orientation and full articulation of a human hand from markerless visual observations obtained by a Kinect sensor. We treat this as an optimization problem, seeking for the hand model parameters that minimize the discrepancy between the appearance and 3D structure of hypothesized instances of a hand model and actual hand observations. This optimization problem is effectively solved using a variant of Particle Swarm Optimization (PSO). The proposed method does not require special markers and/or a complex image acquisition setup. Being model based, it provides continuous solutions to the problem of tracking hand articulations. Extensive experiments with a prototype GPU-based implementation of the proposed method demonstrate that accurate and robust 3D tracking of hand articulations can be achieved in near real-time (15Hz).

1 Introduction

The 3D tracking of articulated objects is a theoretically interesting and challenging problem. One of its instances, the 3D tracking of human hands has a number of diverse applications [6, 14] including but not limited to human activity recognition, human-computer interaction, understanding human grasping, robot learning by demonstration, etc. Towards developing an effective and efficient solution, one has to struggle with a number of complicating and interacting factors such as the high dimensionality of the problem, the chromatically uniform appearance of a hand and the severe self-occlusions that occur while a hand is in action. To ease some of these problems, some very successful methods employ specialized hardware for motion capture [21] or the use of visual markers as in [25]. Unfortunately, such methods require a complex and costly hardware setup, interfere with the observed scene, or both.

Several attempts have been made to address the problem by considering markerless visual data, only. Existing approaches can be categorized into model- and appearance-based. Model-based approaches provide a continuum of solutions but are computationally costly and depend on the availability of a wealth of visual information, typically provided by a

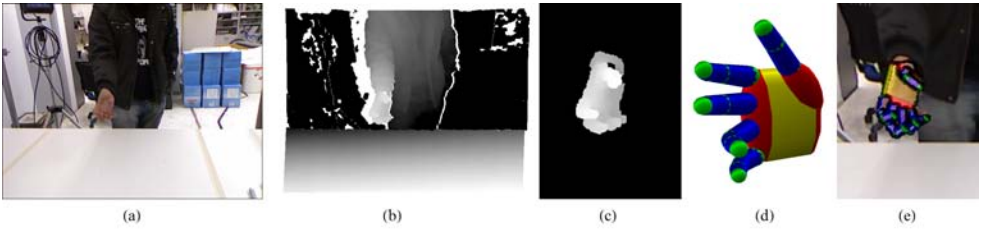


Figure 1: Graphical illustration of the proposed method. A Kinect RGB image (a) and the corresponding depth map (b). The hand is segmented (c) by jointly considering skin color and depth. The proposed method fits the employed hand model (d) to this observation recovering the hand articulation (e).

multicamera system. Appearance-based methods are associated with much less computational cost and hardware complexity but they recognize a discrete number of hand poses that correspond typically to the method's training set.

In this paper, we propose a novel model-based approach to the problem of 3D tracking of hand articulations which is formulated as an optimization problem that minimizes the discrepancy between the 3D structure and appearance of hypothesized 3D hand model instances, and its actual visual observations. Observations come from an off-the-shelf Kinect sensor [13]. Optimization is performed with a variant of an existing stochastic optimization method (Particle Swarm Optimization - PSO). The most computationally demanding parts of the process have been implemented to run efficiently on a GPU. Extensive experimental results demonstrate that accurate and robust tracking is achievable at 15Hz. Thus, to the best of our knowledge, the proposed method is the first that simultaneously (a) provides accurate, continuous solutions to the problem of 3D tracking of hand articulations (b) does not require a complex hardware setup (c) relies solely on markerless visual data (d) is rather insensitive to illumination conditions and (e) runs in near real-time.

1.1 Related work

Moeslund *et al*. [14] provide a thorough review covering the general problem of visual human motion capture and analysis. Human body and human hand pose recovery are problems sharing important similarities such as the tree-like connectivity and the size variability of the articulated parts. A variety of methods have been proposed to capture human hand motion. Erol *et al*. [6] present a review of such methods. Based on the completeness of the output, they differentiate between partial and full pose estimation methods, further dividing the last class into appearance- and model-based ones.

Appearance-based methods typically establish a mapping from a set of image features to a discrete, finite set of hand model configurations [3, 19, 20, 22, 26]. The discriminative power of these methods depends on the invariance properties of the employed features, the number and the diversity of the postures to be recognized and the method used to derive the mapping. Due to their nature, appearance-based methods are well suited for problems such as hand posture recognition where a small set of known target hand configurations needs to be recognized. Conversely, such methods are less suited for problems that require an accurate estimation of the pose of freely performing hands. Moreover, generalization for such methods is achieved only through adequate training. On the positive side, training is

performed offline and online execution is typically computationally efficient.

Model-based approaches [5, 7, 15, 16, 18, 23, 24] generate model hypotheses and evaluate them on the available visual observations. Essentially, this is performed by formulating an optimization problem whose objective function measures the discrepancy between the visual cues that are expected due to a model hypothesis and the actual ones. The employed optimization method must be able to evaluate the objective function at arbitrary points in the multidimensional model parameters space, so, unlike appearance-based methods, most of the computations need to be performed online. The resulting computational complexity is the main drawback of these methods. On the positive side, such methods do not require training and are also more easily extendable.

Another categorization can be defined, based on how partial evidence regarding the individual rigid parts of the articulated object contributes to the final solution. We differentiate among *disjoint evidence methods* that consider individual parts in isolation prior to evaluating them against observations [7, 18, 22, 24] and *joint evidence methods* [3, 5, 15, 16, 19, 20, 23, 26] that consider all parts in the context of full object hypotheses. Disjoint evidence methods usually have lower computational requirements than joint-evidence ones, but need to cope explicitly with the difficult problem of handling part interactions such as collisions and occlusions. In joint-evidence methods, part interactions are effortlessly treated but their computational requirements are rather high. Until recently, the only available joint-evidence methods were appearance-based. As an example, Shotton *et al*. propose in [22] an appearance-based, disjoint evidence method for human body pose estimation with remarkable computational performance.

This paper presents a model-based method that treats 3D hand pose recovery as a minimization problem whose objective function is the discrepancy between the 3D structure and appearance of hypothesized 3D hand model instances, and visual observations of a human hand. Observations come from an off-the-shelf Kinect sensor. Optimization is performed through a variant of PSO tailored to the needs of the specific problem. Other versions of PSO have been employed in the past for human body pose tracking [9] and multicamera-based hand pose estimation [15].

Under the taxonomy of [6], the present work is a full, model-based pose estimation method that employs a single hypothesis. Furthermore, according to the categorization introduced earlier, it is a joint-evidence method. From a methodological point of view, the mostly related existing method [15] treats the problem of 3D hand pose estimation as an optimization problem that is solved through canonical PSO. However, the observations in [15] are 2D silhouettes of a hand extracted from a multicamera system. In the present work, the observation is the RGB plus depth images provided by a Kinect sensor. As a direct consequence, the objective function is different, the computational requirements are much smaller, the required camera setup is greatly simplified and the resulting system can be operational in situations where illumination conditions may vary substantially.

Another closely related work is that of Hamer *et al*. [7]. In both works the input is range data and a model-based approach is adopted. Hamer employs Belief Propagation which is well-suited for specific interdependency patterns among the parameters: the dependency graph must be a tree. Since the fingers usually interact (occlude or touch) with each other, special, explicit handling of such interactions is required. In our work, self-occlusions are naturally and effortlessly treated since we adopt a joint-evidence approach.

2 Tracking hand articulations based on the Kinect

The input to the proposed method (see Fig.1) is an image acquired using the Kinect sensor, together with its accompanying depth map. Skin color detection followed by depth segmentation is used to isolate the hand in 2D and 3D. The adopted 3D hand model comprises of a set of appropriately assembled geometric primitives. Each hand pose is represented as a vector of 27 parameters. Hand articulation tracking is formulated as the problem of estimating the 27 hand model parameters that minimize the discrepancy between hand hypotheses and the actual observations. To quantify this discrepancy, we employ graphics rendering techniques to produce comparable skin and depth maps for a given hand pose hypothesis. An appropriate objective function is thus formulated and a variant of PSO is employed to search for the optimal hand configuration. The result of this optimization process is the output of the method for the given frame. Temporal continuity is exploited to track the hand articulation in a sequence of frames. The remainder of this section describes these algorithmic steps in more detail.

2.1 Observing a hand

The input to the method is a 640×480 RGB color image of a hand and a corresponding depth image, as these are provided by the Kinect sensor [13]. Skin color is detected as in [7] and the resulting largest skin colored blob is kept for further consideration. A conservative estimation of the hands spatial extend is computed by dilating this blob with a circular mask of radius $r = 5$. Given the estimation of the 3D position of the tracked hand for the previous frame, skin colored 3D points that are within a preset depth range (25cm) from that estimation are kept, whereas the remaining depth map is set to zero. The observation model $O = (o_s, o_d)$ that feeds the rest of the process consists of the 2D map o_s of the segmented skin color and the corresponding depth map o_d.

2.2 Modeling a hand

The employed hand model consists of a palm and five fingers. The palm is modeled as an elliptic cylinder and two ellipsoids for caps. Each finger consists of three cones and four spheres, except for the thumb which consists of an ellipsoid, two cones and three spheres. Similarly to [15] we build all the necessary geometric primitives for the hand using two basic 3D geometric primitives, a sphere and a cylinder, enabling a high degree of computational parallelism (see Sec. 2.5). The hand model is depicted in Fig. 1(d) with color-coded geometric primitives (yellow: elliptic cylinders, red: ellipsoids, green: spheres, blue: cones).

The kinematics of each finger is modeled using four parameters encoding angles, two for the base of the finger and two for the remaining joints. Bounds on the values of each parameter are set based on anatomical studies [1]. The global position of the hand is represented using a fixed point on the palm. The global orientation is parameterized using the redundant representation of quaternions. The resulting parameterization encodes a 26-DOF hand model with a representation of 27 parameters.

2.3 Evaluating a hand hypothesis

Having a parametric 3D model of a hand, the goal is to estimate the model parameters that are most compatible to the visual observations (Sec. 2.1). To do so, given a hand pose

hypothesis h and camera calibration information C, a depth map $r_d(h,C)$ is generated by means of rendering. By comparing this map with the respective observation o_d, a "matched depths" binary map $r_m(h,C)$ is produced. More specifically, a pixel of r_m is set to 1 if the respective depths in o_d and r_d differ less than a predetermined value d_m or if the observation is missing (signified by 0 in o_d), and 0 otherwise. This map is compared to the observation o_s, so that skin colored pixels that have incompatible depth observations do not positively contribute to the total score (Sec. 2.3, Eq.(2)).

A distance measure between a hand pose hypothesis h and the observation maps O is established. This is achieved by a function $E(h,O)$ that measures the discrepancy between the observed skin and depth maps O computed for a given frame and the skin and depth maps that are rendered for a given hand pose hypothesis h:

$$E(h,O) = D(O,h,\mathrm{C}) + \lambda_k \cdot kc(h). \tag{1}$$

In Eq.(1), λ_k is a normalization factor. The function D in Eq.(1) is defined as

$$D(O,h,C) = \frac{\sum min(|o_d - r_d|, d_M)}{\sum(o_s \vee r_m) + \varepsilon} + \lambda \left(1 - \frac{2\sum(o_s \wedge r_m)}{\sum(o_s \wedge r_m) + \sum(o_s \vee r_m)}\right). \tag{2}$$

The first term of Eq.(2) models the absolute value of the clamped depth differences between the observation O and the hypothesis h. Unless clamping to a maximum depth d_M is performed, a few large depth discrepancies considerably penalize an otherwise reasonable fit. This fact, in turn, creates large variations of the objective function's value near the optimum, hindering the performance of any adopted optimization strategy. A small value ε is added to the denominator of this term to avoid division by zero. The second term of Eq.(2) models the discrepancies between the skin-colored pixels of the model and the observation. λ is a constant normalization factor. The sums are computed over entire feature maps.

The function kc in Eq.(1) adds a penalty to kinematically implausible hand configurations. An elaborate collision scheme was considered for kc, taking into account all possible pairs of relatively moving hand parts. Experimental results have demonstrated that for the majority of encountered situations, it suffices to penalize only adjacent finger interpenetration. Thus, in the current implementation: $kc(h) = \sum_{p\in \mathrm{Q}} - \min(\phi(p,h),0)$, where Q denotes the three pairs of adjacent fingers, excluding the thumb, and ϕ denotes the difference (in radians) between the abduction-adduction angles of those fingers in hypothesis h. In all experiments, λ was set to 20 and of λ_k to 10. The depth thresholds were set to $d_m = 1cm$ and $d_M = 4cm$.

2.4 Stochastic optimization through particle swarms

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart in [10, 11]. PSO is a stochastic, evolutionary algorithm that optimizes an objective function through the evolution of atoms of a population. A population is essentially a set of particles that lie in the parameter space of the objective function to be optimized. The particles evolve in runs which are called generations according to a policy which emulates "social interaction".

Every particle holds its current position (current candidate solution and kept history) in a vector x_k and its current velocity in a vector v_k. Vector P_k stores the position at which each particle achieved, up to the current generation k, the best value of the objective function. Finally, the swarm as a whole, stores in vector G_k the best position encountered across all particles of the swarm. G_k is broadcast to the entire swarm, so every particle is aware of the

global optimum. The update equations that reestimate each particle's velocity and position in every generation k are

$$v_{k+1} = w(v_k + c_1 r_1 (P_k - x_k) + c_2 r_2 (G_k - x_k)) \tag{3}$$

and

$$x_{k+1} = x_k + v_{k+1}, \tag{4}$$

where w is a constant *constriction factor* [4]. In Eq. (3), c_1 is called the *cognitive component*, c_2 is termed the *social component* and r_1, r_2 are random samples of a uniform distribution in the range $[0..1]$. Finally, $c_1 + c_2 > 4$ must hold [4]. In all performed experiments the values $c_1 = 2.8$, $c_2 = 1.3$ and $w = 2/\left|2 - \psi - \sqrt{\psi^2 - 4\psi}\right|$ with $\psi = c_1 + c_2$ were used.

Typically, the particles are initialized at random positions and zero velocities. Each dimension of the multidimensional parameter space is bounded in some range. If, during the position update, a velocity component forces the particle to move to a point outside the bounded search space, a handling policy is required. A variety of alternative policies have been proposed in the relevant literature [8]. The "nearest point" method was chosen in our implementation. According to this, if a particle has a velocity that forces it to move to a point p_o outside the bounds of the parameter space, that particle moves to the point p_b inside the bounds that minimizes the distance $|p_o - p_b|$.

In this work, PSO operates in the 27-dimensional 3D hand pose parameter space. The objective function to be optimized (i.e., minimized) is $E(O,h)$ (Eq. 1) and the population is a set of candidate 3D hand poses hypothesized for a single frame. Thus, the process of tracking a human hand requires the solution of a sequence of optimization problems, one for each acquired frame. By exploiting temporal continuity, the solution over frame F_t is used to generate the initial population for the optimization problem for frame F_{t+1}. More specifically, the first member of the population h_{ref} for frame F_{t+1} is the solution for frame F_t; The rest of the population consists of perturbations of h_{ref}. The variance of these perturbations is experimentally determined as it depends on the anticipated jerkiness of the observed motion and the image acquisition frame rate. The optimization for frame F_{t+1} is executed for a fixed amount of generations. After all generations have evolved, the best hypothesis h_{best} is dubbed as the solution for time step $t+1$.

The above described PSO variant successfully estimates the $6D$ global pose of the hand. However, the estimation of the 20 remaining parameters that are related to finger angles is not equally satisfactory. The swarm quickly converges to a point close to the optimum in a behavior that in the relevant literature [11] is termed "swarm collapse". However the estimation of the parameters for the fingers often gets stuck to local minima. To overcome this problem and increase accuracy, we employed a PSO variant that performs randomization on the 20 dimensions corresponding to finger joint angles, similar to that suggested in [27]. More specifically, every i_r generations, half of the particles are disturbed, each in a different, randomly chosen finger joint dimension d_r. The value that is assigned to $x_t[d_r]$ is a sample of the uniform distribution in the permitted range for d_r. The value of i_r was set to 3 generations in all experiments.

2.5 GPU acceleration

The most computationally demanding part of the proposed method is the evaluation of a hypothesis-observation discrepancy $E(h,O)$ and, especially, its term D. The computation of

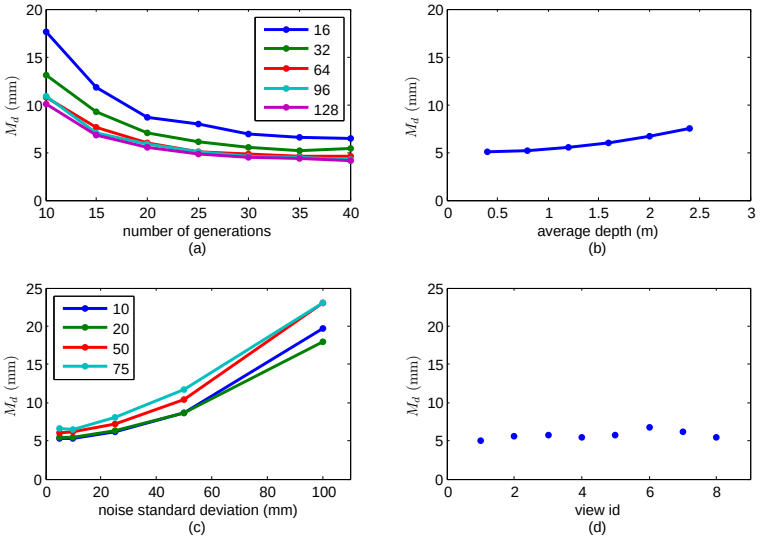


Figure 2: Quantitative evaluation of the performance of the method with respect to (a) the PSO parameters (b) the distance from the sensor (c) noise and (d) viewpoint variation.

D involves rendering, pixel-wise operations between an observation and a hypothesis map and summation over the results. We exploit the inherent parallelism of this computation by performing these operations on a GPU. Furthermore, by evaluating simultaneously the function D for many hypotheses h_i (i.e., for all the particles of a PSO generation), we minimize the overhead of communication between the CPU and the GPU. Hardware instancing is employed to accelerate the rendering process, exploiting the fact that the hand model is made up of transformed versions of the same two primitives (a cylinder and a sphere). The pixel-wise operations between maps are inherently parallel and the summations of the maps are performed efficiently by employing a pyramidal scheme. More details on the GPU implementation are provided in [19].

3 Experimental evaluation

The experimental evaluation of the proposed method was based on synthetic data with ground truth information and on real-world sequences obtained by a Kinect sensor. The proposed method runs on a computer equipped with a quad-core Intel i7 950 CPU, 6 GBs RAM and an Nvidia GTX 580 GPU with 1581 *GFlops* processing power and 1.5 GBs memory. On this system, the average frame rate is 15Hz. As discussed in [19] there is still room for performance improvements.

Synthetic data were used for the quantitative evaluation of the proposed method. This is a common approach in the relevant literature [7, 15] because ground truth data for real-world image sequences is hard to obtain. The employed synthetic sequence consists of 360 consecutive hand poses that encode everyday hand motions as simple as waving and as complex as object grasping. Rendering was used to synthesize the required input O for each considered hand pose. To quantify the accuracy in hand pose estimation, we adopt the metric used in [7].

More specifically, the distance between corresponding phalanx endpoints in the ground truth and in the estimated hand model is measured. The average of all these distances over all the frames of the sequence constitutes the resulting error estimate Δ.

Several experiments were carried out to assess the influence of several factors to the performance of the method. Figure 2(a) illustrates the behavior of the method with respect to the PSO parameters (number of generations and particles per generation). The product of these parameters determines the computational budget of the proposed methodology, as it accounts for the number of objective function evaluations. The horizontal axis of the plot denotes the number of PSO generations. Each plot of the graph corresponds to a different number of particles per generation. Each point in each plot is the median M_d of the error Δ for 20 repetitions of an experiment run with the specific parameters. A first observation is that M_d decreases monotonically as the number of generations increase. Additionally, as the particles per generation increase, the resulting error decreases. Nevertheless, employing more that 25 generations and more than 64 particles results in insignificant improvement of the method's accuracy. The gains, if any, are at most $0.5mm$. For this reason, the configuration of 64 particles for 25 generations was retained in all further experiments.

Another investigation considered the effect of varying the distance of the hand from the hypothesized sensor. This explores the usefulness of the method in different application scenarios that require observations of a certain scene at different scales (e.g., close-up views of a hand versus distant views of a human and his/her broader environment). To do this, we generated the same synthetic sequences at different average depths. The results of this experiment are presented in Fig. 2(b). At a distance of half a meter the error is equal to $5mm$. As the distance increases, the error also increases; Interestingly though, it doesn't exceed $7.5mm$ even at an average distance of $2.5m$.

The method was also evaluated with respect to its tolerance to noisy observations. Two types of noise were considered. Errors in depth estimation were modeled as a Gaussian distribution centered around the actual depth value with the variance controlling the amount of noise. Skin-color segmentation errors were treated similarly to [19], by randomly flipping the label (skin/non-skin) of a percentage of pixels in the synthetic skin mask. Figure 2(c) plots the method's error in hand pose estimation for different levels of depth and skin segmentation error. As it can be verified, the hand pose recovery error is bounded in the range $[5mm..25mm]$, even in data sets very heavily contaminated with noise.

We also assessed the accuracy in hand pose estimation with respect to viewpoint variations. This was achieved by placing the virtual camera at 8 positions dispersed on the surface of a hemisphere placed around the hypothesized scene. The data points of Fig. 2(d) demonstrate that viewpoint variations do not significantly affect the performance of the method.

Several long real-world image sequences were captured using the PrimeSense Sensor Module of OpenNI [17]. The sequences exhibit hand waving, palm rotations, complex finger articulation as well as grasp-like hand motions. The supplemental material accompanying the paper[1] provides videos with the results obtained in two such sequences (1341 and 1494 frames, respectively). Indicative snapshots are shown in Fig. 3. As it can be observed, the estimated hand model is in very close agreement with the image data, despite the complex hand articulation and significant self occlusions.

Finally, besides tracking, we tested the capability of the proposed method to perform automatic hand model initialization, i.e., single-frame hand pose estimation. Essentially, this boils down to the capability of PSO to optimize the defined objective function even when

[1] Also available at http://www.youtube.com/watch?v=Fxa43qcm1C4

Figure 3: Indicative results on real-world data.

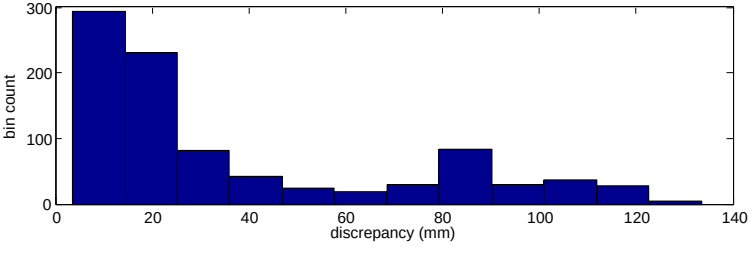


Figure 4: Performance of single-frame hand pose estimation.

parameter ranges are very broad. To do so, the proposed algorithm run many times, each initialized at different hand positions and orientations close to the observed hand (the largest skin color blob). The best scoring hypothesis of this process was kept as the recovered pose. To assess the method, a set of 45 frames was selected at regular intervals from a real-world sequence and each hand pose recognition was performed 20 times. For the quantitative assessment of the hand pose recognition accuracy, we used as a reference the hand model parameters that were recovered from an experiment that tracked the hand articulation over the whole sequence. Figure 4 shows the histogram of estimation error M_d for all the performed (20×45) experiments. As it can be verified, in 74% of them, the estimated pose deviated $4cm$ or less from the tracked pose. The secondary histogram peak around $8cm$ corresponds to some ambiguous poses for which sometimes the mirrored pose was estimated.

4 Discussion

We proposed a novel model-based method for efficient full DOF hand model initialization and tracking using data acquired by a Kinect sensor. The combination of (a) a careful modeling of the problem (b) a powerful optimization method (c) the exploitation of modern GPUs and, (d) the quality of the data provided by the Kinect sensor, results in a robust and efficient method for tracking the full pose of a hand in complex articulation. Extensive experimental results demonstrate that accurate and robust 3D hand tracking is achievable at 15Hz. Thus,

it is demonstrated that model-based joint-evidence tracking is feasible in near real-time. It is important to note that there is no inherent limitation that prevents the proposed method to be used on any other type of depth images resulting, for example, from standard dense stereo reconstruction methods.

Acknowledgments

This work was partially supported by the IST-FP7-IP-215821 project GRASP. The contribution of Konstantinos Tzevanidis and Pashalis Padeleris, members of CVRL/FORTH, is gratefully acknowledged.

References

[1] Irene Albrecht, Jörg Haber, and Hans-Peter Seidel. Construction and Animation of Anatomically Based Human Hand Models. In *Eurographics symposium on Computer animation*, page 109. Eurographics Association, 2003.

[2] Antonis A. Argyros and Manolis Lourakis. Real-time Tracking of Multiple Skin-colored Objects with a Possibly Moving Camera. pages 368–379. Springer, 2004.

[3] Vassilis Athitsos and Stan Sclaroff. Estimating 3D Hand Pose From a Cluttered Image. *CVPR*, pages II–432–9, 2003.

[4] Maurice Clerc and James Kennedy. The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[5] Martin De La Gorce, Nikos Paragios, and David J. Fleet. Model-based Hand Tracking With Texture, Shading and Self-occlusions. *CVPR*, (June):1–8, June 2008.

[6] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based Hand Pose Estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, 2007.

[7] Henning Hamer, Konrad Schindler, E. Koller-Meier, and Luc Van Gool. Tracking a Hand Manipulating an Object. In *ICCV*, 2009.

[8] Sabine Helwig and Rolf Wanka. Particle Swarm Optimization in High-Dimensional Bounded Search Spaces. In *Swarm Intelligence Symposium*, pages 198–205. IEEE, 2007.

[9] Vijay John, Spela Ivekovic, and Emanuele Trucco. Articulated Human Motion Tracking with HPSO. *International Conference on Computer Vision Theory and Applications*, 2009.

[10] James Kennedy and Russ Eberhart. Particle Swarm Optimization. In *International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, January 1995.

[11] James Kennedy, Russ Eberhart, and Shi Yuhui. *Swarm intelligence*. Morgan Kaufmann, 2001.

[12] Nikolaos Kyriazis, Iason Oikonomidis, and Antonis A. Argyros. A GPU-powered Computational Framework for Efficient 3D Model-based Vision. Technical Report TR420, ICS-FORTH, July 2011.

[13] Microsoft Corp. Redmond WA. Kinect for Xbox 360.

[14] Thomas B Moeslund, Adrian Hilton, and Volker Kru. A Survey of Advances in Vision-based Human Motion Capture and Analysis. *Computer Vision and Image Understanding*, 104:90–126, 2006.

[15] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Markerless and Efficient 26-DOF Hand Pose Recovery. *ACCV*, pages 744–757, 2010.

[16] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Full DOF Tracking of a Hand Interacting with an Object by Modeling Occlusions and Physical Constraints. In *ICCV*, 2011. To appear.

[17] OpenNI. PrimeSense Sensor Module, 2011. URL https://github.com/PrimeSense/Sensor.

[18] James M. Rehg and Takeo Kanade. Model-based tracking of self-occluding articulated objects. In *ICCV*, pages 612–617. IEEE, 1995.

[19] Javier Romero, Hedvig Kjellström, and Danica Kragic. Monocular Real-time 3D Articulated Hand Pose Estimation. *International Conference on Humanoid Robots*, pages 87–92, December 2009.

[20] R. Rosales, Vassilis Athitsos, L. Sigal, and Stan Sclaroff. 3D Hand Pose Reconstruction Using Specialized Mappings. *ICCV*, pages 378–385, 2001.

[21] Mark Schneider and Charles Stevens. Development and Testing of a New Magnetic-tracking Device for Image Guidance. *SPIE Medical Imaging*, pages 65090I–65090I–11, 2007.

[22] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. *CVPR*, 2011.

[23] Björn Stenger, P.R.S. Mendonça, and Roberto Cipolla. Model-based 3D Tracking of an Articulated Hand. *CVPR*, pages II–310–II–315, 2001.

[24] Erik B. Sudderth, Michael I. Mandel, William T. Freeman, and Alan S. Willsky. Visual Hand Tracking Using Nonparametric Belief Propagation. In *CVPR Workshop*, pages 189–189, 2004.

[25] Robert Y. Wang and Jovan Popović. Real-time Hand-tracking With a Color Glove. *ACM Transactions on Graphics*, 28(3):1, July 2009.

[26] Ying Wu and Thomas S. Huang. View-independent Recognition of Hand Postures. In *CVPR*, volume 2, pages 88–94. IEEE Comput. Soc, 2000.

[27] Toshiyuki Yasuda, Kazuhiro Ohkura, and Yoshiyuki Matsumura. Extended PSO with Partial Randomization for Large Scale Multimodal Problems. In *World Automation Congress*, number 1, pages 1–6. IEEE, April 2010.

Tracking the Articulated Motion of Two Strongly Interacting Hands*

I. Oikonomidis, N. Kyriazis, A.A. Argyros
Institute of Computer Scoence, FORTH
{oikonom, kyriazis, argyros}@ics.forth.gr

Abstract

We propose a method that relies on markerless visual observations to track the full articulation of two hands that interact with each-other in a complex, unconstrained manner. We formulate this as an optimization problem whose 54*-dimensional parameter space represents all possible configurations of two hands, each represented as a kinematic structure with* 26 *Degrees of Freedom (DoFs). To solve this problem, we employ Particle Swarm Optimization (PSO), an evolutionary, stochastic optimization method with the objective of finding the two-hands configuration that best explains the RGB-D observations provided by a Kinect sensor. To the best of our knowledge, the proposed method is the first to attempt and achieve the articulated motion tracking of two strongly interacting hands. Extensive quantitative and qualitative experiments with simulated and real world image sequences demonstrate that an accurate and efficient solution of this problem is indeed feasible.*

1. Introduction

The problem of tracking the articulation of the human body from markerless visual observations is of both theoretical interest and practical importance. From a theoretical point of view, the problem is intriguing since humans solve it effortlessly and effectively. From an applications oriented perspective, a solution to this problem facilitates non-intrusive human motion capture and constitutes a fundamental building block towards human activity recognition, human-computer interaction, robot learning by demonstration, etc.

Despite the significant progress in the last years, the problem remains unsolved in its full extent [12]. Difficulties stem from the high dimensionality of the configuration space of the human body, the varying appearance of humans and the self-occlusions of human parts.

In this work we are particularly interested in the problem of tracking hand articulations. We define a hand pose to be a point in a 26-dimensional configuration space that spans the global position and orientation of the hand plus the 20 joint angles between various hand parts. Because of its flexibility that induces generally a concave shape, a performing hand is severely self occluded even when observed from purposefully selected viewpoints. Thus, the markerless tracking of a hand constitutes a high dimensional search problem that needs to be solved based on incomplete and possibly ambiguous observations.

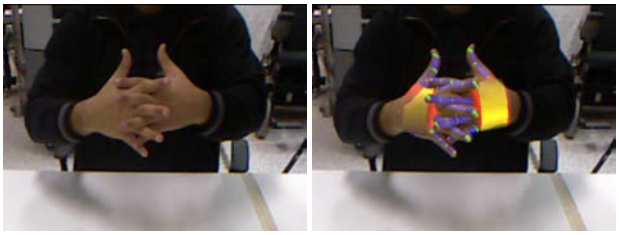


Figure 1. Left: A view of two interacting hands. Right: The configuration of the two hands as estimated by the proposed method, superimposed on the left frame (cropped 320×240 regions from the original 640×480 images).

Tracking two hands in interaction with each other is an even more interesting problem. The interest stems from the fact that a plethora of human activities (object grasping and manipulation, sign language, social interaction) involve collaborative use and strong interaction of both hands. Consider, as an example, the situation shown in Fig. 1. For a human observer, the interpretation of the joint configuration of the two hands is immediate. Even more interestingly, this interpretation is associated to the joint hand configuration rather than to each individual hand. Thus, the availability of computational techniques that are able to jointly infer the full articulation of the hands in such scenarios, opens new avenues in the interpretation of human activities.

Compared to the already difficult problem of tracking the articulation of a single hand, the problem of tracking two hands is even more challenging. If the two hands are clearly separated in the field of view of the observer, it would suffice to solve two instances of the single-hand tracking problem. However, if hands interact with each other, the situation becomes much more complicated. Besides the self-occlusions of each individual hand, further occlusions are introduced because of the complex inter-relations of

the two hands, each hiding important observations of the other. Even further, the available, fewer observations become more ambiguous since the existence of parts from two hands increase the number of potential interpretations. Essentially, each hand acts as a distractor to the interpretation of the other.

The direct implication of the above observations is that it is very difficult for any tracker of a single hand to cope effectively with the problem of tracking two interacting hands. The configuration of each hand can only be inferred in the context of its interaction with the other. This calls for a holistic approach, in which a joint model of the two interacting hands is considered. In such a framework, the desired outcome is the two-hands configuration that not only best explains all available observations, but also the ones that are missing due to the hands interaction.

In this paper we follow this approach. We consider a model of two interacting hands and we formulate an optimization problem whose solution is the position, pose and full articulation of two hands that best explain the set of all available visual observations. We also demonstrate that despite its large dimensionality, this problem can be solved both effectively and efficiently.

1.1. Related work

To the best of our knowledge, there is no existing work that addresses the problem of tracking the full articulation of two interacting hands from markerless visual observations. We therefore provide an overview of works on single-hand articulation tracking and discuss their potential extendability to the problem of tracking two interacting hands.

Single hand pose estimation and tracking methods can be categorized into appearance- and model-based ones [6]. *Appearance-based methods* employ an offline training process for establishing a mapping from a set of image features to a finite set of hand model configurations [3, 19, 23, 18, 20]. The discriminative power of these methods depends on the invariance properties of the employed features, the number and the diversity of the postures to be recognized and the method used to derive the mapping. Appearance-based methods are appropriate for recognizing a small set of known and diverse target hand configurations and less suitable in situations where accurate pose estimation of a freely performing hand is required. The suitability of such methods in scenarios involving two hands seems questionable. This is because the offline training process should consider the combinatorial space of the configurations of the two hands as well as the change in appearance of these configurations because of the different viewpoints of observation.

Model-based approaches [17, 21, 22, 5, 7, 13, 14, 15] generate hand model hypotheses and evaluate them on the available visual observations. Essentially, this is performed by formulating an optimization problem whose objective function measures the discrepancy between observed and synthesized visual cues that are generated based on a certain hand model hypothesis. The employed optimization method should be able to evaluate the objective function at arbitrary points in the multidimensional model parameters space. Thus, unlike appearance-based methods, most of the computations need to be performed online. On the positive side, such methods avoid the time and effort consuming task of training and they provide continuous solutions to the problem of hand pose recovery.

Another categorization is based on how partial evidence regarding the individual rigid parts of the articulated object contributes to the final solution [14]. *Disjoint evidence methods* [17, 22, 7, 20] consider individual parts in isolation prior to evaluating them against observations. *Joint evidence methods* [13, 14, 15, 21, 5, 3, 19, 23, 18] consider all parts in the context of complete articulated object hypotheses. By construction, joint-evidence methods treat part interactions effortlessly, but their computational requirements are rather high. Disjoint evidence methods usually have lower computational requirements than joint-evidence ones, but need to cope explicitly with part interactions such as collisions and occlusions. Since such issues are pronounced in the problem of two hands tracking, disjoint evidence methods are expected to perform worse than joint evidence methods in this problem.

1.2. Contribution

In terms of the previously described classifications, this paper presents a model-based, joint-evidence method for tracking the full articulation of two interacting hands. Observations come from an off-the-shelf Kinect sensor. Two hands tracking is formulated as an optimization problem. The objective function to be minimized quantifies the discrepancy between the 3D structure and appearance of hypothesized configurations of two hands and the corresponding visual observations. Optimization is performed through a variant of a stochastic, evolutionary optimization method (Particle Swarm Optimization - PSO) tailored to the needs of the specific problem.

From a methodological point of view, the proposed approach combines the merits of two recently proposed, state-of-the-art methods for tracking hand articulations [14, 15]. More specifically, in [14], Oikonomidis et al. proposed a joint-evidence method for tracking the full articulation of a single, isolated hand based on the RGB-D data provided by a Kinect sensor. We extend this approach so that it can track two strongly interacting hands.

Our method is also related to the one presented in [15] that tracks a hand interacting with a known rigid object. The fundamental idea behind that work is to model hand-object relations and to treat occlusions as a source of information

rather than as a complicating factor. We extend this idea by demonstrating that it can be exploited effectively in solving the much more complex problem of tracking two articulated objects (two hands). Additionally, this more complex problem is solved based on input provided by a compact Kinect sensor, as opposed to the multicamera calibrated system employed in [15].

Experimental results demonstrate that the accuracy achieved in two hands tracking is in the order of 6mm, in scenarios involving very complex interaction between two hands. Interestingly, despite the large increase in the dimensionality of the problem compared to [14] (from 27 to 54 problem dimensions), the computational budget required for achieving this accuracy is only slightly increased.

The major contributions of this work can be summarized as follows:

- We present the first method for accurate, robust and efficient tracking of the articulated motion of two hands in strong interaction, a problem that has never been addressed before.
- We demonstrate that the core method presented in [14] can be naturally extended to handle the problem of tracking the articulation of two interacting hands.
- We demonstrate that the idea of modeling context and occlusions as presented in [15] can be exploited towards tracking the articulation of two interacting hands.
- We demonstrate that despite the doubling of the dimensionality of the problem compared to [14], the proposed approach achieves comparable accuracy with a comparable computational budget.

2. Tracking two interacting hands

The proposed method achieves tracking of two interacting hands by directly attributing sensory information to the joint articulation of two synthetic and symmetric 3D hand models, of known size and kinematics (see Fig. 2). For given articulations of two hands we are able to coarsely predict what the Kinect would perceive, by simulating the acquisition process, i.e. producing synthetic depth maps for specific camera-scene calibrations. Having established a parametric process that produces comparable data to the Kinect's input, we perform tracking by searching for the parameters that produce depth maps which are most similar to the actual input.

Tracking is performed in an online fashion, where at each step and for every new input an optimization problem is solved. A variant of the PSO search heuristic is used to minimize the discrepancy between the actual Kinect input and simulated depth maps, generated from hypothesized articulations. The best scoring hypothesis constitutes the solution for the current input. The discrepancy measure is carefully formulated so that robustness is achieved. Towards computational efficiency, temporal continuity is exploited at each optimization step.

2.1. Input/preprocessing

The input of the Kinect [11] consists of a RGB image I and a corresponding depth map D, i.e. a depth value for every pixel in I. The dimensions of both arrays are 640×480. A skin color map o_s is produced from I, by means of [2]. From o_s and D a new depth map o_d is computed, where only depth values of D that correspond to skin colored pixels in o_s are kept.

2.2. Model/search space

We define a parametric model of the joint kinematics of two hands. As already discussed, it is of vital importance to consider both hands cojointly, so that we can effectively perform inference over their interaction. The parametric model of the two hands coincides with the search space of each optimization step. Each of the hands has 27 parameters, that represent the hand's pose (3-D position and 4-D quaternion-encoded orientation) and 4-D articulations of each of the 5 fingers (a 2-D revolute joint that connects the palm with the finger and two 1-D revolute joints that connect adjacent phallanges). The ranges of parameter values are linearly bounded, according to anatomical studies [1]. For two hands the dimensionality of the search space amounts to twice the dimensionality for one hand (i.e. 54), as we do not consider any additional constraints over their joint motion.

2.3. Simulation/comparable features

For each point h in the search space (see Sec. 2.2) a mapping to the feature space of the actual observations is required. We simulate the acquisition process of the depth sensor of the Kinect by means of rendering. Each point h defines two 3D skeletons by applying forward kinematics over the parameters detailed in Sec. 2.2. These skeletons are skinned with appropriately transformed instances of 3D spheres and cylinders. The usage of only two primitives proves to be computationally efficient (see Sec. 2.7). Given the calibration information C for the Kinect, we rasterize a depth map $r_d(h, C)$ from the implicit 3D structure described so far. The resulting model is very similar to the one used in [14]. The rendered 3D structure is depicted in Fig. 2(d).

2.4. Discrepancy/objective function

The objective function to be optimized is essentially a penalty function to be minimized. This penalty is defined

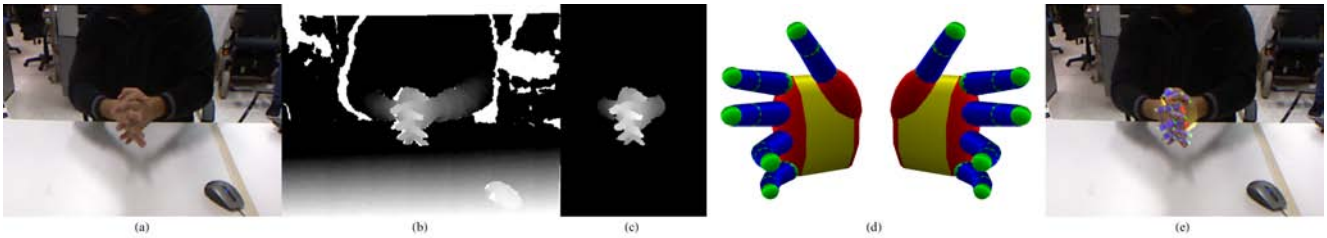


Figure 2. By masking the depth information (b), with a skin color detection performed upon RGB data (a), a depth map (c) of image regions corresponding to hands is extracted, from Kinect input. The proposed method fits the 54-D joint model of two hands (d) onto these observations, thus recovering the hand articulation that best explains the observations (e).

with respect to a tracking frame's observation $O = \{o_s, o_d\}$ and a rendered depth map $r_d(h, C)$ that is generated from a hypothesis h. The penalty function $E(\cdot)$ consists of two terms, a prior term $P(\cdot)$ and a data term $D(\cdot)$:

$$E(O, h, C) = P(h) + \lambda_k \cdot D(O, h, C), \tag{1}$$

where $\lambda_k = 2$ is a normalization factor.

The box bounds of the search space are not expressive enough to tightly define the region of valid hand articulations. Within these bounds, $P(\cdot)$ penalizes invalid articulation hypotheses. In this work we invalidate articulations where adjacent fingers inter-penetrate. Thus,

$$P(h) = \sum_{p \in \mathrm{Q}} -\min(\phi(p, h), 0), \tag{2}$$

where Q amounts to the pairs of adjacent fingers, and ϕ is the abduction-adduction difference (in rads) of adjacent fingers. We have indeed tried elaborate and more computationally expensive collision models to penalize inter-penetration but we have found simple angle differences to efficiently resolve challenging tracking scenarios.

The term $D(\cdot)$ quantifies the incompatibility of input O to an articulation hypothesis h. Essentially, it is the result of the comparison of two parts. The first part consists of the input depth map o_d and the skin map o_s. The other part, and with respect to a hypothesis h, consists of a simulated depth map $r_d(h, C)$ and an implicitly defined skin map $r_s(h, C)$, that is set at points where $r_d(h, C)$ is occupied. The main purpose of $D(\cdot)$ is to penalize depth discrepancies. However, to make it robust, a few more points need to be addressed.

Unless depth differences are clampled within a predetermined range, large differences, that can be due to noise, dominate and produce a false high penalty. By clamping we make $D(\cdot)$ smoother and, thus, add noise tolerance in its optimization. Moreover, we also consult the overlap of the actual and simulated skin maps. More specifically, hypotheses resulting in significant overlap with actual skin maps are preferred even if they result in slightly greater depth discrepancies. Empirical evaluation has proven that this approach eliminates strong local minima around the global minimum and therefore facilitates the convergence of the optimization process to its true optimum.

The aforementioned are encoded in the following penalty function:

$$\begin{aligned} D(O, h, C) \quad = \quad & \lambda \frac{\sum min(|o_d - r_d|, d_M)}{\sum(o_s \vee r_m) + \epsilon} + \\ & \left(1 - \frac{2\sum(o_s \wedge r_m)}{\sum(o_s \wedge r_m) + \sum(o_s \vee r_m)}\right), \end{aligned} \tag{3}$$

where $\lambda = 0.05$ acts as a normalization factor and ϵ is added to denominators in order to avoid possible divisions by zero. Differences are normalized over their effective areas.

2.5. Search/optimization

The challenging task of optimization at each tracking frame is delegated to the powerful Particle Swarm Optimization (PSO) search heuristic [8, 9]. PSO is an evolutionary optimization algorithm that receives an objective function $F(\cdot)$ and a search space S and outputs the optimum of $F(\cdot)$ in S, while treating it as a black box. Being evolutionary, it is parameterized with respect to a population of particles. These parameters amount to the particle count N and the generation count G. Three additional parameters, namely w (constriction factor [4]), c_1 (cognitive component) and c_2 (social component), adjust the behavior of the algorithm.

For each generation k and particle i PSO maintains a state that consists of a global optimum position G_k, a local optimum $P_{k,i}$, the current position $x_{k,i}$ and the current velocity $v_{k,i}$. Initially, particles are sampled uniformly in S. At each generation, the velocity of each particle is updated according to

$$v_{k+1,i} = w(v_{k,i} + c_1 r_1 (P_{k,i} - x_{k,i}) + c_2 r_2 (G_k - x_{k,i})) \tag{4}$$

and the current position of each particle is updated according to:

$$x_{k+1,i} = x_{k,i} + v_{k+1,i}. \tag{5}$$

$P_{k+1,i}$ is set to

$$P_{k+1,i} = \begin{cases} x_{k+1,i}, & F(x_{k+1,i}) < F(P_{k+1,i}) \\ P_{k,i}, & \text{otherwise} \end{cases} \tag{6}$$

G_k is set to the best scoring particle's $P_{k,i}$:

$$G_{k+1} = P_{k+1,l}, \text{ with } l = \arg\min_m \left(F\left(P_{k+1,m}\right)\right). \quad (7)$$

Variables r_1, r_2 represent uniformly distributed random numbers in the range $[0, 1]$.

As suggested in [4], we fix the behavioral parameters to $c_1 = 2.8$, $c_2 = 1.3$ and

$$w = 2/\left|2 - \psi - \sqrt{\psi^2 - 4\psi}\right|. \quad (8)$$

We have experimentally confirmed that for the w as defined in Eq.(8), any combination that satisfies $c_1 + c_2 = 4.1$ achieves essentially the same optimization performance.

There are traits that make PSO attractive to use in a tracking method. It is derivative-agnostic, which makes it easy to try and optimize arbitrary objective functions, with no limitations over convexity, continuity etc. Moreover, its performance depends on essentially two parameters, namely N and G.

In the proposed method a variant of PSO is considered that better suites our tracking requirements. As already stated in [14], the original PSO algorithm has been effective in accurately recovering the pose of the hand's palm (6 DoFs). However, less accuracy has been observed for the fingers (the rest of the 20 DoFs). This occurs due to premature "collapsing" [9] of the population. In order to alleviate this, we employ additional randomization over the remaining parameters, so that their range is better explored [24]. This process is applied to the joint parameter space of both hands (54 DoFs) and for the 40 parameters that regard the 10 fingers.

Additionally, we exploit the parallel nature of PSO by delegating evaluations of individual particles to distinct computational cores of a parallel platform. Each generation is evaluated in parallel, given that the score of each particle is independent to any other. This introduces significant benefits with respect to execution times.

2.6. Tracking loop

In order to perform tracking across time we iterate over instances of the same optimization problem. Each iteration is performed on new input provided from the Kinect and yields a new pose estimate. In order to provide such an estimate, $E(\cdot)$ is minimized by PSO. What differs from frame to frame is the input and the effective search area that is provided to PSO.

For every new frame, the newly acquired input is preprocessed and mapped into the feature space of skin and depth measurements, as variable O. All subsequent evaluations of $E(\cdot)$ are performed based on this input. Every hypothesis h that is generated by PSO is rendered and thus mapped to the same feature space. PSO drives an exploratory course

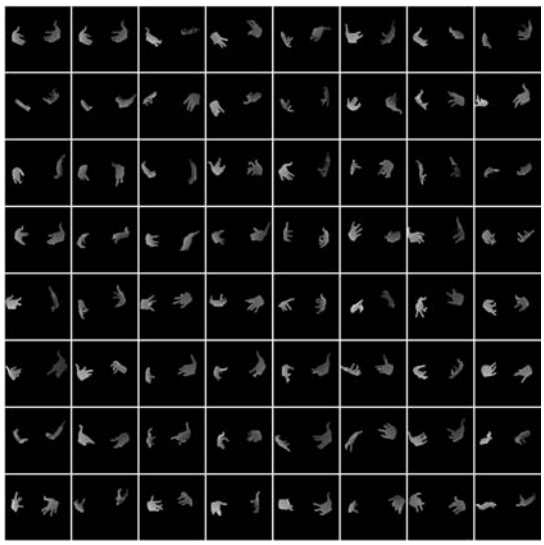


Figure 3. Feature mapping of an entire generation of model hypotheses that can be generated and evaluated in sub-millisecond time scale on a GPU.

in which multiple invocations ($N \times G$) of $E(O, h, C)$ are made. The optimal hypothesis

$$h_{max} = \arg\min_h E(O, h, C) \quad (9)$$

is output as the inferred articulation for the current tracking frame.

Although the original PSO requires a uniform initialization of its population in S we decide to exploit temporal continuity and constrain the effective search area, for the sake of convergence and computational efficiency. To do so, for every next tracking frame we initialize the population to be in the vicinity of h_{max} of the previous frame. The optimum h_{max} of the previous frame is copied to the new population. The rest of the population consists of random perturbations of h_{max}.

2.7. Parallel implementation

The execution time of the presented tracking loop is dominated by the evaluation of the data term $D(\cdot)$ of the penalty function $E(\cdot)$ (see Eq. (1)). 3D rendering and operations over entire maps induce costs that are prohibitive for mainstream CPUs but can be efficiently handled by contemporary GPUs. We exploit parallelism by considering renderings of multiple hypotheses, simultaneously, in big tiled renderings. Essentially, an entire generation is feature-mapped upon a single 2D array, as shown in Fig.3. Per pixel computations are implemented using shaders and the required summations are performed by means of *mip* (multum in parvo) mapping with the addition operator. Following the guidelines of [10], we employ hardware instancing and multi-viewport clipping in order to efficiently cope with

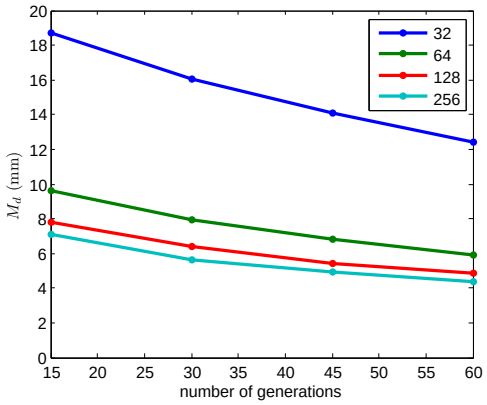


Figure 4. Quantitative evaluation of the performance of the method with respect to the PSO parameters.

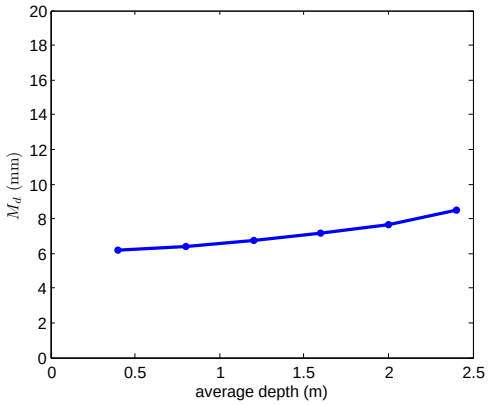


Figure 5. Quantitative evaluation of the performance of the method with respect to the average distance from the sensor.

many model hypotheses that consist of homogeneous transformations of just a sphere and a cylinder.

3. Experimental evaluation

Synthetic data as well as real-world sequences obtained by a Kinect sensor were used to experimentally evaluate the proposed method. Experiments were performed on a computer equipped with a quad-core Intel i7 950 CPU, 6 GBs RAM and an Nvidia GTX 580 GPU with $1581\ GFlops$ processing power and 1.5 GBs of memory.

3.1. Experiments on synthetic data

The quantitative evaluation of the proposed method has been performed using synthetic data. This approach is often encountered in the relevant literature [7, 13, 15, 14] because ground truth data for real-world image sequences is hard to obtain. The employed synthetic sequence consists of 300 consecutive hand poses that encode typical interactions of two hands. Rendering was used to synthesize the required input O. To quantify the accuracy in hand pose estimation, we adopt the metric used in [7]. More specifically, the distance between corresponding phalanx endpoints in the ground truth and in the estimated hand poses is measured. The average of all these distances, for both hands, over all the frames of the sequence constitutes the resulting error estimate Δ. It is worth noting that these distances include estimations for hand points that, because of occlusions, are not observable.

The influence of several factors to the performance of the method was assessed in respective experiments. Figure 4 illustrates the behavior of the method with respect to the PSO parameters (number of generations and particles per generation). The product of these parameters determines the computational budget of the proposed methodology, i.e. the number of objective function evaluations. The horizontal axis of the plot denotes the number of PSO generations. Each plot of the graph corresponds to a different number of particles per generation. Each point in each plot is the median M_d of the error Δ for 20 repetitions of an experiment run with the specific parameters. A first observation is that M_d decreases monotonically as the number of generations increase. Additionally, as the particles per generation increase, the resulting error decreases. Nevertheless, employing more that 45 generations and more than 64 particles results in disproportionally small improvement of the method's accuracy. The gains are at most $2mm$ or roughly 30%, for a 5-fold increase in computational budget. For this reason, the configuration of 64 particles for 45 generations was retained in all further experiments. In terms of computational performance, tracking is achieved at a framerate of 4Hz on the computational infrastructure described in Sec.3.

In another experiment we assessed the effect of varying the distance of the hands from the hypothesized sensor, exploring the usefulness of the method in different application scenarios that require observations of a certain scene at different scales (e.g., close-up views of hands versus distant views of a human and his/her broader environment). To do this, we generated the same synthetic sequences at different average depths. The results of this experiment are presented in Fig. 5. At a distance of 50cm the error is equal to $6mm$. As the distance increases, the error also increases; Interestingly though, it doesn't exceed $8.5mm$ even at an average distance of $2.5m$.

The tolerance of the method to noisy observations was also evaluated. Two types of noise were considered. Errors in depth estimation were modeled as a Gaussian distribution centered around the actual depth value with the variance controlling the amount of noise. Skin-color segmentation errors were treated similarly to [18], by randomly flipping the label (skin/non-skin) of a percentage of pixels

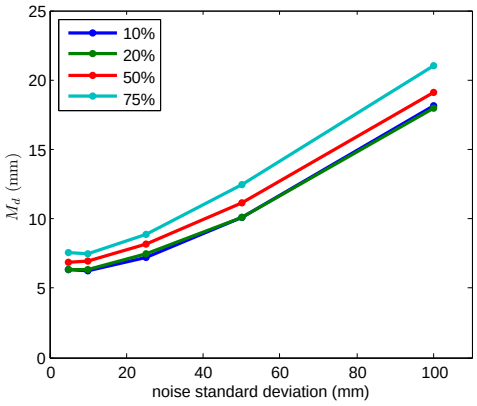


Figure 6. Quantitative evaluation of the performance of the method with respect to depth and skin-lcolor detection noise.

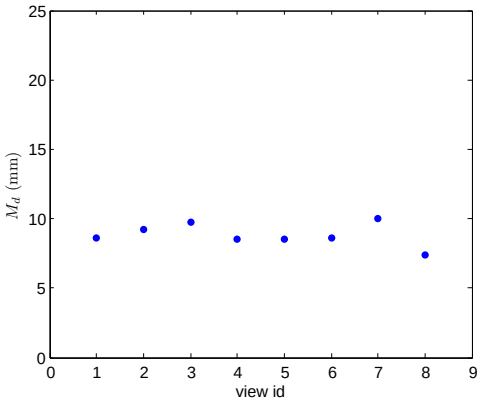


Figure 7. Quantitative evaluation of the performance of the method with respect to viewpoint variation.

in the synthetic skin mask. Figure 6 plots the method's error in hand pose estimation for different levels of depth and skin segmentation error. As it can be verified, the hand pose recovery error is bounded in the range $[6mm..23mm]$, even in data sets very heavily contaminated with noise.

The accuracy in hand pose estimation with respect to viewpoint variations was also assessed. This was achieved by placing the virtual camera at 8 positions dispersed on the surface of a hemisphere placed around the hypothesized scene. The data points of Fig. 7 demonstrate that viewpoint variations do not significantly affect the performance of the method.

In a final experiment, we measured the performance of a single hand tracker [14] on the synthetic data set of the previous experiments. To do so, we employed our own implementation of that method. The resulting error M_d for this experiment was 145mm. In practice, the single hand tracker is able to track accurately one of the two hands while it is not in interaction with the other. However, as soon as occlusions become extended due to hands interaction (for example, when one hand passes in front of the other), the track is often completely lost.

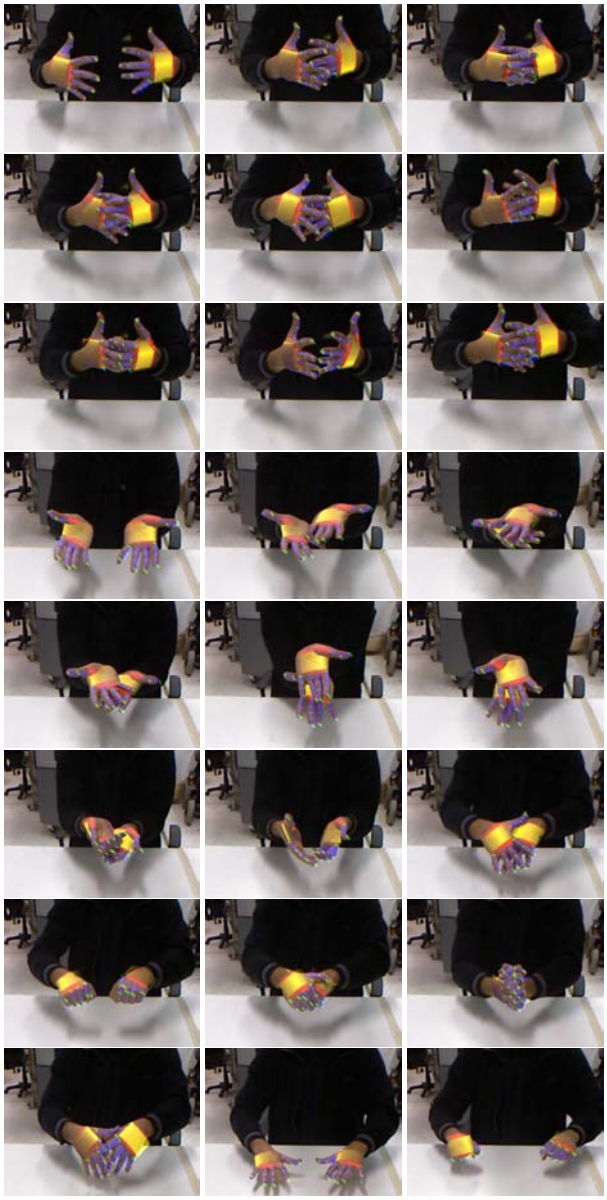


Figure 8. Snapshots from an experiment where two hands interact with each other (cropped 320×240 regions from the original 640×480 images).

3.2. Experiments on real world sequences

Towards the qualitative evaluation of the proposed approach in real data, several long real-world image sequences were captured using the PrimeSense Sensor Module of OpenNI [16]. The supplemental material accompanying the paper provides a video with the results obtained from

one such sequence (1776 frames). Indicative snapshots are shown in Fig. 8. As it can be observed, the estimated hand models are in very close agreement with the image data, despite the complex articulation and strong interactions of the two hands.

4. Discussion

We proposed a method for tracking the full articulation of two strongly interacting hands, based on observations acquired by the Kinect sensor. The problem was formulated as an optimization problem in a 54-dimensional parameter space spanning all possible configurations of two hands. Optimization seeks for the joint hand configuration that minimizes the discrepancy between rendered hand hypotheses and actual visual observations. Particle Swarm Optimization proved to be competent in solving this high dimensional optimization problem. More specifically, extensive experimental results demonstrated that accurate and robust tracking of two interacting hands can be achieved with an accuracy of 6mm at a framerate of 4Hz. Experimental results also demonstrated that in the presence of strong hand interactions, the straightforward alternative of solving two instances of a single hand tracking problem results in a much lower accuracy. The proposed approach is the first to achieve a solution to this interesting and challenging problem. Hopefully, it will constitute an important building block in a large spectrum of application domains that critically depend on the accurate markerless perception of bi-manual human activities.

References

[1] I. Albrecht, J. Haber, and H.-P. Seidel. Construction and Animation of Anatomically Based Human Hand Models. In *Eurographics symposium on Computer animation*, page 109. Eurographics Association, 2003. 3

[2] A. A. Argyros and M. Lourakis. Real-time Tracking of Multiple Skin-colored Objects with a Possibly Moving Camera. In *ECCV*, pages 368–379. Springer, 2004. 3

[3] V. Athitsos and S. Sclaroff. Estimating 3D Hand Pose From a Cluttered Image. In *CVPR*, pages II–432–9. IEEE, 2003. 2

[4] M. Clerc and J. Kennedy. The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *Transactions on Evolutionary Computation*, 6(1):58–73, 2002. 4, 5

[5] M. De La Gorce, N. Paragios, and D. J. Fleet. Model-based Hand Tracking With Texture, Shading and Self-occlusions. In *CVPR*, pages 1–8. IEEE, Jun. 2008. 2

[6] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based Hand Pose Estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, 2007. 2

[7] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a Hand Manipulating an Object. In *ICCV*, 2009. 2, 6

[8] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, Jan. 1995. 4

[9] J. Kennedy, R. Eberhart, and S. Yuhui. *Swarm intelligence*. Morgan Kaufmann, 2001. 4, 5

[10] N. Kyriazis, I. Oikonomidis, and A. A. Argyros. A GPU-powered Computational Framework for Efficient 3D Model-based Vision. Technical Report TR420, ICS-FORTH, Jul. 2011. 5

[11] Microsoft Corp. Redmond WA. Kinect for Xbox 360. 3

[12] T. B. Moeslund, A. Hilton, and V. Kru. A Survey of Advances in Vision-based Human Motion Capture and Analysis. *Computer Vision and Image Understanding*, 104:90–126, 2006. 1

[13] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Markerless and Efficient 26-DOF Hand Pose Recovery. In *ACCV*, pages 744–757. Springer, 2010. 2, 6

[14] 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 *BMVC*, Aug. 2011. 2, 3, 5, 6, 7

[15] 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. IEEE, Nov. 2011. 2, 3, 6

[16] OpenNI. PrimeSense Sensor Module, 2011. 7

[17] J. M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *ICCV*, pages 612–617. IEEE, 1995. 2

[18] J. Romero, H. Kjellström, and D. Kragic. Monocular Real-time 3D Articulated Hand Pose Estimation. In *International Conference on Humanoid Robots*, pages 87–92. IEEE, Dec. 2009. 2, 6

[19] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D Hand Pose Reconstruction Using Specialized Mappings. *ICCV*, pages 378–385, 2001. 2

[20] 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*. IEEE, 2011. 2

[21] B. Stenger, P. Mendonça, and R. Cipolla. Model-based 3D Tracking of an Articulated Hand. In *CVPR*, pages II–310–II–315. IEEE, 2001. 2

[22] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Visual Hand Tracking Using Nonparametric Belief Propagation. In *CVPR Workshop*, pages 189–189, 2004. 2

[23] Y. Wu and T. S. Huang. View-independent Recognition of Hand Postures. In *CVPR*, volume 2, pages 88–94. IEEE, 2000. 2

[24] T. Yasuda, K. Ohkura, and Y. Matsumura. Extended PSO with Partial Randomization for Large Scale Multimodal Problems. In *World Automation Congress*, pages 1–6. IEEE, Apr. 2010. 5

Binding Vision to Physics Based Simulation: The Case Study of a Bouncing Ball

Nikolaos Kyriazis[1,2]
http://www.ics.forth.gr/~kyriazis
Iason Oikonomidis[1,2]
http://www.ics.forth.gr/~oikonom
Antonis A. Argyros[1,2]
http://www.ics.forth.gr/~argyros

[1] Computational Vision and Robotics Lab., Institute of Computer Science, FORTH
[2] Computer Science Department, University of Crete, Greece

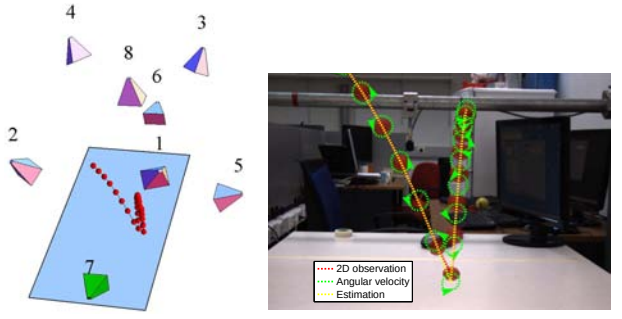
Abstract

A dynamic scene and, therefore, its visual observations are invariably determined by the laws of physics. We demonstrate an illustrative case where *physical explanation*, as a vision prior, is not a commodity but a necessity. By considering the problem of ball motion estimation we show how physics-based simulation in conjunction with visual processes can lead to the reduction of the visual input required to infer physical attributes of the observed world. Even further, we show that the proposed methodology manages to reveal certain physical attributes of the observed scene that are difficult or even impossible to extract by other means. A series of experiments on synthetic data as well as experiments with image sequences of an actual ball, support the validity of the proposed approach. The use of generic tools and the top-down nature of the proposed approach make it general enough to be a likely candidate for handling even more complex problems in larger contexts.

1 Introduction

Computer vision is concerned with the understanding of the physical world through the analysis of its image(s). Such an understanding may be defined at various levels of abstraction. Whatever the level of abstraction may be, this understanding is always associated with a context, i.e. an assumption of a generative process that produces the observations. It is convenient to think about such a context as a set of rules that transform some initial conditions into images. In this work, we are interested in deriving a *physically plausible explanation* of a dynamic scene. Thus, the respective rules governing the generative process are the *laws of physics*.

We argue that by exploiting this type of context as a prior, we can derive very useful information for a dynamic scene, that is difficult or even impossible to derive by other means. Consider for example the testbed scenario according to which we are interested in estimating the state of a uniformly colored bouncing ball through its observation by a single or by multiple calibrated cameras (Fig. 1). By employing standard computer vision techniques, accounting for the position of the ball at each time step is not trivial. The possibly inadequate acquisition frame rate may lead to aliasing and the possibly large shutter time may lead to



(a) (b)

Figure 1: A ball is thrown towards a table with a high back-spin. By incorporating physics-based simulation, we infer the ball's 3D trajectory (a), and its linear and angular velocities *from a single camera* (cam. 7). The proposed method identifies that a back-spin is the cause of the reduction of the outgoing angle of the bounce. The green ellipses in (b) are projections of an equator of the ball and the arrows represent the direction of the estimated angular speed.

motion blur. On top of the above mentioned difficulties, for some aspects of the state of the ball (i.e., its orientation and/or angular velocity) there is no direct evidence, whatsoever. The problem becomes even more challenging when we are interested in solving the above problems based on single-camera observations and/or when, due to occlusions, the available set of visual observations becomes even more limited.

We show that through the direct incorporation of explicit physics we are able to tackle these challenges. We demonstrate how hidden variables like the position of the ball when it is occluded, its orientation and angular velocities, can be estimated. We highlight that physics provide a strong prior, which permits the successful treatment of these challenges, even for the case of single camera 2D observations that may be incomplete due to occlusions. The incorporation of physics is performed in a clean, top-down fashion that could be generalized and scaled towards solving larger problems in different contexts.

The proposed framework becomes possible because of the evolution of optimization methods, the advancement of physics-based simulation and the availability of substantial computational power. Powerful optimization techniques enable efficient optimization of hard, multi-dimensional problems [16]. Physics simulation has advanced to a point where computational demands can be efficiently handled, realism is a common denominator in most physics simulators and the extension of simulators is easy due to their carefully designed software architectures. Moreover, parallel multicore technologies like contemporary CPUs and GPUs allow for the computation of thousands of simulations per second. Although not exploited in this work, the latter further extends our method's potential.

2 Relevant work

In the past, several researchers have stressed the benefits stemming from the consideration of physics as integral part of computer vision processes. Although beyond the scope of this work, we mention approaches [12, 17, 19, 25] that exploit the physical nature of light to process images and estimate or predict otherwise unaccountable information.

The prevalent case study of employing physics in vision is the problem of 3D human

tracking. The dynamics of the human body can be exploited towards the formation of strong priors. Popović and Witkin [21] rectified 3D motion capture data to make them compliant to physical constraints. Vondrak *et al.* [24] fused motion planning, contact dynamics and a ground assumption to track humans from multiple cameras. Brubaker *et al.* [6, 7, 9] employed realistic metaphors of the lower body dynamics to estimate and predict walking. Going further, they incorporated a friction model for a ground that affords human motion upon it [8].

There are also approaches that reflect physics implicitly or metaphorically. Brand *et al.* [4, 5] exploited the physical notion of causality in order to perform qualitative reasoning in computer vision problems. Delamarre [14] assigned a physical behaviour to a contour model that drove the optimization process of recovering it. Chen *et al.* [11] were able to track a basketball, while in the air and despite occlusions, by assuming the parabolic nature of free flight. Papadourakis and Argyros [20] identified the physical notion of object permanence as the ambiguity resolver for the case of multiple objects tracking. Sethi and Roy-Chowdhury [22] gave physical substance to image features and used methods, usually employed in physics, in order to model activities in image sequences.

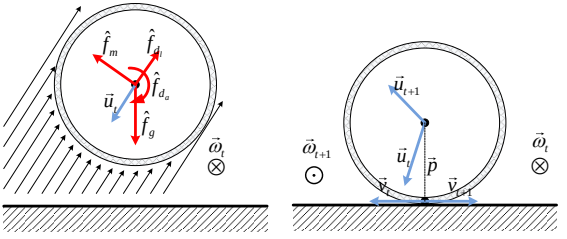
This work is most closely related to the works in [3, 15, 18]. Metaxas and Terzopoulos [18] defined a continuous Kalman filter that was able to track a deformable object. This was achieved by a detailed motion model that was tightly coupled to the optimization process. Although interesting, the extensibility of their approach is hindered by this tight coupling. Bhat *et al.* [3] performed 3D tracking of an object by searching over parameterized experiments that optimally project back to an image sequence. However, the shape of the object and the restriction that it is tracked while in flight does not expose the full potential of employing physics. Finally, Duff and Wyatt [15] used physical simulation and search heuristics to track a fast moving ball, despite occlusions. They reasoned upon the ball's 2D position but they did not consider the 3D case, or the hidden variables of ball orientation and angular velocity.

Despite the significant amount of existing work, no existing study demonstrates the full potential of binding vision to physics-based simulation. We try to fill this gap by proposing a method that is generic, top-down, simulation based and incorporates realistic simulation of physics. As a result, and to the best of our knowledge, the proposed method is the first to consider physical properties that can be estimated through physics-based simulation, even in the case of single camera observations and severe occlusions.

3 Methodology

Let a colored ball be thrown on a table so that it bounces for several times and then rests. The 2D image position of the ball can be easily recovered for every time step and for every camera that views it for the case of moderate velocities. Accurate recovery is problematic for the case of larger velocities and especially around bounces, due to blurring and aliasing (see Fig. 1). These problems hinder a bottom-up resolution of the problem, but, as it will be shown, they do not prevent a top-down approach from being effective.

We consider the *physical explanation* e of the bouncing of the observed ball. We assume that certain scene properties (mass, inertia, collision properties) and initial conditions (position and velocities of the throw), together with the laws of physics, generate a 3D trajectory which optimally projects back to all cameras and matches the observations o. We define an objective function S that quantifies the discrepancy between the actual observations and the



(a) (b)

Figure 2: The two phases of the bouncing ball (a) flight and (b) bounce. Red arrows represent impulses and blue arrows represent velocities. Angular velocities are perpendicular to the image plane. Black arrows represent the air flow with respect to the flying ball.

camera back-projection of a simulated parameterized ball throwing experiment. The latter can be sub-sampled to match the acquisition rate of the actual camera set. This also accounts for the aliasing effects of the acquisition process.

Since whatever is observed must be *physically plausible*, the *physical explanation* e is the minimizer parameter vector x of this objective function. In notation:

$$e = \arg\min_x S(o,x) \quad \text{where} \quad S(o,x) = \text{BackProjectionError}\left(o, \text{Simulation}\left(x\right)\right) . \tag{1}$$

Our method receives 2D or 3D trajectories that represent the course of a bouncing ball and outputs the parameters of a simulated experiment that optimally matches the observations. In the next sections we describe in detail the parameter space of the simulations (search space for e), the simulation procedure as well as the minimization process.

3.1 Physics of the bouncing ball

In order to account for the dynamics of the trajectory of a bouncing ball we explicitly reason upon an idealized, yet sufficient, physics model. We identify two alternating phases, namely ball flying and ball bouncing. The two phases are detailed in Fig. 2. Since we consider average effects over generally small time intervals, we discuss impulses rather than forces. For a time interval dt and a function of force $\vec{f}$ over that interval, the respective impulse is defined as $\hat{f} = \int \vec{f} dt$.

During its flight (Fig. 2(a)), the ball undergoes velocity changes that are inflicted by the gravitational attraction and the resistance of the air. Gravity constantly exerts a downwards impulse of $\hat{f}_g = m \cdot \vec{g}$, where m is the mass of the ball and $\vec{g}$ is the gravitational acceleration. Given enough air resistance, at each time step t, the linear velocity $\vec{u}_t$ and angular velocity $\vec{\omega}_t$ are decreased in magnitude due to friction (linear damping d_l and angular damping d_a). Also, an impulse $\hat{f}_m = K \cdot (\vec{u}_t \times \vec{\omega}_t)$ that is perpendicular to the linear velocity and the axis of angular velocity, makes the ball travel in a curved trajectory [1]. For every part of the flight the standard equations of motion hold and suffice in order to predict the state of the ball.

At every bounce (Fig. 2(b)), a portion of the ball's vertical energy is lost according to an elasticity factor $\beta \in [0,1]$. An amount of dynamic friction redistributes energy between its linear and angular motion in the horizontal, according to a friction factor $\alpha \in [-1,1]$. The friction model adopted here is an extension of [7] to the 3D case and identifies friction as the reason that scales the total linear velocity $\vec{v}_{t+1}$ of the contact point. This modeling

accounts for a great variety of frictions. For example, both the glass ball (no friction) and the super ball (extreme friction) can be modeled for $\alpha = 1$ and $\alpha = -1$, respectively. The vertical axis of rotation has no contribution to the horizontal contact. Moreover, the impulse which changes the horizontal linear momentum is also the negative impulse that changes the horizontal angular momentum. All the aforementioned constraints define a system of equations:

$$\begin{array}{lcl} S_y\vec{u}_{t+1} & = & -\beta S_y\vec{u}_t \\ S_y\vec{\omega}_{t+1} & = & S_y\vec{\omega}_t \\ S_{xz}\vec{v}_{t+1} & = & \alpha S_{xz}\vec{v}_t \\ m\cdot\vec{p}\times S_{xz}\left(\vec{v}_{t+1}-\vec{v}_t\right) & = & -I\cdot S_{xz}\left(\vec{\omega}_{t+1}-\vec{\omega}_t\right) \end{array} \quad \text{with} \quad \begin{array}{c} S_y = Diag\left([0,1,0]\right) \\ S_{xz} = Diag\left([1,0,1]\right) \\ \vec{v}_k = \vec{u}_k + \vec{p}\times\vec{\omega}_k \end{array} \tag{2}$$

These equations linearly relate the pre-bounce velocities $\vec{u}_t, \vec{\omega}_t$ to post-bounce velocities $\vec{u}_{t+1}, \vec{\omega}_{t+1}$. Solving this system for time $t+1$ yields the post-bouncing state of the ball.

3.2 Physics-based simulation

Ubiquitous physics simulators are already able to account for the most part of the presented physics modeling. They also ease the incorporation of more detailed models via modular architectures that are carefully designed for that purpose. In our scenario we augment such a simulator by incorporating the effects of $\hat{f}_g$ and $\hat{f}_m$ and by adjusting the collision module so that it also accounts for the exchange of horizontal energy. The vertical is already in agreement with our equations. The parameter vector of a simulation is $(m, I, \beta, \alpha, d_l, d_a, K, \vec{s}_0, \vec{u}_0, \vec{\omega}_0, T)$. The state of the ball $\left(\vec{l}_t, \vec{q}_t, \vec{u}_t, \vec{\omega}_t\right)$ is the result of the invocation of the simulator for a given parameter vector, where t is a time step of the whole duration T and $\vec{l}_t$, $\vec{q}_t$, $\vec{u}_t$, $\vec{\omega}_t$ represent position, orientation, linear and angular velocity (all in 3D space).

3.3 Optimization through Differential Evolution (DE)

Differential Evolution (DE) [13, 23] is an evolutionary optimization method. It depends on only a few parameters that have an intuitive explanation and exhibits remarkable performance in difficult problems of large dimensionality. DE effectively handles real-valued multidimensional, potentially non-linear and non-differentiable objective functions. It performs optimization by evolving a set of N_H hypotheses H_g and dimensionality D. Being evolutionary, DE is defined via its mutation, crossover and selection mechanisms that are applied at every generation g.

During mutation, every hypothesis $h_{g,i} \in H_g$ becomes a linear combination of three randomly selected, pairwise different hypotheses of the previous generation $g-1$.

$$h_{g,i} = h_{g-1,r_1} + F\left(h_{g-1,r_2} - h_{g-1,r_3}\right), r_j \sim U\left(0, \left|H_g\right|\right) \wedge r_k \neq r_l \forall k,l. \tag{3}$$

Mutation is controlled by the differentiation factor $F \in [0,2]$. Each mutated hypothesis $h_{g,i}$ is then combined with $h_{g-1,i}$ in order to produce a replacement candidate $\hat{h}_{g,i}$ in the crossover phase.

$$\hat{h}_{g,i}(j) = \begin{cases} h_{g,i}(j), & r_j \leq CR \vee j = idx_{g,i} \\ h_{g-1,i}(j), & \text{otherwise} \end{cases}, j = 1,\ldots,D, r_j \sim U(0,1), idx_{g,i} \sim U(1,D) \tag{4}$$

In Eq. (4), $h_{g,i}(j)$ denotes the j-th component of the i-th hypothesis in the g-th generation. The crossover constant CR controls the combination of individual parameters of $h_{g,i}$ and $h_{g-1,i}$. A random parameter index $idx_{g,i}$ is preselected in order to ensure that at least one parameter of the mutated vector will survive the crossover.

Finally, in the selection phase, the replacement candidate actually replaces the original one in the next generation, if it scores better in the objective function.

The original algorithm is parallel, in the sense that two consecutive generations are two distinct sets. We consider a serial variant, where the two generations are mixed. This means that a mutation may be based on already mutated vectors in the same generation. We have experimentally observed this mixing to add quicker reflexes to the algorithm, leading to faster convergence. We also consider a dithering parameter δ that modulates F at each generation. Dithering improves convergence and helps in avoiding local optima [10]. The DE variant of our choice appears with the coding DE/rand/1/bin in [23]. The input of DE is a real-valued objective function f (in our case, BackProjectionError in Eq. (1)), the number of generations N_G, hypotheses per generation N_H and constants F, CR, δ. The output of DE is the real-valued parameter vector that optimizes f. In all experiments we used the following parameterization for DE: $(N_G, N_H, F, CR, \delta) = (300, 72, 0.9, 0.9, 1.5)$.

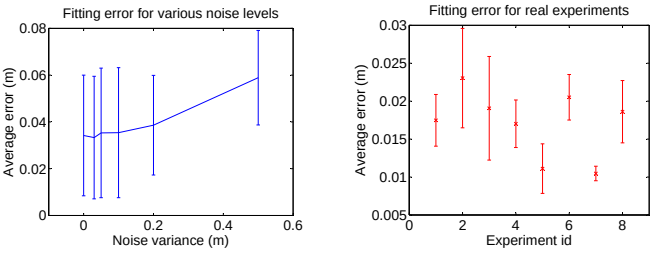
4 Experimental results

A series of experiments were conducted to assess our method's ability to account for 3D and 2D observations of a bouncing ball. From an implementation point of view, we used the DE implementation of the SwarmOps[1] library, the Newton Game Dynamics[2] simulator and the MATLAB[3] platform for the rest of the logic.

4.1 Results on synthesized image sequences

A first series of experiments were carried out to assess the capability of the proposed method to come up with physically plausible explanations of various simulated ball throws, performed in different world contexts and initial conditions. We distinguished the parameters representing scene properties $(m, I, \beta, \alpha, d_l, d_a, K)$ and those representing initial conditions $(\vec{s}_0, \vec{u}_0, \vec{\omega}_0)$. We generated 3 random scene property parameter vectors and 3 random initial condition vectors. We then considered all possible combinations resulting in a total of 9 experiments. The experiment parameterizations generated 9 ball 3D trajectories for a time duration of $T = 4s$, each. Each 3D trajectory was considered in conjunction with 6 levels of Gaussian noise at each of the 3 spatial dimensions, separately and with variances $0m$, $0.03m$,$0.05m$,$0.1m$, $0.2m$ and $0.5m$, respectively. For each set of parameters, 20 repetitions were executed. This protocol led to a total of $3 \times 3 \times 6 \times 20 = 1080$ experiments accounting for various world properties, initial conditions and amounts of noise. For each experiment, the physics-based simulator produced a ground truth 3D trajectory of the ball and the proposed method was employed to provide a physical explanation of it. We evaluated the optimization accuracy by measuring, for each experiment, the average of the Euclidean distances between corresponding points of the simulated and the recovered 3D ball trajectories. The results presented in Fig. 3(a) show that the proposed method is able to perform

[1] http://www.hvass-labs.org/projects/swarmops/c/
[2] http://www.newtongamedynamics.com
[3] http://www.mathworks.com/products/matlab/



(a) (b)

Figure 3: The mean values and standard deviations for the errors on the experiments with (a) synthetic and (b) real observations.

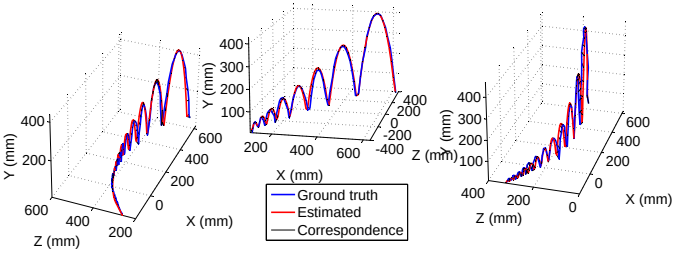


Figure 4: Examples of actual and estimated 3D trajectories. The illustrated trajectories have high left, low left and high right curvature, respectively. The respective average trajectory point estimation errors were 1.38cm, 0.75cm and 0.83cm.

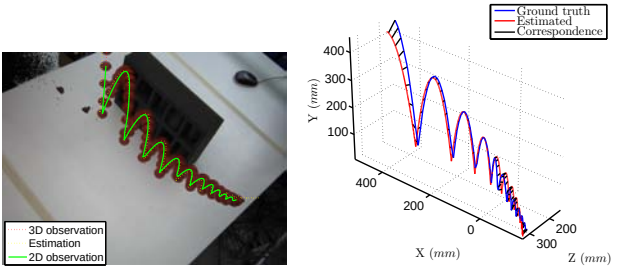
well even under severe Gaussian noise. This is because by conception, the method allows for physically plausible solutions, only. Thus, observations that are heavily contaminated by this type of noise cannot distract the estimation towards physically implausible solutions.

4.2 Multiview estimation of 3D trajectories

Experiments analogous to those of Sec. 4.1 were also performed in the real world, i.e., using multicamera observations of an actual bouncing ball. For these experiments, we employed the setup that is illustrated in Fig. 1(a). It consists of a $2 \times 1m^2$ hard table, a red table tennis ball of radius $2cm$, and 8 synchronized and calibrated Flea2 PointGrey cameras. All cameras provided images at a resolution of 1280×960 and at an acquisition rate of $30fps$. Processing was performed on a workstation that has an Intel Core i7 950 CPU @ $3.07Ghz$ and $6GB$ of RAM. All computations were performed on a single thread of a single core of the CPU.

As a first step, the ball was detected in every frame for all sequences (all cameras inclusive). We applied color thresholding to isolate red areas in every frame. We then filtered each extracted connected blob based on its shape, to ensure high confidence detection and excluded partially occluded and/or significantly blurred detections. 3D ball positions were then estimated through multiview 3D reconstruction of the ball centroids.

We conducted several ball throwing experiments in our physical setup. We selected 8 of them according to our empirical criterion of diversity. Each one was input to our method 20 times. The optimization process of Sec. 4.1 was employed, where synthetic data were



(a) (b)

Figure 5: Estimation of the 3D trajectory of a ball from single camera 2D observations (camera 3) and the assumption of a given physical world.

replaced by real data. The obtained results are presented in Fig. 3(b). Some examples of actual trajectories and the respective estimates are shown in Fig. 4. As it can be verified, the proposed method faithfully reproduces the actual 3D observations.

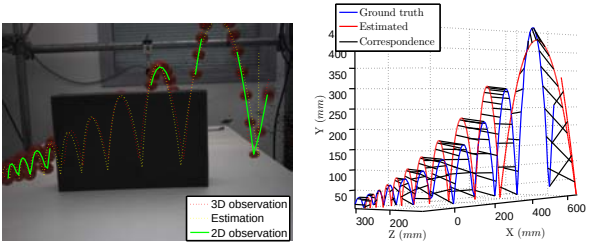
4.3 Single view estimation of 3D trajectories

We are interested in estimating the ball 3D trajectory by a single camera. Without any physics-based prior information and for the case of a ball of known size, single view ball 3D localization depends on the ability to accurately estimate the ball's projected shape and size. In practice, this is problematic due to acquisition and processing artifacts, which lead to errors in depth estimation that are difficult to treat in a bottom-up fashion. However, by modeling the physics of the process, we are able to infer depth from a more reliable source, the 2D trajectory of the ball on the image plane of a single camera. To demonstrate this, we optimized S for 2D observations of a single camera and the non trivial cases of non-planar (due to spin) trajectories. During optimization, the simulator generated 3D data from which 2D reference trajectories were produced by means of projection. The back-projection error, i.e., the average Euclidean distance between back-projected and observed 2D positions of the ball was guiding the optimization process. An exemplar 3D estimation is illustrated in Fig. 5. It can be verified that the estimation from a single camera is almost indistinguishable from the ground truth.

Interestingly, no post-processing is required to enforce the plausibility of the solution because implausible hypotheses are not considered at all. Even more importantly, even though 3D estimation from 2D trajectories relies on the knowledge of the respective ball heights, we do not account for this knowledge explicitly. 3D reconstruction comes effortlessly, as a byproduct of physics-based simulation. Another interesting observation is that the points at which bounces are observable suffer from aliasing. However, since we also sub-sample the simulator's behavior at real acquisition rate, we also account for this type of aliasing.

4.4 Seeing through walls with a single camera

We also tested our method's effectiveness under considerable lack of constraints, i.e., in the case of partial observations due to occlusions. We recorded ball throws that were largely invisible to camera 6 due to a purposefully placed large obstacle (see Fig. 6(a)). Based on this partial evidence, the proposed method estimated the 3D trajectory of the ball from the



(a) (b)

Figure 6: Single view estimation of the 3D trajectory of a ball from partial 2D observations (camera 6) and the assumption of a given physical world.

(single) view of camera 6. The ball was still visible to some of the rest of the cameras. This information was only used to estimate a kind of ground truth for the 3D trajectory of the ball. Figure 6 shows the actual observations, the ball trajectory as this was estimated by camera 6 and the ground truth as this was measured by the rest of the cameras. The estimation of the 3D trajectory (see Fig. 6(b)) is not that accurate due to the lack of enough constraints. Still, it is quite satisfactory given the fact that it has been obtained through single camera observation and in the presence of occlusions.

4.5 Inferring angular velocity

The ball's angular velocity cannot be estimated by any direct vision method in any of the considered experiments. However, evidence regarding this parameter is encapsulated in the overall dynamic behavior of the ball. By seeking for a physically plausible explanation of the observed scene, the proposed approach reveals, as a byproduct, information regarding the hidden variable of angular velocity.

We performed a series of ball throws with high back-spin, so that the ball resists its original tendency to move forward (Fig.1). We then optimized S for the resulting 2D observations of camera 7. An exemplar bounce is shown in Fig. 1(b). The proposed method inferred the 3D state of the ball accurately (once more, ground truth and estimated ball positions are indistinguishable). Moreover, as demonstrated in Fig. 1(b), we were able to compute a qualitative measure of the ball's angular velocity.

5 Summary

We presented a method that interprets a dynamic scene by binding vision to physics based simulation. We combined a powerful optimization method and a detailed physics model of a bouncing ball in order to track the latter in challenging scenarios. We experimentally demonstrated that accounting for physics does not simply constitute yet another complementary source of information but rather, a strong prior that permits the treatment of underconstrained vision problems. In fact, we demonstrated that by incorporating physics, we may require less cameras/observations to obtain the same type of information or even gain access to information that is otherwise "invisible" to a vision system. Given the continuous advancement in optimization techniques [16], simulation tools and computational power,

we believe that the proposed method holds great potential towards addressing problems of greater dimensionality and complexity.

Acknowledgments

This work was partially supported by the IST-FP7-IP-215821 project GRASP. The contributions of E. Tzamali, member of CML/FORTH, and P. Padeleris, member of CVRL/FORTH, are gratefully acknowledged.

References

[1] A. Armenti. *The Physics of Sports*. Copernicus Books, 1992.

[2] P.J. Aston and R. Shail. The Dynamics of a Bouncing Superball with Spin. *Dynamical Systems*, 22(3):291–322, 2007.

[3] K. Bhat, S. Seitz, J. Popović, and P. Khosla. Computing the Physical Parameters of Rigid-body Motion from Video. In *ECCV 2002*, pages 551–565. Springer, 2002.

[4] M. Brand. Seeing physics, or: Physics is for prediction. In *Proceedings of the Workshop on Physics-based Modeling in Computer Vision: June 18-19, 1995, Cambridge, Massachusetts*, page 144. IEEE Computer Society, 1995.

[5] M. Brand. Physics-Based Visual Understanding. *Computer Vision and Image Understanding*, 65(2):192–205, 1997.

[6] M Brubaker and Leonid Sigal. Physics-based Human Motion Modeling for People Tracking: A Short Tutorial. *Image (Rochester, N.Y.)*, pages 1–48, 2009.

[7] MA Brubaker and DJ Fleet. The Kneed Walker for Human Pose Tracking. In *CVPR 2008*, pages 1–8. IEEE, 2008.

[8] M.A. Brubaker, L. Sigal, and D.J. Fleet. Estimating Contact Dynamics. In *ICCV 2009*, pages 2389–2396. IEEE, 2009.

[9] M.A. Brubaker, D.J. Fleet, and A. Hertzmann. Physics-based Person Tracking Using the Anthropomorphic Walker. *International Journal of Computer Vision*, 87(1):140–155, 2010.

[10] U.K. Chakraborty. *Advances in Differential Evolution*. Springer Publishing Company, Incorporated, 2008.

[11] H.T. Chen, M.C. Tien, Y.W. Chen, W.J. Tsai, and S.Y. Lee. Physics-based Ball Tracking and 3D Trajectory Reconstruction with Applications to Shooting Location Estimation in Basketball Video. *Journal of Visual Communication and Image Representation*, 20 (3):204–216, 2009.

[12] O. Cossairt, S. Nayar, and R. Ramamoorthi. Light Field Transfer: Global Illumination Between Real and Synthetic Objects. In *ACM SIGGRAPH 2008*, pages 1–6. ACM, 2008.

[13] S. Das, A. Abraham, and A. Konar. Particle Swarm Optimization and Differential Evolution Algorithms: technical analysis, applications and hybridization perspectives. *Advances of Computational Intelligence in Industrial Systems*, pages 1–38, 2008.

[14] Q. Delamarre and O. Faugeras. 3D Articulated Models and Multiview Tracking with Physical Forces. *Computer Vision and Image Understanding*, 81(3):328–357, 2001.

[15] D.J. Duff, J. Wyatt, and R. Stolkin. Motion Estimation using Physical Simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1511–1517. IEEE, 2010.

[16] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing Results of 31 Algorithms from the Black-box Optimization Benchmarking BBOB-2009. In *Proceedings of the 12th Annual Conference Comp on Genetic and Evolutionary Computation*, pages 1689–1696. ACM, 2010.

[17] I. Ihrke, K.N. Kutulakos, H. Lensch, M. Magnor, and W. Heidrich. Transparent and Specular Object Reconstruction. In *Computer Graphics Forum*. Wiley Online Library, 2010.

[18] D. Metaxas and D. Terzopoulos. Shape and Nonrigid Motion Estimation through Physics-based Synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.

[19] S.K. Nayar and S.G. Narasimhan. Vision in Bad Weather. In *ICCV 1999*, volume 2, pages 820–827. IEEE, 1999.

[20] V. Papadourakis and A. Argyros. Multiple Objects Tracking in the Presence of Long-term Occlusions. *Computer Vision and Image Understanding*, 114(7):835–846, 2010.

[21] Z. Popović and A. Witkin. Physically Based Motion Transformation. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 11–20. ACM Press/Addison-Wesley Publishing Co., 1999.

[22] R.J. Sethi and A.K. Roy-Chowdhury. Physics-based Activity Modelling in Phase Space. In *Seventh Indian Conference on Computer Vision, Graphics and Image Processing*, pages 170–177. ACM, 2010.

[23] R. Storn and K. Price. Differential Evolution–A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4): 341–359, 1997.

[24] M. Vondrak, L. Sigal, and OC Jenkins. Physical Simulation for Probabilistic Motion Tracking. In *CVPR 2008*, pages 1–8. IEEE, 2008.

[25] T.E. Zickler, P.N. Belhumeur, and D.J. Kriegman. Helmholtz Stereopsis: Exploiting Reciprocity for Surface Reconstruction. In *International Journal of Computer Vision*, volume 49, pages 215–227. Springer, 2002.

FORTH-ICS / TR-420 July2011

A GPU-powered computational framework for efficient 3D model-based vision†

Nikolaos Kyriazis, Iason Oikonomidis, Antonis A. Argyros

[0]† This work was partially supported by the IST-FP7-IP-215821 project GRASP.

A GPU-powered computational framework for efficient 3D model-based vision

Nikolaos Kyriazis, Iason Oikonomidis, Antonis A. Argyros

Computational Vision and Robotics Laboratory
Institute of Computer Science
Foundation for Research and Technology — Hellas (FORTH)
N. Plastira 100, Vassilika Vouton
Heraklion, Crete, 700 13 Greece

Web: `http://www.ics.forth.gr/cvrl`
E-mail: {kyriazis | oikonom | argyros}@ics.forth.gr
Tel: +30 2810 391600, Fax: +30 2810 391601

Technical Report FORTH-ICS / TR-420— July2011

Abstract

We present a generic computational framework that exploits GPU processing to cope with the significant computational requirements of a class of model-based vision problems. We study the structure of this class of problems and map the involved processes to contemporary GPU architectures. The proposed framework has been validated through its application to various instances of the problem of model-based 3D hand tracking. We show that through the exploitation of this framework near real-time performance is achieved in problems that are prohibitively expensive to solve on CPU-only architectures. Additional experiments performed in various GPU architectures demonstrate the scalability of the approach and the distribution of the execution time among the involved processes.

1 Introduction

In computer vision, several problems are solved by employing search and optimization methods. For example, in top-down model-based approaches, model instantiations are searched for or fitted in observations, in specific contexts. In this work we are interested in the class of top-down methods that estimate the pose and/or track the articulation of piecewise rigid objects, based on multiple visual cues, such as color images, depth maps etc. Depending on the employed model, the problem can be as simple as the recovery of the 3D position and pose of a rigid object or as complex as the tracking of articulated objects such as the human body or hand.

This broad class of problems is amenable to a generic formulation. For a given object model, candidate hypotheses can be generated. Given a process that quantifies the compatibility of a hypothesis to the observations and a systematic approach for generating the hypotheses, the problem can be solved by searching for the best scoring hypothesis. The scoring process ultimately reduces to comparing model hypotheses and observations at the pixel level, on a pre-defined feature space. The major drawback of this generic formulation is its computational requirements stemming from the significant processing associated to the scoring criterion as well as to the evaluation of mutliple hypotheses. Thus, the definition of an efficient computational framework that addresses the computational requirements of related methods is very important to their deployment in real-world applications. In this work, we propose such a computational framework that is based on the careful mapping of the involved processes to GPU architectures.

2 Relevant Work

Many computer vision tasks are susceptible to GPU acceleration. This is well understood and has lately been availed by a number of researchers, who have proposed efficient solutions for a variety of vision problems. For example, Choudhary et al. [1] were able to accelerate bundle adjustment and perform faster large-scale 3D reconstruction through a CPU/GPU architecture. Fulkerson and Soatto [5] provided a CUDA-based GPU implementation of the exact quick shift algorithm that enables $10 - 50$ times faster image segmentation compared

to a CPU implementation. Gwosdek et al. [7] performed fast variational optic flow computations by accelerating the employed Fast Explicit Diffusion Solver on the GPU using CUDA. Stühmer et al. [17] accelerated a Generalized Thresholding Scheme using CUDA, in order to also perform variational optic flow, in real-time. Tzevanidis et al. [18] constructed textured 3D meshes from multicamera observations in real time, by means of an efficient CUDA-based 3D visual hull computation. Zhu et al. [20] studied GPU accelerated dense stereo computations and demonstrated that both accuracy preservation and speed increase can be achieved in a transition from CPU-based to GPU-based algorithms.

GPU-based solutions have also been proposed for instances of the problems of pose estimation and articulated object tracking, that is the class of problems relevant to this work. De La Gorce et al. [2] employed a realistic textured 3D model of a human hand in order to track it in image sequences. GPU acceleration came in the form of a straightforward rendering mechanism that was invoked in a per hypothesis basis. Hammer et al. [8] took GPU acceleration further by considering multiple hypotheses simultaneously in tiled renderings, again with respect to 3D hand tracking. Ganapathi et al. [6] followed a similar approach for the problem of 3D human body pose estimation. Although brief, the described GPU acceleration raises some important issues that are also addressed in this work. Friborg et al. [4] reported a detailed CUDA-based implementation of an inherently parallel likelihood computational scheme, in the context of articulated object tracking.

Despite the significant amount of existing work, there has not been a generic CPU/GPU framework that considers a class of problems rather than a single problem instance. Thus, in this work, we systematically study the structure of a class of 3D model based vision problems and we propose a generic CPU/GPU framework for addressing their significant computational requirements. In spite of the common trend, we select a GPU-independent software architecture, Direct3D, that makes the application of our framework GPU invariant and, simultaneously, demonstrates that former GPU software pipelines are not obsolete. Still, we do consider the potential benefits from graphics-free GPU architectures, such as CUDA and OpenCL and we provide a brief discussion on their applicability and merits to our problem. The usefulness of the adopted approach is documented through experimental results obtained from the application of the proposed framework to various instances of the problem of 3D hand tracking [12–14] which show that the proposed framework achieves almost real-time performance in this type of problems. Additional experiments performed in various GPU architectures demonstrate the scalability of the approach and the distribution of the execution time among the involved processes.

3 Methodology

The high-level outline of the proposed framework is illustrated in Fig. 1. We assume that an *observation* process that is executed on the CPU, feeds the framework with raw input data. These data (acquired from a single or a multiple camera system and/or other sensing modalities) are then transformed into visual cues, that are relevant to the task at hand, through *preprocessing* (e.g., background subtraction, edge detection). Given the resulting visual cues,

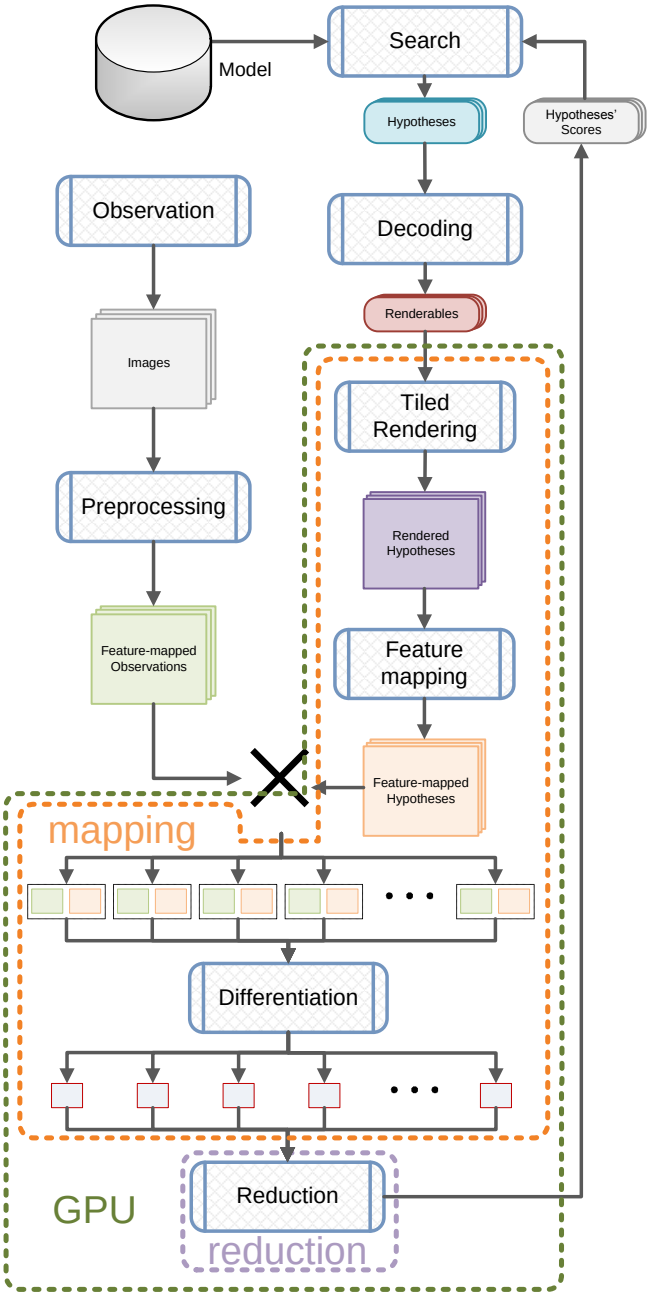


Figure 1: The proposed computational framework. Every iteration of the defined loop amounts to an efficient differentiation of observations and hypotheses, in a top-down, model-based approach.

a *search* process iteratively estimates the parameters of a preselected *model* by testing the compatibility of various hypothesized model instances to the visual cues. The hypothesized model instances are *rendered* and then *feature-mapped*, so as to be comparable to the observations at the pixel level. During each iteration of the *search* process, hypotheses rendering is performed on a big tiled map by a *tiled rendering* process. Since the observed visual cues and the rendered model hypotheses are comparable at the pixel level, their *differentiation* and *reduction* results in a total observation-hypothesis distance. All computed distances are then exploited by the *search* process so that better hypotheses can be further explored. The whole process terminates as soon as a termination criterion (usually related to the achieved accuracy and/or a predefined number of iterations) has been met.

3.1 Accelerated vs non accelerated processes

From the briefly described processes, *observation* is performed on the CPU and *preprocessing* can be executed either on the CPU or on the GPU. It has been observed [12–14] that these processes only consume a small fraction of the total execution time and therefore their acceleration does not have a significant impact on the overall computational performance.

We consider the *Black Box Optimization* paradigm [9], according to which a *search* process/heuristic investigates the hypothesis space of a *model* in order to identify the hypothesis that optimally fits a set of observations. There is only marginal gain in accelerating the core of such processes, as they are usually very fast. However, it is crucial to accelerate the hypothesis evaluation phase, whose execution time is dominant. Search heuristics can be categorized into serial and parallel, according to their evaluation scheme [16]. Serial heuristics are restricted to evaluate a single hypothesis at a time, while parallel heuristics do not pose such restrictions. Parallel heuristics allow for great acceleration gains in their objective function evaluation phase.

After sets of hypotheses have been proposed, they are decoded into renderable entities, i.e. arbitrarily complex 3D geometric instances. *Decoders* are responsible for mapping a multi-dimensional search space into a decomposition over geometries and their transformations (e.g., kinematics). In the proposed architecture decoders are dynamic libraries, that are loaded and selected at run-time. Decoding is worth accelerating since it involves series of matrix multiplications (geometry transformations).

The described iterative process follows the *map-reduce* scheme [3]. During the *mapping* phase, hypotheses are generated and paired to the corresponding observations (cross product notation in Fig. 1). At the *reduction* phase, the differences of the defined pairs are reduced into hypotheses scores that guide the *search* process. Both phases are GPU accelerated. Since the architecture we propose is generic, in the following sections we only describe the design requirements for each of the involved processes, emphasizing issues related to the GPU acceleration and data communication. Still, in Sec.4, we also provide some details on how the proposed framework is instantiated to efficiently solve various instances of the problem of 3D hand tracking.

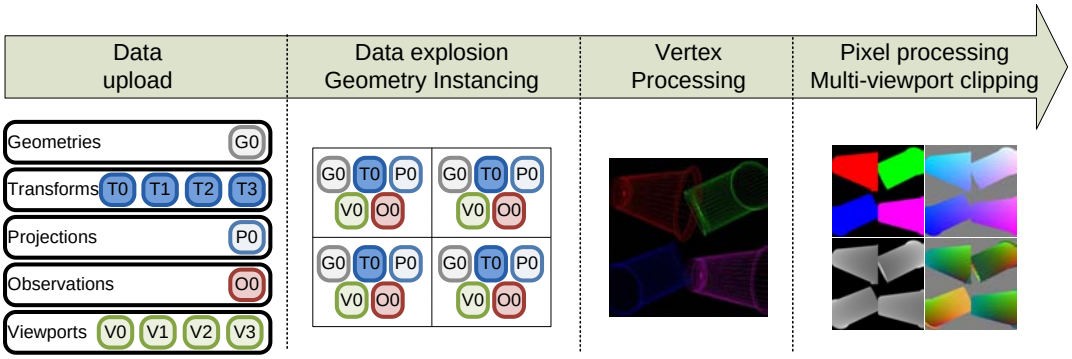


Figure 2: The tiled rendering process. Unique data are uploaded to the GPU, exploded into a tiled plan, processed in the vertex level and output in primary maps for later processing. Although there might be overlap of projected geometry across tiles during vertex processing this is remedied at the pixel-processing stage.

3.2 Tiled rendering

This process receives renderable entities and produces a set of primary outputs for further processing. Renderable entities amount to 3D geometries and geometry transformations and the primary outputs represent view-space 3D information, i.e. per pixel color, 3D position, 3D normal and depth. The input is provided by the *decoder* process. The outputs are post-processed by *feature mapping* and are thus made directly comparable to the respective observations.

Contemporary GPU drivers, GPU architectures and rendering pipelines allow for surprisingly fast parallel processing of massive data. Given proper design, linear increase of data may induce sub-linearly increased execution times, thanks to efficient interleaving of processing/reading/writing instructions [11]. We take parallelization to the limit by processing multiple hypotheses simultaneously. Instead of rendering a single hypothesis at a time we render multiple hypotheses in big *tiled renderings*. We separate the rendering and feature mapping phases in favor of modularity, by employing *deferred shading*. We perform efficient data communication and rendering by employing *geometry instancing*. Proper containment of instantiated geometry in tiles is achieved through *multi-viewport clipping*.

3.2.1 Deferred shading

Deferred shading is a technique commonly used in computer graphics [15]. According to this technique, 3D models are rasterized into a series of *primary outputs* such as color, position, depth and surface normal maps. Essentially, each primary output is an image with each pixel holding 3D information for the rendered models. It is characterized as “deferred” because actual rendering is postponed and at this phase only primary output is produced. An example of *deferred shading* is illustrated in Fig. 3. We employ *deferred shading* for two reasons: (a) we need to isolate the feature mapping process from the rest of the processes for the sake of modularity, (b) primary data may require multiple passes of processing but should not be

(a) (b) (c) (d) (e)

Figure 3: Deferred Shading. (a) the input of 3D model to be rendered and the primary outputs: (b) color, (c) position, (d) depth and (e) normal maps.

multiply computed.

Deferred shading, as a conventional rendering process, involves two successive stages, namely *vertex shading* and *pixel shading*. Given 3D models and view parameters, vertex shading is responsible for transforming every vertex of the input geometry from world coordinates to view coordinates. Then, triangles that are defined by the transformed vertices are rasterized during pixel-processing. Additionally, z-buffering is used in order to produce correct rasterizations of occluded geometry. Each stage is configurable through the integration of appropriate *vertex* and *pixel shaders*, i.e. callback routines, that perform the required processing on the GPU. We implement such shaders in the proposed framework.

3.2.2 Geometry instancing

It is beneficial to reduce communication between the CPU and the GPU in order to eliminate the respective overhead in execution time. For several computational schemes it is common that transferring data across memory spaces is more costly than processing itself. Best performance is achieved when only a few data are transferred to GPU, which are then "exploded" and processed and then "imploded" and transferred back to CPU.

In this work, each tile in a tiled rendering is associated with an actual camera view and a hypothesis. Therefore, for each tile, the following information is required: (a) geometry to be rendered, (b) world transform of this geometry, (c) projection matrix for the actual view and (d) the coordinates of the tile's viewport in the big rendering. However, in a tiled rendering, multiple tiles might refer to the same geometry and/or the same camera view. Therefore, there can be a lot of data reuse in the computations. This is amplified in the quite common case where the geometry itself is so modular that its sub-parts are heavily reused as well (e.g. as in [12], where the hand model consists of appropriate transforms of two geometric primitives). In that case it is highly inefficient to replicate data wherever they are required. Fortunately, in newer *shader models* (3 and above), *hardware instancing* enables an efficient and implicit replication of heavily reused data. Thus, we only upload unique data to the GPU once and explode them during processing by means of indexed referencing. The indexing itself is also implicit as we only define a replication pattern rather than the indices themselves (e.g., repeat datum every n tiles). The corresponding implosion occurs during the reduction phase (Sec. 3.5).

3.2.3 Multi-viewport clipping

If multiple tiles are to be rendered simultaneously it must be guaranteed that the rendered geometry is properly contained therein. That is, we want to avoid rendering the geometry of a given tile over neighboring tiles. However, conventional rendering pipelines do not consider this special case. While geometry clipping is traditionally performed at the vertex processing stage it is complicating to do so in our case. This is because of irregularities in the amount of data that survives clipping which would require an intricate remedy with respect to our target platform.

We chose to perform multi-viewport clipping at the pixel-processing stage where it is both convenient and efficient. After all geometry has been rasterized, a custom pixel shader is invoked to produce the primary map outputs. During geometry instancing, viewport information is attached to every vertex (see Sec. 3.2.2). During rasterization, this information is transferred to the pixels that result from the triangles formed from the processed vertices. Therefore, at pixel-processing each pixel is associated with the viewport in which it should be contained. A custom pixel shader clips pixels that are outside their pre-defined viewports (see Fig. 2).

3.3 Feature mapping

The main task of this process is to make rendered hypotheses comparable to the observations. This is performed by post-processing the 3D information that is contained in the primary outputs of the rendering process. Exemplar *feature mapping* processes can be: (a) the computation of occupancy from the position map, (b) the computation of edges from the normal map, (c) the computation of discrete layers from the depth map, etc. *Feature mapping*, as a straightforward rendering step, is a highly parallel pixel-wise mapping of the primary maps.

3.4 Differentiation

Differentiation is used to quantify the discrepancy between all feature-mapped observations and the corresponding feature-mapped hypotheses. With geometry instancing (Sec. 3.2.2) already one part of the observation-hypothesis pairing has been computed: every hypothesis has been associated to a part of a big tiled rendering which constitutes one operand of the differentiation. The remaining operand needs to be computed in accordance with the aforementioned pairing.

All observations are uploaded to the GPU in the form of multi-channel real-valued 2D textures at appropriate predefined slots. Each texture is assigned to a zero-based slot index that also constitutes its reference. During instancing this information is passed to each tile, so that the contained hypothesis corresponds to an observation. Thus, two big GPU textures are defined: (a) an explicit texture that holds the tiled rendering (first operand) and (b) an implicit tiled texture that is defined by the observation textures (second operand). Those two textures are now susceptible to any pixel-wise differentiation process that quantifies the discrepancy between observations and hypotheses. The result of this differentiation is stored in a multi-channel real-valued 2D texture.

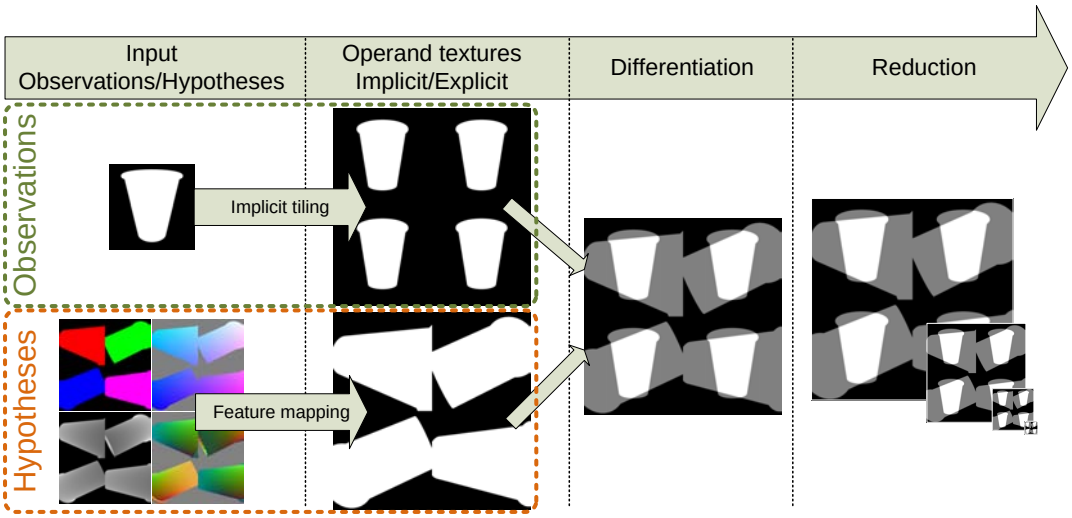


Figure 4: The differentiation process. Primary maps are mapped to the observations' feature space. Observations are implicitly tiled so as to match the tiled rendering of all hypotheses. A pixel wise differentiation is applied and the result is finally summed over the logical tiles by means of subsampling (data implosion).

3.5 Reduction

Once the mapping phase has been completed, the pixel-wise differences (Sec. 3.4) must be reduced to a single value for each tile, that ultimately represents the corresponding hypothesis' distance from the respective observation. We perform this reduction by means of pyramidal computations. Reduction is performed iteratively by subsampling the difference texture with a reduction operator (most commonly the sum operator). This scheme corresponds to the computation of a given MIP (*Multo In Parvo*, i.e., "much in a small space") level of the differences texture [19]. Reduction stops at the level at which the dimensionality of the resulting texture matches the dimensionality of the hypotheses' grid. For the sake of efficiency we select square render and tile sizes that are powers of two. For this process two textures are used that are exchanged in the positions of the input and the output in every subsampling iteration.

4 Applications

The proposed framework has been employed in [12–14], where different instances of the 3D hand tracking problem are treated. In order to exemplify the use of the proposed CPU/GPU architecture, we select the works in [14] and [13] because they differ in the observation model, the employed visual cues and, consequently, in the differentiation and reduction phases. Thus, they are the most diverse from a computational point of view. In both works different variations of the PSO algorithm [10] have been employed as the *search* process. PSO follows a parallel evaluation scheme which is favorable to the employment of the presented framework. The *rendering* process is the same in both cases, thanks to the *decoder* process.

In [14], the *model* accounts for a 3D articulated human hand and a parametric 3D object

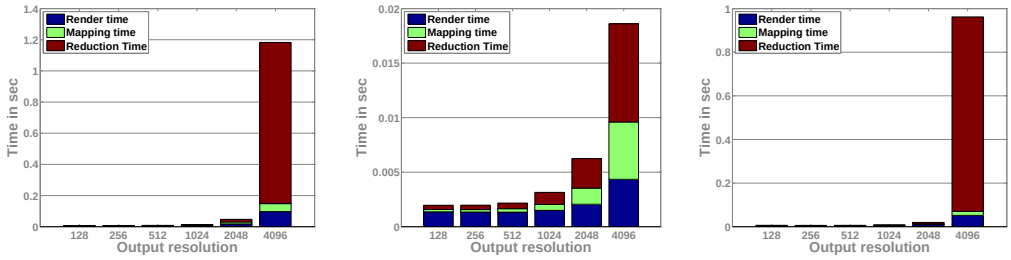
(ellipsoid, cylinder, cuboid) that amount to a total of $34 - 35$ DoFs. The observation process produces multi-frames of up to 8 RGB images per invocation. During preprocessing, input multi-frames are transformed into multi-frames of skin detection results m_s and multi-frames of edge detection results m_{DT} that have undergone a Distance Transform. The decoding phase maps the $34 - 35$ DoFs (depending on the number of DoFs of the object model) to appropriately transformed instances of ellipsoids, cylinders and cuboids. The results of the rendering process are feature-mapped to (a) occupancy images m_c that are produced from the primary color map and correspond to skin detection results and (b) edge detection results m_e, that are produced from the primary normal map and correspond to the edge detection results. The differentiation process generates 4 outputs: (a) $m_s \vee m_c$, (b) $m_s \wedge m_c$, (c) m_e and (d) $m_{DT} \cdot m_e$, where all operations are pixel-wise. Each such output is then reduced, in a per-tile basis, in order to provide tuples of 4 values for each hypothesis, using the sum operator. Each tuple is transformed (more details in [14]) into a single value that represents the distance between a given observation and a hypothesis. The employment of our framework in this application induces a performance of $2fps$ on an Intel Core i7 950 @ $3.07GHz$ with a NVIDIA GeForce GTX 580 GPU. Sample video results are available at `http://youtu.be/N3ffgj1bBGw`.

In [13], the *model* accounts for a 3D articulated hand of 26 DoFs (6 for the hand global position and pose and 20 for hand articulation). The observation process generates a frame set that is composed of a RGB image m_{RGB} and a depth map m_D. During preprocessing, skin detection is applied to m_{RGB} and its result is used to filter out depths, that do not regard the observed hand from m_D, in a new depth map m'_D. The decoding phase maps the 26 DoFs to appropriately transformed instances of ellipsoids and cylinders. The depth primary map that results from the rendering process is subtracted from m'_D, and these differences are summed over each tile to provide the observation-hypothesis distances. The employment of our framework in this application induces a performance of $15fps$ on an Intel Core i7 950 @ $3.07GHz$ with a NVIDIA GeForce GTX 580 GPU. Sample video results are available at `http://youtu.be/Fxa43qcm1C4`.

5 Experiments

The accuracy and computational performance of the proposed framework has been evaluated already in the context of the computer vision problems in which it was employed [12–14]. In this paper, further experiments were designed to highlight the strengths and weaknesses of this framework. More specifically, we show that our framework is able to perform in the order of tens of thousands evaluations of complex 3D hypotheses per second, even on mediocre hardware. Moreover, we identify a bottleneck in the pixel-processing stages, which leaves room for further improvement in performance.

We performed a set of experiments on 3 distinct systems of varied computational power: (a) an Intel Core 2 6600 @ $2.4GHz$ with a NVIDIA GeForce 9600 GT GPU, (b) an Intel Core i7 950 @ $3.07GHz$ with a NVIDIA GeForce GTX 580 GPU and (c) an Intel Core i7 930 @ $2.8GHz$ with a NVIDIA GeForce GTX 295 GPU. The third system has a dual GPU



(a) (b) (c)

Figure 5: Performance profiling of the GPU evaluation. Batches of 64 3D hand hypotheses are evaluated over output textures of increasing dimensions. Figures (a), (b) and (c) correspond to systems (a), (b) and (c). Interestingly, in (a) and (c) and for a resolution of 4096×4096, 87.46% and 92.69% of the time is spent on reduction. This suggests that, for these cases, smaller batches are preferable.

but for the experiments only one of them was used. The three systems vary in GPU core count (96 for system (a), 512 for (b) and 240 for (c)) and represent different GPU generations.

During all experiments we used a geometry that represents an articulated hand. The hand consists of 22 homogenous transformations of a spherical mesh and 15 homogenous transformations of a cylindrical mesh. Each sphere consists of $2 \cdot st \cdot sl$ triangles and each cylinder consists of $2 \cdot st \cdot sl + 2 \cdot sl$ triangles, where sl represents slice count and st represents stack count in a geometry sampling grid. Unless otherwise stated, the values of $sl = 10$ and $st = 10$ were used, so, the total triangle count per hypothesis was 7700. For more details the reader is referred to [12]. To consider a challenging scenario regarding rendering computational requirements, wherever a renderable hypothesis was required, one supplied with a hand fully occupying the viewport (open hand facing a virtual camera). For observations we used synthesized renderings of the same hypothesis (the content of the observations does not influence the computational performance). For the mapping and reduction processes, routines similar to those in [12] were used. Each reported measurement is the mean value of 20 successive iterations of the associated process. The timing operations themselves affect performance negatively as they define multiple synchronization points that break pipelining.

In a first line of experimentation we tested the method's computational efficiency under increasing resolution requirements for the tiled rendering, and thus put more weight on the pixel-processing phase. Batches of 64 hypotheses were rendered on a 8×8 grid and on square textures whose dimension was 128, 256, 512, 1024, 2048 and 4096. Respectively, the dimension of the square tile was 16, 32, 64, 128, 256 and 512. We considered the execution times of *rendering*, *mapping* and *reduction* separately. A total of $64 \times 7700 = 492800$ triangles were rendered in each batch. The obtained results are illustrated in Fig. 5. It can be verified that as the resolution increases, the processing time of the pixel-processing stages (rasterization during rendering, mapping, reduction) increases, too. For higher resolutions, reduction becomes the bottleneck, followed by mapping. The graceful degradation of performance for system (b) and for a resolution of 4096, as opposed to systems (a) and (c), is most probably due to

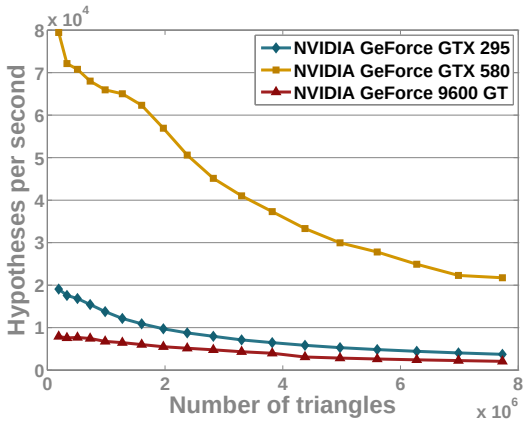


Figure 6: Rendering throughput against geometry complexity. 256 3D hand hypotheses are rendered on a 4096×4096 texture. Although 3D model detail increases with respect to triangle count, performance degradation is graceful due to hardware instancing.

improved memory architecture.

We also tested the rendering performance for increasing complexities of the 3D geometry and thus put more weight in the vertex processing phase. We repeated the previous line of experimentation, excluding mapping and reduction and fixing the tile dimension to 256 and the total dimension to 4096 (hypothesis count is thus 256). We varied the sl and st counts simultaneously in the range $[3, 20]$. This resulted in batches of triangle counts in the range $[193536, 7731200]$. Plots that demonstrate the graceful degradation of the performance are shown in Fig. 6. As the detail of the model increases, more vertices are required to be processed. Also, the tight spacing between small triangles in high levels of detail stresses the rasterization process, which has to also resolve more depth conflicts. Because of the simultaneous issuing of all geometry and due to efficient interleaving, the GPU performs well.

Finally, we tested the scalability of the proposed framework in order to highlight the benefit from considering batches of hypotheses simultaneously. We fixed the dimension of the tile to 128 and considered batches of 1, 4, 16, 64, 256 and 1024 hypotheses. This generated a requirement for output textures of dimensions 64, 128, 256, 512, 1024, 2048 and 4096, respectively. For each batch size we measured the hypothesis evaluation throughput by dividing its size with the required processing time. The results are shown in Fig. 7. Given the requirement for the evaluation of many hypotheses, it is evidently beneficial to consider larger batches for simultaneous evaluation. However, as the batch sizes grow, the required resolution is increased and the whole process becomes pixel-processing bound. Systems (a) and (c) were affected negatively, but system (b) remained unaffected.

In order to cope with batch sizes that are not power-of-two we processed them by sequential decomposition in power-of-two batches. The results of the previous experiment but for non-power-of-two batch sizes are shown in Fig. 8. Performance presents a pattern across sizes, due to the decomposition, and remained good for batch sizes that required few decom-

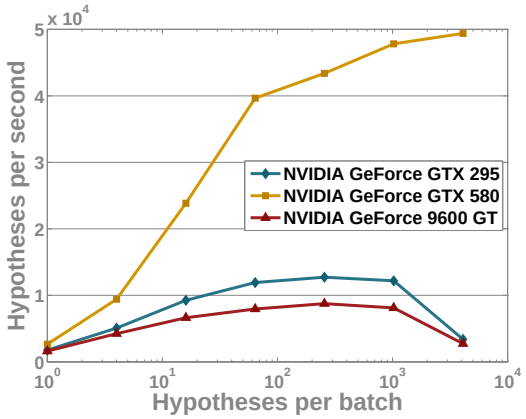


Figure 7: Evaluation throughput against batch sizes. Each hypothesis is evaluated on a 64×64 tile. As the batch size increases the evaluation throughput increases as well. The deterioration of performance for 1024 hypotheses for systems (a) and (c) is due to the requirement for an output texture of 4096×4096, which, as shown in Fig. 5, is problematic.

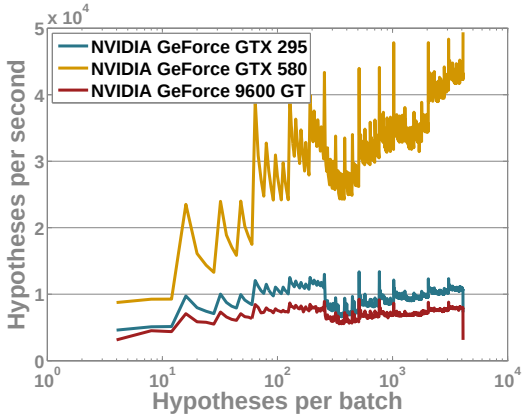


Figure 8: The same experiment as in Fig. 7, but for batch sizes that are not powers of two.

positions. State changes, like the requirement for different sizes of output textures in the same batch, affected performance negatively.

In every case, we were able to evaluate tens of thousands of complex 3D hypotheses per second. This has been most beneficial for the works in [12–14], where near real-time performance was achieved for different challenging instances of the problem of 3D hand tracking.

6 Conclusions

In this paper we provided a detailed description of a CPU/GPU framework that targets a class of computer vision problems. We demonstrated the computational benefits of this framework in a series of experiments performed on various GPU systems. The proposed framework is not restricted to a single application and has already been employed in three similar, yet essentially different problems. It has been shown that an effective GPU-invariant solution is realizable. More specifically, it has been demonstrated that the proposed framework can evaluate tens of thousands of elaborate 3D hypotheses per second. The efficiency of the proposed architecture makes it a likely candidate for several other real-time 3D computer vision applications.

Still, there is room for further improvements. We selected Direct3D as the rendering platform because the class of problems we are interested in optimally maps to the features provided by this platform. However, by also being graphics-oriented, the selected platform induces restrictions and limitations: (a) Direct3D, although GPU-invariant, is not OS-invariant, (b) data-parallel, non-graphics-oriented tasks have to be forced through a graphics-oriented pipeline, which induces unnecessary overheads, (c) numbers need to be powers-of-two in order to achieve optimal performance in a platform that does not favor elaborate gather/scatter operations. Preliminary tests have shown that further speedup can be achieved by complementing or replacing Direct3D with other platforms, such as CUDA, OpenCL etc. For example, a $20\times$ speed-up on the pixel-processing stages (Fig. 5), which was achieved through CUDA, might offer an overall $2 - 10\times$ speed-up for large resolutions (see Fig. 5).

Acknowledgments

This work was partially supported by the IST-FP7-IP-215821 project GRASP.

References

[1] Siddharth Choudhary, Shubham Gupta, and PJ Narayanan. Practical time bundle adjustment for 3d reconstruction on the gpu. In *ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010)*, 2010.

[2] M. de La Gorce, N. Paragios, and DJ Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *CVPR 2008)*, pages 1–8. IEEE, 2008.

[3] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[4] R.M. Friborg, Sø ren Hauberg, and Kenny Erleben. GPU Accelerated Likelihoods for Stereo-Based Articulated Tracking. In *ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010)*, 2010.

[5] Brian Fulkerson and Stefano Soatto. Really quick shift: Image segmentation on a GPU. In *ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010)*, 2010.

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

[7] Pascal Gwosdek, Henning Zimmer, Sven Grewenig, A. Bruhn, and J. Weickert. A highly efficient GPU implementation for variational optic flow based on the Euler-Lagrange framework. In *ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010)*, 2010.

[8] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *ICCV 2009)*, pages 1475–1482. IEEE, 2009.

[9] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1689–1696. ACM, 2010.

[10] J. Kennedy. Swarm intelligence. *Handbook of Nature-Inspired and Innovative Computing*, pages 187–219, 2006.

[11] Nvidia. Compute unified device architecture programming guide. *NVIDIA: Santa Clara, CA*, 83:129, 2007.

[12] I. Oikonomidis, N. Kyriazis, and A. Argyros. Markerless and efficient 26-dof hand pose recovery. In *ACCV 2010*, pages 744–757. Springer, 2010.

[13] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC 2011*. BMVA, 2011.

[14] I. Oikonomidis, N. Kyriazis, and A. Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *ICCV 2011*. IEEE, 2011.

[15] M. Pharr and R. Fernando. Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation. 2005.

[16] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[17] J. Stühmer, Stefan Gumhold, and Daniel Cremers. Parallel Generalized Thresholding Scheme for Live Dense Geometry from a Handheld Camera1. In *ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010)*, 2010.

[18] K. Tzevanidis, X. Zabulis, T. Sarmis, P. Koutlemanis, N. Kyriazis, and A. Argyros. From multiple views to textured 3d meshes: a gpu-powered approach. In *ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010)*, pages 5–11, 2010.

[19] L. Williams. Pyramidal parametrics. In *ACM SIGGRAPH Computer Graphics*, volume 17, pages 1–11. ACM, 1983.

[20] Ke Zhu, Matthias Butenuth, and Pablo Angelo. Comparison of Dense Stereo using CUDA. In *ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010)*, 2010.

Efficient Model-based Tracking of the Articulated Motion of Hands

Iason Oikonomidis, Nikolaos Kyriazis, Antonis A. Argyros

Institute of Computer Science, FORTH, Greece

AND

Department of Computer Science, University of Crete, Greece

PROBLEM STATEMENT

Track the 3D position, orientation and full articulation (26 DoFs) of a human hand that possibly manipulates an object, given a sequence of either multi-view or RGB-D frames of the scene.

MOTIVATION

The markerless tracking of hand articulations is a challenging problem with diverse applications such as H.C.I., understanding human grasping, robot learning by demonstration, etc.

MAIN IDEA

Jointly consider the observed scene: extract full-image features and produce full hypotheses about it. Compare hypotheses and observed features in parallel [4]. Use the resulting scores to drive an iterative optimization process using Particle Swarm Optimization (PSO) [5].

PROPOSED METHOD

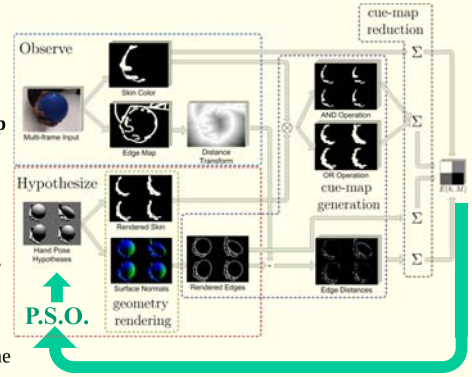
Input

- Frames are acquired by a **multi-camera** setup or a **Kinect**.
- **Edge** (black) and **skin color** (red) cues are extracted for the multi-view case.
- **Objects** are assumed **not** to be **skin colored**.

- For the case of Kinect, skin color and depth cues, along with the **temporal continuity** assumption are used to **segment** the hand.

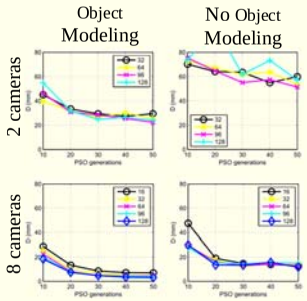
Fit model to data

- A parametric hand model is employed [1].
  - ❑ Comprised of 15 cylinders and 22 spheres.
  - ❑ 26 DoFs: 6 for global pose and 20 kinematics angles.
  - \+ For the hand-object case, add a parametric object (9 or 10 DoFs).
- From a full configuration (all 26 DoFs of the hand model plus potentially the object DoFs), a **skin occupancy map,** an **edge map** and a **depth map** can be **synthesized** by means of rendering.
- These **maps** are used to quantify the **discrepancy** between **observation** and **hypothesis** (objective function).
- The objective function also **penalizes physically implausible** configurations (hand-hand and hand-object collision checking).
- A **variant** of the **PSO** method [5] searches in the model parameter space for the best scoring configuration.
  - ❑ **Efficient** evaluation of **multiple** hypotheses on the **GPU** [4].
- Candidate poses for the **next frame** are obtained by **perturbing** the solution of the **previous frame**.



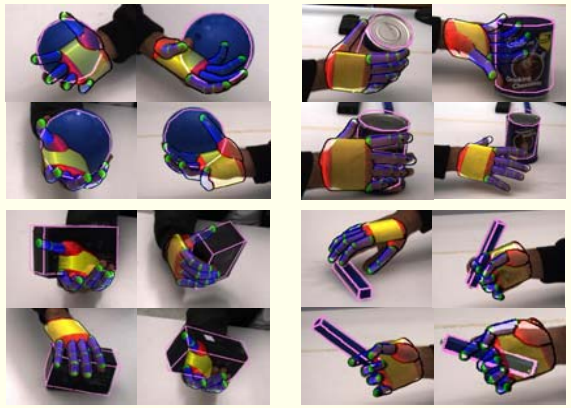
EXPERIMENTAL RESULTS

Quantitative evaluation on synthetic data



64 particles and 40 generations for 4 views yield 2fps on a modern PC

Input from a Multi-view System



Input from Kinect

- Single-view 3D cues
- Hand in isolation

64 particles and 30 generations yield 15fps on a modern PC

STRENGTHS OF THE APPROACH

- Occlusions serve as visual cues through modeling.
- Joint optimization: no simplifying assumptions over the problem structure, simultaneous consideration of all parameters.
- Careful design and exploitation of parallelism in a GPU implementation [4] lead to a computationally efficient system that accepts input of multiple modalities [1-3].
- Minimally invasive markerless approach.

KEY REFERENCES

1. Oikonomidis, I., Kyriazis, N., Argyros, A. A. "Markerless and Efficient 26-DOF Hand Pose Recovery". *ACCV*, 2010.
2. Oikonomidis, I., Kyriazis, N., Argyros, A. A. "Full DOF Tracking of a Hand Interacting with an Object by Modeling Occlusions and Physical Constraints". *ICCV*, 2011.
3. Oikonomidis, I., Kyriazis, N., Argyros, A. A. "Efficient Model-based 3D Tracking of Hand Articulations using Kinect". *BMVC*, 2011.
4. Kyriazis, N., Oikonomidis, I., Argyros., A. A. "A GPU-powered Computational Framework for Efficient 3D Model-based Vision". *Technical Report TR420, ICS-FORTH*, 2011.
5. Kennedy, J., Eberhart, R. "Particle swarm optimization". *International Conference on Neural Networks*, 1995.

For more information, visit http://www.ics.forth.gr/~oikonom or contact {oikonom, kyriazis, argyros}@ics.forth.gr

This work was partially supported by the IST-FP7-IP-215821 project GRASP

A GPU-powered Computational Framework for Efficient 3D Model-based Vision

Nikolaos Kyriazis, Iason Oikonomidis, Antonis A. Argyros

Institute of Computer Science, FORTH, Greece AND Department of Computer Science, University of Crete, Greece

PROBLEM

Provide efficient implementations for hypothesize-and-test vision methods that incorporate intense rendering as means of simulation.

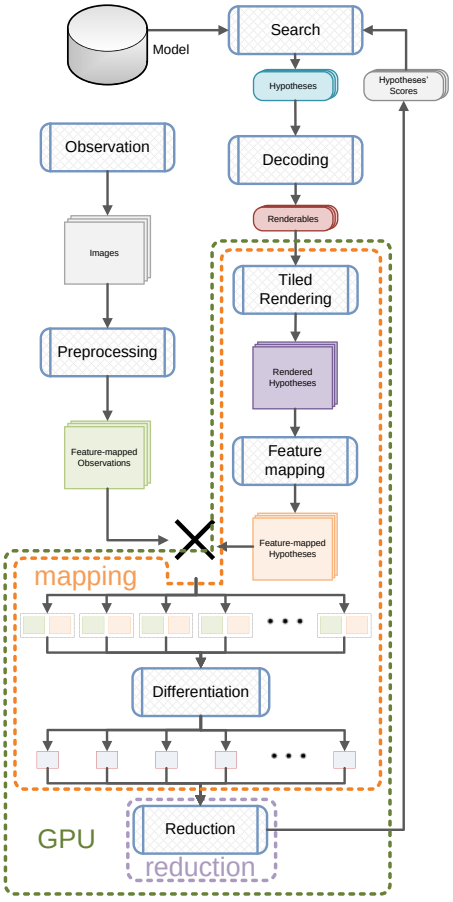
MOTIVATION

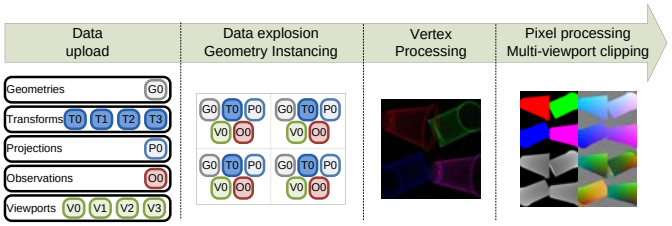
In computer vision, several problems are solved by employing hypothesize-and-test methods. Hypotheses can be made comparable to acquired images by means of 3D rendering.

MAIN IDEA

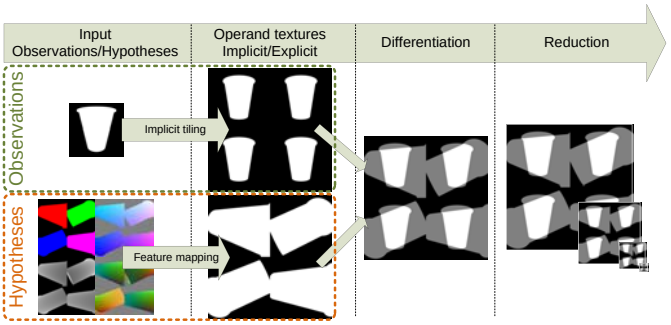
- 3D rendering is an inherently parallel process that is delegated to parallel hardware (GPUs)
- Parallel test/comparison criteria constitute the dominant case
- Exploitation of GPUs beyond traditional 3D rendering to satisfy the challenging computational demands of 3d model-based vision methods

METHOD





The tiled rendering process. Unique data are uploaded to the GPU, exploded into a tiled plan, processed in the vertex level and output in primary maps for later processing. Although there might be overlap of projected geometry across tiles during vertex processing this is remedied at the pixel-processing stage.



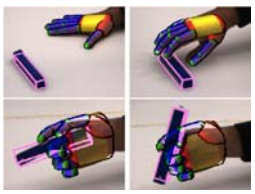
The differentiation process. Primary maps are mapped to the observations' feature space. Observations are implicitly tiled so as to match the tiled rendering of all hypotheses. A pixel wise differentiation is applied and the result is finally summed over the logical tiles by means of subsampling (data implosion).

APPLICATIONS

Tracking of "kinematic forests"

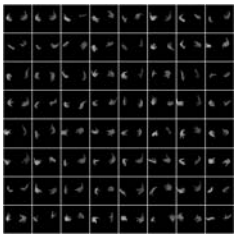
3D hand tracking from multiple cameras [1, 2]
(2 fps for 4 cameras)

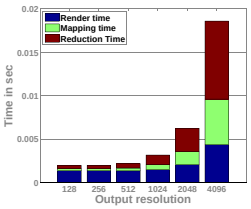
3D hand tracking from Kinect [1, 3]
(15 fps for 1 sensor)

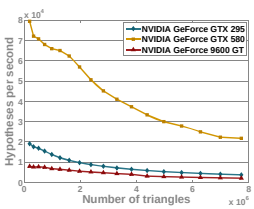


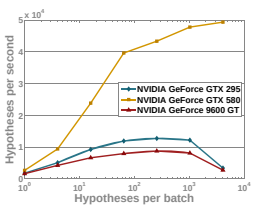
3D hand-object tracking from multiple cameras [1, 4]
(2 fps for 4 cameras)

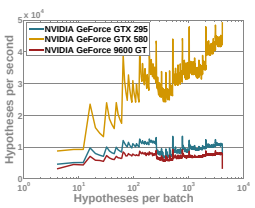
EXPERIMENTS











CONTRIBUTIONS

- Studied a challenging problem whose solution yields significant impact
- Identified a architecture with carefully designed modularity
- Presented an implementation that is based on GPU independent, commoditty pipeline, namely Direct3D 9
- Provided 3 distinct applications on the 3D articulated tracking problem

REFERENCES

[1] N. Kyriazis, I. Oikonomidis, and A. Argyros. A gpu-powered computational framework for efficient 3d model-based vision. Technical Report TR420, ICS-FORTH, July 2011.

[2] I. Oikonomidis, N. Kyriazis, and A. Argyros. Markerless and efficient 26-dof hand pose recovery. In *ACCV 2010*, pages 744–757. Springer, 2010.

[3] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC 2011*. BMVA, 2011.

[4] I. Oikonomidis, N. Kyriazis, and A. Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *ICCV 2011*. IEEE, 2011.

MORE INFORMATION

Web: http://www.ics.forth.gr/ kyriazis/?e=2 E-mail: {kyriazis,oikonom,argyros}@ics.forth.gr

This work was partially supported by the IST-FP7-IP-215821 project GRASP