Boosting the Performance of Model-based 3D Tracking by Employing Low Level Motion Cues

Ammar Qammaz[1]
ammarkov@ics.forth.gr
Nikolaos Kyriazis[1]
kyriazis@ics.forth.gr
Antonis A. Argyros[21]
argyros@ics.forth.gr

[1] Insitute of Computer Science, FORTH,
N. Plastira 100, Vassilika Vouton,
GR70013, Heraklion, Crete, Greece

[2] Computer Science Department,
University of Crete,
Heraklion, Crete, Greece

Abstract

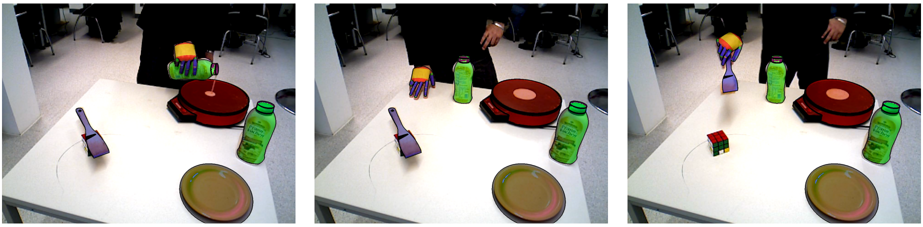
3D tracking of objects and hands in an object manipulation scenario is a very interesting computer vision problem with a wide variety of applications ranging from consumer electronics to robotics and medicine. Recent advances in this research topic allow for 3D tracking of complex scenarios involving bimanual manipulation of several rigid objects using commodity hardware and with high accuracy. The problem with these approaches is that they treat tracking as a search problem whose dimensionality increases with the number of objects in the scene. This fact typically limits the number of the tracked objects and/or the processing framerate. In this paper we present a method that utilizes simple low level motion cues for dynamically assigning computational resources to parts of the scene where they are actually required. In a series of experiments, we show that this simple idea improves tracking performance dramatically at a cost of only a minor degradation of tracking accuracy.

1 Introduction

There is a very large number of real-life scenarios in which a person manipulates several objects in the environment. Examples can be drawn from diverse origins, *e.g.* manual and industrialized assembly, tool use, food preparation, surgery operation, *etc*. Automating the extraction of rich and accurate 3D information regarding the state of such scenes through computer vision techniques is an interesting problem whose solution is of fundamental importance towards supporting goals such as activity recognition, inspection, surveillance, *etc*.

Interestingly, with the exception of rare cases such as juggling, it is usual that a person only manipulates a few objects or tools at a time. As an example, in a pancake cooking scenario (see Fig. 1), at each point in time the person actuates a bottle of pancake mix, a spatula, a plate, a cooker, *etc*., one after the other or in small subsets. While there are several objects that require accurate 3D tracking overall, only a few of them move at any given time. We argue that this simple observation is very valuable as its exploitation can dramatically improve the computational profile of 3D multi-object tracking methods.

Pages 144.1-144.11
DOI: https://dx.doi.org/10.5244/C.29.144



(a) Pouring pancake mix (b) Waiting for the mix to cook (c) Flipping the pancake

Figure 1: When making pancakes in real-life, several objects remain static for considerable time intervals. The proposed method can very efficiently skip through these idle times without sacrificing tracking accuracy. The tracked objects are the human hand, the spatula, the pancake mix bottle, the plate and the cooker.

This work deals with the problem of tracking multiple interacting objects so as to best utilize the limited processing resources of a computational system. Tracking multiple interacting objects in 3D is a difficult task. Adding a requirement for computational efficiency only makes the problem harder. Towards this goal, the presented work extends the state-of-the-art method by Kyriazis and Argyros [8] by introducing a dynamic distribution of computational budget among an Ensemble of Collaborative Trackers (ECT). The dynamic distribution of computational budget is based on change detection techniques, which draws cues from the RGBD input as well as from the feedback of the tracking method itself. Through extensive experiments and comparative analyses we show that the proposed work improves the state-of-the-art in multi-object 3D tracking by maintaining (*i.e.*, not compromising) their high accuracy but by doing so in fractions of the originally required computational time. For scenarios in which every object is moving the proposed method gracefully regresses to [8].

2 Related Work

To the best of our knowledge, the exploitation of low level motion cues to speed up the 3D tracking of multiple objects has not been addressed in the relevant literature.

With respect to 3D tracking of objects, there is a great amount of work which considers one or a few objects at a time and there are few approaches which deal with multiple objects. All reviewed methods include a bottom-up (*e.g.* preprocessing) and a top-down (*e.g.* fusion) component. There is variability, however, on the strength of the contribution and the significance of each of these components in the resulting method. Many works regard tracking a single hand in 3D. Hand-tracking methods in which the bottom-up component is prevalent are those based on machine learning [5, 6, 15, 19, 22]. Hand-tracking methods with a stronger top-down component are founded on hypothesize-and-test optimization [2, 9, 10, 11, 14, 17, 18]. There are also hand-tracking methods where the bottom-up and top-down components have balanced contributions [1, 3, 20, 23]. In tracking multiple interacting objects and/or hands there is more focus on the top-down component [7, 8, 12, 13, 16, 21]. This is an interesting fact, which we understand as follows: for the case of multiple objects with intense interaction, a model-based approach requires generalization from simpler object/scene models to more complex ones. This is achieved easily through intuitive top-down design. On the negative side, the computational requirements of such methods are exponential to the number of the objects to be tracked. Bottom-up, learning-based methods are not as easily generalizable to complex scenes. This is because extending the scene by even one new object requires retraining of the method from scratch. Moreover, the space of hand/object interactions that needs to be learned becomes easily extremely large and sampling it at a reasonable density becomes prohibitively costly, if possible at all.

The works that are most related to ours are the approaches by Kyriazis and Argyros on top-down 3D tracking of multiple active objects from RGBD input [7, 8]. In both approaches special attention is paid on scalability. In [7], it is assumed that the state of passive objects is invariably determined by the motion of an active hand. Therefore, they tracked a simulation configured over hand motion alone. This assumption effectively reduced the size of the tracking problem. However, if the physical model is kept simple, this assumption may be too restrictive in the nature of scenes amenable to such tracking, as acknowledged in [8]. The requirement for tracking multiple active entities in a scalable fashion is tackled in [8]. By employing a distinct tracker per object and by establishing communications among all trackers, the problem of joint estimation of the state of all objects is achieved. Each tracker is optimized independently, but in each independent optimization, evidence from other trackers is also considered.

In this work, as in [8], we employ an Ensemble of Collaborating Trackers (ECT) to allow tracking of multiple active objects. Nevertheless, in contrast to [8], each tracker has a variable computational budget allowance. We perform change detection to assess how much attention each of the trackers requires and distribute computational resources accordingly. By doing so, we are able to significantly improve the state-of-the-art tracking throughput results of [8] with negligible deterioration of accuracy.

3 Methodology

The methodological part of our contribution can be briefly described as an extra processing node in the pipeline of [8] which it extends. Our contribution, in terms of implications and impact, is described in Section 4.

The tracking approach in [8] regards a set of semi-independent trackers. Each tracker is associated with a distinct object in the scene. For an object to be tracked, a separate optimization problem is solved, one for each frame and each object. Each optimization problem is numerically solved using a black box optimizer, *i.e.* a variant of the Particle Swarm Optimization (PSO) algorithm. This algorithm treats the objective function as an oracle and queries it on purposefully evolved "guesses" in the search space, in order to find the optimum. The harder the problem is, the more guesses are required for adequate accuracy to be achieved. For example, tracking the pose and the articulation of the hand amounts to solving a 27-parameter optimization problem. This is much harder than solving for the 3D pose of a rigid object (6 parameters). ECT assigns a fixed amount of computational resources to each tracking sub-problem which depends on this notion of complexity and which is empirically estimated.

In this work, we quantify at run time how hard the tracking of an object should be, not only based on its intrinsic complexity, but also based on its observed dynamics. An object that appears static in the recent temporal window requires less resources compared to an object whose state is more dynamic. Thus, change detection is performed on the image space of color intensities and depth measurements of the RGBD input.

In Sec. 3.1, for the purposes of self-completeness, we provide a brief overview of how the baseline, ECT method [8] works. In Sec. 3.2 we present the visual cues that can provide valuable hints on the hardness of the tracking problem. Finally, in Sec. 3.3, we show how these cues are used to determine the computational budget of each tracker, resulting in the proposed method, F.ECT.

3.1 Collaborative Tracking (ECT)

The tracking methodology is built on the premise of a generative model **M**. **M** has as many degrees of freedom (DoFs) as required to fully specify the 3D pose and articulation of every object in a scene. Here, we are considering rigid objects and articulated hands. Rigid objects are described by their 3D shape, in the form of 3D triangular meshes, and 6 DoFs (3D translation and 3D rotation), and are redundantly represented by 7 values (rotations are represented by quaternions). The human hand is described by 26 DoFs and a set of triangular meshes for the parts, and are represented by 27 values, as in [9].

M can generate rendered images from such parametrizations of scenes, as in [8]. These images are directly comparable to RGBD input. Foreground detection in RGB can be corresponded to rendered silhouettes. Depth measurements can be corresponded to rendered depth maps. By differentiating the generated data and the observations, an objective function is established, defined over the parametrization of the scene. The minimum value of this objective function corresponds to the optimal fit of **M** to the observations, *i.e.* the best matching scene 3D configuration.

For each tracker and each frame in time this objective function is defined as:

$$\mathbf{E}(x,o,\hbar,D_T) = \|\mathbf{M}(x,\hbar,D_T) - \mathbf{P}(o,\hbar)\|, \tag{1}$$

where x is a hypothesized 3D configuration of the tracked object, o is the set of observations for the current frame, $\hbar$ is the tracking history, D_T is the state of the rest of the scene as broadcast by the rest of the trackers and **P** is the process of mapping the observations to the same feature space as the generated data (silhouettes and depth maps). The norm of the difference in Eq. (1) is a notational abstraction, which refers to the silhouette and depth differentiation computations of [9] (term D therein). Tracking, *i.e.* computing the object state s for each frame, amounts to minimizing Eq. (1) with respect to its free parameters x:

$$s \overset{\Delta}{=} \arg\min_{x} \mathbf{E}(x,o,\hbar,D_T). \tag{2}$$

M can generate data by also incorporating knowledge from the rest of the trackers D_T. The broadcast information D_T, incorporated in the computations of the current frame, corresponds to the estimates of all trackers in the previous frame (see one frame lag reference in [8]). In our case, and for each tracker and frame, **M** incorporates all tracker information into a single silhouette map, by computing the pixel-wise logical disjunction of all rendered silhouettes for all trackers, and into a single depth map, by performing z-buffering over all rendered depth maps for all trackers.

The required minimization of Eq. (2) is performed by the PSO variant described in [9]. Search is initialized in the vicinity of the solutions for the previous frame. The variant is differentiated from the original PSO algorithm [4] in that it performs an additional step of randomization in the finger articulation parameters of the hypotheses, during evolution. This implies that for the case of rigid objects the PSO variant is the same as the original PSO.

The only free parameters of the described procedure are the budget allocated to PSO for each optimization required, across all trackers and frames. In [8] these budgets were predetermined and always remained fixed during tracking. In this work, the main goal is the evidence-based dynamic adjustment of these budgets.

3.2 Low Level Motion Cues

The expected complexity of tracking an object in each frame depends on whether it moves or not. This regards the image-space vicinity of the object and incorporates low-level features, tracking feedback and a temporal smoothness prior.

During tracking and for each object, there is a 3D state estimate held by the corresponding tracker, regarding the last processed frame. The 3D estimate for the i^{th} tracker is individually back-projected to the image plane and the 2D bounding box b_i of the back-projection is computed. Given color intensity images $I_{c,t}$ and depth images $I_{d,t}$, defined over time t, the amount of motion for object i is quantified as the average pixel-wise difference inside b_i, between the current frame, at time t, and the previous frame, at time $t-1$:

$$m_{l,i}(t) = \frac{1}{N_{b_i}} \sum_{p \in b_i} \left\| I_{l,t}(p) - I_{l,t-1}(p) \right\|, \tag{3}$$

where N_{b_i} denotes the area of the bounding box b_i. Accordingly, for each tracker i, motion intensity in RGB images is defined as $m_{c,i}$ and motion amount in depth measurements is defined as $m_{d,i}$.

Assuming temporal continuity, we also incorporate the tracked velocity of each object into the computations of its motion state. For each object i and for the frame in time t, with 3D state estimates for the two previous frames $s_{i,t-1}$ and $s_{i,t-2}$, we compute the magnitude of the velocity:

$$\bar{v}_i = \left\| s_{i,t-1} - s_{i,t-2} \right\|. \tag{4}$$

In depth measurements, a value of 0 represents the lack of measurement. There are several reasons that lead to missing measurements in structured light depth acquisition. We are specifically interested in the sudden change of the number of missing measurements, which usually occurs as the angle at which object surfaces are viewed is close to some tolerance of slant. To capture such events we quantify this structured noise as follows:

$$e_i = \frac{1}{N_{b_i}} \left\| \sum_{p \in b_i} \left(I_{d,t}(p) \neq 0 \right) - \sum_{p \in b_i} \left(I_{d,t-1}(p) \neq 0 \right) \right\|. \tag{5}$$

Each object i is also corresponded to a timeout value $\mathfrak{t}_i$. This value holds the number of frames, back in time, since object i was last classified as moving. Thresholding of the aforementioned quantities yields the classification of the motion state for object i. More specifically, given thresholds T_c regarding motion in color images, T_d regarding motion in depth images, T_v regarding velocity magnitude and T_e regarding structured noise, each object i is given a motion label $\mathfrak{m}_i$, which is set to 1 if the object is classified as moving and 0 otherwise:

$$\mathfrak{m}_i = \begin{cases} 1, & (m_{c,i} > T_c) \vee \left(m_{d,i} > T_d\right) \vee (v_i > T_v) \vee (e_i > T_e) \vee (\mathfrak{t}_i > 0) \\ 0, & \text{otherwise} \end{cases}. \tag{6}$$

In special note, if $\mathfrak{m}_i$ is set to 1 due to $\mathfrak{t}_i > 0$, alone, then $\mathfrak{t}_i$ is not reset. Timeout is suppressed in the presence of evidence suggesting so. If the last T_g generations, computed

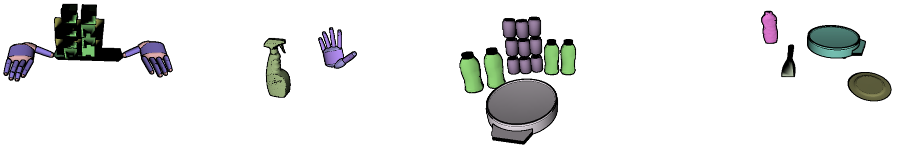


Figure 2: Frames from the tracked sequences: “two hands”, “spray”, “pancakes”, “cans”.

for the tracking of object i in the previous frame, did not lead to an improvement of the state estimate then $\mathfrak{t}_i$ is set to 0.

3.3 Dynamic Budget Distribution (F.ECT)

As described in Sec. 3.1, Particle Swarm Optimization (PSO) operates by evolving a population of particles. Evolution is performed in steps, called generations. In PSO, two main phases comprise the computation of a generation. A first phase is the fitness computation of all particles. Fitness computation is the computationally heaviest operation, as it involves the evaluation of the objective function at the hypotheses carried within the particles. This is a parallel step, as the evaluation of each particle is independent with respect to any other. A lightweight step follows, where, according to the fitness evaluations, all particles “fly” towards the optimal particle. The amount of total objective function invocations, which is equal to the product of particles and generations, is the budget required by a single optimization. If several similar optimizations are carried out, as it is the case here and in [8], generation computations across such optimizations can be merged into wider parallel computations performed on GPGPU architectures. Thus, the same number of objective function evaluations are executed more quickly if it is the product of more particles and less generations.

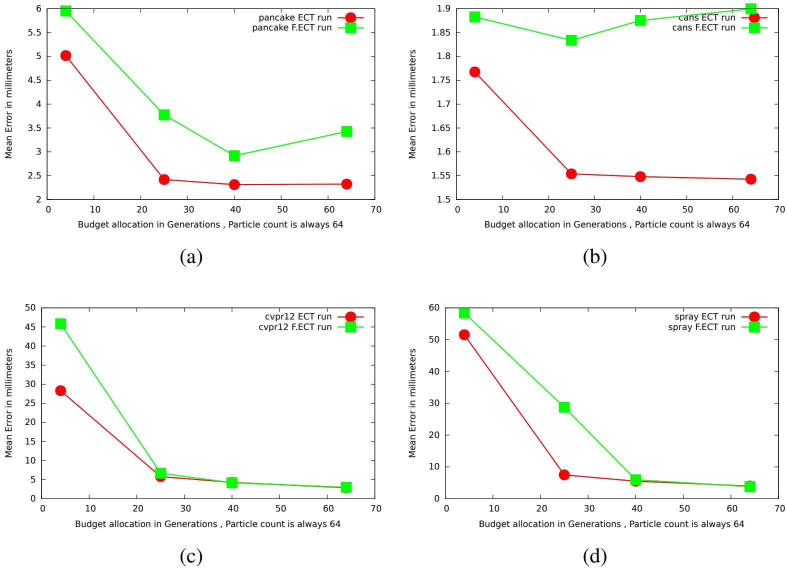
Budget has the trivial minimum value of 0. Moreover, as it has also been shown in [9, 12], a budget of 64 particles running for 64 generations suffices to track a hand, even in interaction with another hand. No more budget should be required to track simpler structures such as rigid objects.

Our dynamic budget allocation policy assigns a minimum budget B_{min} of 64 particles running for 4 generations to the objects that are static and a maximum budget B_{max} that never exceeds the aforementioned budget of 64 particles, 64 generations. thus, the budget $B_i, i = 1, \dots, N$ for all N trackers is defined as

$$B_i = \begin{cases} B_{max}, & \mathfrak{m}_i = 1 \\ B_{min}, & \text{otherwise} \end{cases}. \tag{7}$$

4 Experiments

In order to evaluate the proposed method we conducted several experiments on real and synthetic datasets. The configuration for the method thresholds described in Eq.(6) were empirically identified and remain the same across all experiments (T_c=30, T_d=10.6 mm, T_v=1.5 mm, T_e=7% , t_i=4). All real-life datasets (both standard and home-built) were acquired by PrimeSense RGBD technology, *i.e.* either a Microsoft Kinect or an ASUS Xtion. All ex-



(a)

(b)

(c)

(d)

Figure 3: (a), (b), (c), (d): ECT vs F.ECT with respect to error as a function of number of generations for the Pancake, Cans, Two Hands and Spray datasets, respectively.

periments were run on a machine with the following specifications: CPU Intel i7-4790 CPU Processor @ 3.60 Ghz, GPU NVidia GTX970.

4.1 Experiments on Real Sequences

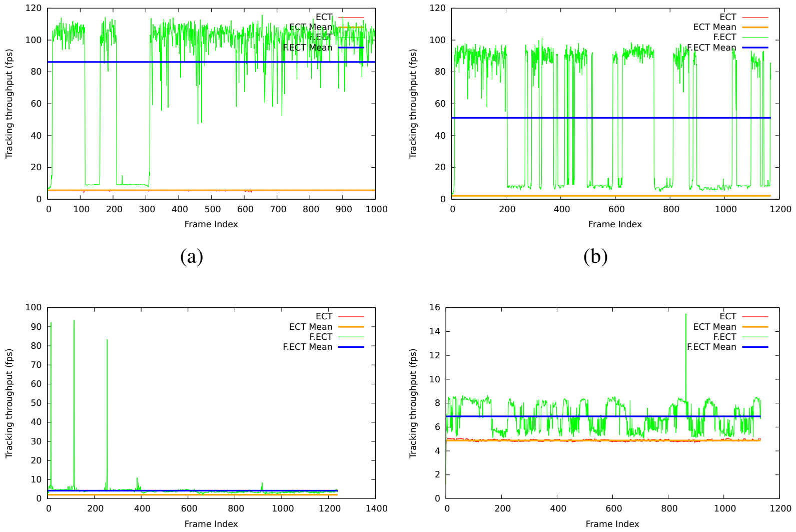
In order to demonstrate the behaviour of our method relative to the nature of the tracked scenes we performed four characteristic real-life experiments. The results are compared, in terms of quality and execution speed, with that of ECT [8].

Two hands dataset [8]: This dataset contains two complex entities (hands) in interaction with fifteen rigid objects. The idle time for objects is average. The tracking of such scenarios yields a small improvement by incorporating the presented method. Indicatively, framerate effectively doubles with a mean framerate of 4.19 (F.ECT) vs 2.06 (ECT).

Spray dataset: In this dataset, a hand manipulates a spray bottle. Thus, the scenario involves very few objects of mixed structural complexity (both rigid and articulated) with almost no idle time. In such scenarios there is a marginal improvement by incorporating the proposed method. Indicatively, framerate only has a marginal 1.3x speedup (6.16 fps for F.ECT vs 4.74 fps for ECT).

Cans dataset: The Cans dataset regards the stacking and unstacking of 9 cans and 4 bottles. This scenario involves a large number of objects of low structural complexity and considerable idle time. Significant improvements can be achieved in such scenarios using the proposed method. We get a 23x speed up (51.13fps for F.ECT) vs 2.19 fps for ECT).

Pancake dataset: This considers a real-life cooking scenario in which a pancake is prepared. This experiment involves a medium number of objects of mixed structural complexity with large idle intervals. In this scenario, mostly due to the idle intervals, we obtain a 15x speed up (86.22 fps for F.ECT vs 5.62fps for ECT).



(a) (b)

(c) (d)

Figure 4: (a), (b), (c), (d): ECT vs F.ECT with respect to framerate for the Pancake, Cans, Two Hands and Spray datasets, respectively.

A video showing ECT vs F.ECT tracking for all the employed datasets is available at http://youtu.be/nPru6PpWrK4.

4.2 Experiments on Synthetic Data

The acquisition of actual ground truth in 3D tracking scenarios of the described complexity in structure and size is hard. In order to assess the accuracy of the proposed method quantitatively, we employed synthetic data. These were constructed by rendering the already tracked real sequences. The initial tracking was computed using ECT running with trackers configured with 64 particles and 64 generations. Then, the solutions were rendered to form a new dataset for which the ground truth would be known.

We performed experiments to assess the accuracy that is achieved by ECT and F.ECT as a function of the actual budget that is assigned to a moving object. More specifically, we considered fixed budgets of 4, 25, 40 and 64 generations, always consisting of 64 particles. Every tracking experiment, *i.e.* every combination of tracking sequence and tracking method, was repeated 5 times in order to marginalize the stochastic effect of PSO in the tracking results. The plots in Fig. 3 show a summary of the experiment results. In all of them, it is clear that the accuracy achieved by F.ECT is comparable to that of ECT. However, as shown in Fig. 4, the computational performance of F.ECT is remarkably better.

Table 1 summarizes the results obtained for the synthetic counterparts of the four experiments (Section 4.1) and shows the framerate and accuracy for ECT and F.ECT. As it can be verified, depending on the characteristics of the scenario, F.ECT achieves a significant speedup which, in certain cases is dramatic. At the same time, the tracking error is only mildly affected.

Dataset	No of Objects	Object Complexity	Idle time	fps (ECT/F.ECT)	Error (ECT/F.ECT)
Two hands	17	Mixed	Medium	2.06 / 4.19	2.99 / 3.07
Spray	2	Mixed	Small	4.74 / 6.16	3.80 / 3.98
Cans	13	Low	Large	2.19 / 51.13	1.55 / 1.92
Pancake	4	Low	Large	5.62 / 86.22	2.30 / 3.10

Table 1: Data sets, their key characteristics and performance indicators (see text for details).

5 Discussion

In this paper, we demonstrated that simple motion cues can be indeed leveraged to improve the computational performance of model-based 3D object tracking. The intuitive idea of focusing computational resources to active (i.e., non static) objects results in dramatic performance gains, at a minor tracking accuracy degradation. This way, tracking performance is not any more a function of the number of the involved entities but rather a function of the complexity of the actual motion and the interaction of objects present in the scene. Therefore, the proposed approach makes it possible to handle accurately and at interactive framerates, scenes and scenarios that the current state of the art can only handle in offline mode.

Current and future work focuses on extending this idea in two different ways. First, the budget allocated to each tracker in the ensemble may become a function of the quantification of its motion as opposed to a fixed budget that is decided based on the binary decision of whether it moves or not. Second, the dynamic and variable allocation of resources can be performed not only on the basis of low level motion, but also on higher level information and cues, including activity interpretation, gaze tracking, etc.

Acknowledgements

This work was partially supported by projects FP7-IP-288533 Robohow and FP7-ICT-2011-9 WEARHAP.

References

[1] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. Motion capture of hands in action using discriminative salient points. In *Computer Vision–ECCV 2012*, pages 640–653. Springer, 2012.

[2] Martin de La Gorce, Nikos Paragios, and David J Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference On*, pages 1–8. IEEE, 2008.

[3] Henning Hamer, Konrad Schindler, Esther Koller-Meier, and Luc Van Gool. Tracking a hand manipulating an object. In *Computer Vision, 2009 IEEE 12th International Conference On*, pages 1475–1482. IEEE, 2009.

[4] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[5] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

[6] Eyal Krupka, Alon Vinnikov, Ben Klein, Aharon Bar Hillel, Daniel Freedman, and Simon Stachniak. Discriminative ferns ensemble for hand pose recognition. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3670–3677. IEEE, 2014.

[7] Nikolaos Kyriazis and Antonis Argyros. Physically plausible 3d scene tracking: The single actor hypothesis. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 9–16. IEEE, 2013.

[8] Nikolaos Kyriazis and Antonis Argyros. Scalable 3d tracking of multiple interacting objects. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3430–3437. IEEE, 2014.

[9] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, volume 1, page 3, 2011.

[10] Iason Oikonomidis, Manolis IA Lourakis, and Antonis A Argyros. Evolutionary quasi-random search for hand articulations tracking. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3422–3429. IEEE, 2014.

[11] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2088–2095. IEEE, 2011.

[12] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Tracking the articulated motion of two strongly interacting hands. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1862–1869. IEEE, 2012.

[13] Karl Pauwels, Leonardo Rubio, and Eduardo Ros. Real-time model-based articulated object pose detection and tracking with variable rigidity constraints. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3994–4001. IEEE, 2014.

[14] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1106–1113. IEEE, 2014.

[15] Javier Romero, Hedvig Kjellström, Carl Henrik Ek, and Danica Kragic. Non-parametric hand pose estimation with object context. *Image and Vision Computing*, 31(8):555–564, 2013.

[16] Mathieu Salzmann and Raquel Urtasun. Physically-based motion models for 3d tracking: A convex formulation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2064–2071. IEEE, 2011.

[17] Björn Stenger, Arasanathan Thayananthan, Philip HS Torr, and Roberto Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1372–1384, 2006.

[18] Erik B Sudderth, Michael I Mandel, William T Freeman, and Alan S Willsky. Visual hand tracking using nonparametric belief propagation. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 189–189. IEEE, 2004.

[19] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3786–3793. IEEE, 2014.

[20] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (TOG)*, 33(5):169, 2014.

[21] Ba-Ngu Vo, Ba-Tuong Vo, Nam-Trung Pham, and David Suter. Joint detection and estimation of multiple objects from image observations. *Signal Processing, IEEE Transactions on*, 58(10):5129–5141, 2010.

[22] Robert Wang, Sylvain Paris, and Jovan Popović. 6d hands: markerless hand-tracking for computer aided design. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 549–558. ACM, 2011.

[23] Chi Xu and Li Cheng. Efficient hand pose estimation from a single depth image. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3456–3462. IEEE, 2013.