2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

# Object Assembly Guidance in Child-Robot Interaction using RGB-D based 3D Tracking

Jack Hadfield[1,2], Petros Koutras[1,2], Niki Efthymiou[1,2],
Gerasimos Potamianos[1,3], Costas S. Tzafestas[2], Petros Maragos[1,2]

***Abstract*— This work examines how and to what benefit an autonomous humanoid robot can supervise a child in an object assembly task. In order to understand the child's actions, a novel 3D object tracking algorithm for RGB-D data is employed. The tracker consists of two stages: the first performs a tracking-by-detection scheme on the color stream, to locate the objects on the image plane, while the second uses a particle filter that operates on the depth data stream to refine the first stage output and infer the objects' rotations. Given the six degrees-of-freedom of the assembly part poses, the system is able to recognize which connections have been completed at any given time. This information is then used to select an appropriate verbal or gestural response for the robot. Experimental results show that (a) the tracking algorithm is accurate, fast and robust to severe occlusions and fast movements, (b) the proposed method of assembly state estimation is indeed effective, and (c) the resulting Child-Robot Interaction scenario is educational and enjoyable for the children involved.**

## I. INTRODUCTION

An interesting application of human-robot interaction can be found in the domain of object assembly tasks. The human agent attempts to build a specific structure out of a number of individual parts, while the robot is charged with the task of assisting the human through a series of instructions, task order suggestions and proposed corrections. This form of guidance may be advantageous over other methods, such as a set of written instructions or a video, because of the feedback provided. Robotic assistance can thus lead to reduced completion times with fewer mistakes.

In the context of child-robot cooperation, the assembly could be a toy, a puzzle, a simple building block structure, etc. In such cases, the presence of a robot can make the task more enjoyable for the child. At the same time, the robot can evaluate a wide range of the child's abilities, including motor skills, hand-eye coordination, spatial ability or short-term memory retention.

Autonomous assembly guidance requires an understanding of the assembly state at any time, so as to provide relevant feedback with as little delay as possible. This understanding is commonly achieved through computer vision techniques, by inferring the positions of all the assembly parts from a given set of images. In this work we focus on visual tracking of the parts as they are moved around in the 3-dimensional workspace, hereby referred to as *3D tracking*. In this way, there is no restriction on the assembly form, meaning that any feasible 3D construction can be tracked.

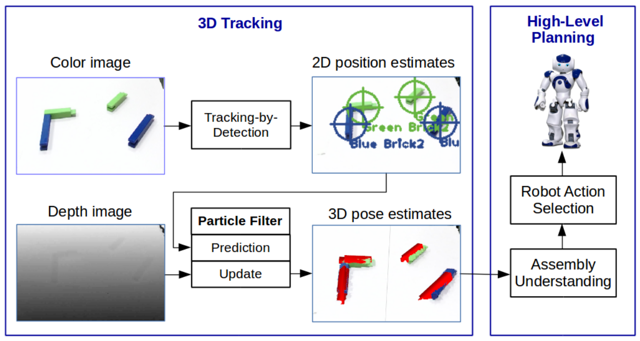

Fig. 1: Overview of the assembly guidance system, with the 3D tracker's basic architecture shown on the left and the high-level planning on the right. A tracking-by-detection algorithm estimates the assembly parts pixel coordinates on the color stream, which are passed to the prediction phase of the particle filter. The final pose estimates are refined based on the depth image. The assembly state is inferred from the 3D poses of its parts and is subsequently used to drive the robot actions.

We present a method of assembly supervision through 3D tracking using a depth sensor. Such sensors provide both color and depth streams (RGB-D) and are generally preferred for 3D tracking over planar color-only cameras, because the additional depth channel contains much richer spatial information. The parts are tracked in real-time using an algorithm consisting of two stages: the first one performs tracking-by-detection on the RGB stream, and the second uses the resulting 2D position estimates in conjunction with the depth stream to infer the six degrees-of-freedom (DoF) of the assembly parts poses. This information is used to identify correct and incorrect connections, and ultimately to control the robot's feedback, in the form of verbal instructions, pointing gestures and head movements. The outline of the overall system can be seen in Fig. 1.

As such, the contributions of this paper are three-fold. Firstly, we propose an efficient and robust approach to tracking multiple interacting objects from an RGB-D stream, which, unlike previously proposed methods, can recover from temporary failures without requiring reinitialization, while at the same time dealing with full occlusions. Secondly, we demonstrate how the developed 3D tracking algorithm can be used to effectively estimate the state of an assembly task in real-time. Finally, we conduct an evaluation with

*This work was supported by the EU Horizon 2020 project Babyrobot, under grant 687831.

[1]Athena Research and Innovation Center, Maroussi 15125, Greece
[2]School of ECE, National Technical Univ. of Athens, 15773 Athens, Greece `jack103@windowslive.com`, `{pkoutras,ktzaf,maragos}@cs.ntua.gr`, `nefthymiou@central.ntua.gr`
[3]Dept. of ECE, University of Thessaly, 38221 Volos, Greece `gpotam@ieee.org`

a number of children, verifying that when using a fully autonomous humanoid robot to supervise the process, based on the estimated assembly state, the task can become more interesting and educational.

This paper is organized as follows. In Section II we examine related works in both assembly guidance and 3D tracking. In Section III we explain the problem of assembly understanding and the notation used in this work. In Section IV we describe the two-stage tracking algorithm that we used. In Section V we demonstrate a use case of the developed system in which a robot interacts with a child, and consequently we evaluate it. Finally, in Section VI we present our conclusions.

## II. RELATED WORK

### A. Assembly Assistance

A number of previous works have studied human-robot collaboration in an assembly setting. While some of these examined how a human could teach a task to a robot [1], [2], [3], we deal with the inverse problem, where the robot is the supervisor. In [4] a robot explained to a child how to construct a puzzle, with the aim of sustaining the child's attention for a longer duration than if an adult was present instead. Mechanisms with which robots can improve performance in collaborative actions were explored in [5], through experimentation on assembly task guidance.

Works on automated assembly guidance that used forms of feedback other than robots, such as augmented reality, are also of interest here. Earlier research in this area relied solely on color image streams, to identify and track various objects related to the assembly. For his PhD thesis [6], Molineros built an automated assembly assistant, which tracked the assembly process after placing fiducial markers on each component. While this facilitates the tracking problem, it is not always practical. In a more recent work [7], the authors proposed a method of assisting the assembly of a planar puzzle built of pentomino pieces. The parts were recognized using a curve matching technique [8].

With the advent and popularization of depth sensors in recent years, certain implemented systems incorporate depth information when supervising the assembly. Under the assumption that the parts shapes will be rectangular, it can be beneficial to consider the virtual workspace as a lattice. In [9], the assembly parts consisted of Duplo® blocks, while [10] dealt with LEGO® bricks. In both cases, the space was voxelized according to the dimensions of the blocks, and the assembly model was thought to reside within this voxelized space. The assembly state was derived by checking which voxels have been filled.

In contrast to such works, we focused on developing a method that is functional for a wide range of part shapes, so long as they are non-deformable. We chose to track the 6-DoF poses of all the parts, and to check the assembly's connections based on the parts' relative positions. A similar idea, albeit for augmented reality, was presented in [11], where the parts were identified using 3D feature descriptors and tracked with the Iterative Closest Point (ICP) algorithm [12]. However, as noted in the article, the system was not robust in occluded environments, and at the time of writing had been tested on assemblies of only two parts. Occlusion handling, in particular, is an essential property for trackers in assembly supervision, because the parts are often covered by one another or by the users hands.

### B. 3D Object Tracking

With respect to model-based 3D tracking, we can generally divide most related work into three categories. The first category treats the problem as one of energy minimization, where the energy function depends on the discrepancy between the observation and the estimate, and optionally on other terms that enforce real-world constraints. The second group includes learning-based methods that employ machine learning techniques to estimate the unknown variables. The third category uses Recursive Bayesian Estimation to infer the objects' poses, given a series of measurements taken from the images. Here, we examine works that use depth or RBG-D data and not those that operate solely on color images.

Belonging to the first group, the ICP algorithm [12], or one of its many variants, can be used to match a mesh model of an object to the point cloud derived from a depth image. In such cases, the energy function is composed of a set of point-to-point or point-to-plane distances. In [13], the error function contains two terms, one related to that of the ICP algorithm and the other to motion constraints. In this way, multiple objects can be tracked concurrently in real-time, while self occlusions and mutual occlusions are successfully handled, but not occlusions induced by hands and other unrelated objects.

In [14], the energy function comprises two terms, with the first referring to the discrepancy between the observed depth image and the image rendered from the state estimate, and the second aimed at penalizing intersections. The minimization is performed by particle swarm optimization. In [15], the energy contains terms with similar objectives, but is formulated using the signed distance function (SDF), thus allowing the problem to be solved by gradient descent. Energy minimization-based algorithms generally suffer from an inability to rediscover objects when lost, as they often get trapped in local minima.

Learning-based 3D tracking is a relatively new approach to the problem. Learning can be carried out on real or rendered depth images, using random forests [16], [17], [18], or deep convolutional neural networks [19]. A major disadvantage of learning-based methods is the fact that they need large amounts of training data. Ground truth for 6-DoF poses can be difficult to obtain, while the alternative option of rendering synthetic images requires careful planning, to ensure that these images match those captured in the real world.

Methods that employ recursive Bayesian estimation usually extract features from the images and feed these features to some version of a Kalman filter. In [20], shape and silhouette features are passed on to an unscented Kalman filter (UKF) to determine an object's 3D pose. In [21], spatially and texturally important features are tracked using an optical flow algorithm and then fed to an extended Kalman filter (EKF). When operating directly on the depth image pixel values, the measurement noise tends to be non-Gaussian, meaning that filters such as the UKF and EKF are unsuitable. To deal with this problem, a robustification method is used in [22] that replaces the actual measurements with pseudo-measurements by applying a suitable feature function. In [23], on the other hand, the inference problem is solved using a particle filter, which due to its non-parametric nature is capable of operating directly on observations corrupted by fat-tailed noise. Our approach is largely based on this work, but (a) allows us to incorporate color information into the process model and (b) better handles multiple objects.

## III. PROBLEM DESCRIPTION AND NOTATION

In order to understand the state of an assembly, it can be helpful to break it down into a series of single connections between parts. A connection is essentially a restriction on the relative movement between its corresponding parts, and can thus be described by a necessary condition imposed upon the parts' relative pose. In the general case, this condition can reduce the degrees of freedom by a number between one and six. In this work we only considered rigid connections, ie. those that reduce the number by six. Therefore, a connection can be defined by a required relative pose between its parts. Thus, if we know the 6-DoF poses of all the parts involved in the assembly, we can easily estimate the state of each connection, and, by extension, of the entire assembly.

Before describing the proposed method of 3D tracking, we explain the notation used in this work. The 3-dimensional vectors $\mathbf{p}^k$ and $\boldsymbol{\theta}^k$ are used to represent the position and orientation (in Euler angle form) of an object $k$ respectively, while $\mathbf{r}^k = (\mathbf{p}^k, \boldsymbol{\theta}^k)$ denotes its 6-DoF pose. With $\mathbf{R}^k$ we refer to the rotation matrix equivalent to $\boldsymbol{\theta}^k$. The vector $\mathbf{v}^k$ denotes the velocity (linear and angular) of the object. The previous variables, when lacking the superscript $k$, denote the concatenation of all objects' respective variables, eg. $\mathbf{p}$ denotes the concatenation of all object positions, $\boldsymbol{\theta}$ all object orientations etc. Regarding the data streams, the color image is denoted by $I_c$, while the depth values are contained in the vector $\mathbf{z}$. The superscript $i$, when used, refers to a single pixel. For all variables, the subscript $t$ refers to the time step.

Let $\mathbf{c} = (c_1, c_2, \ldots, c_{N_c})$ denote a binary vector of length $N_c$, where $N_c$ is the number of connections that make up the assembly, with each element $c_m$ signifying whether a certain connection, indexed by $m$, has been completed or not. For any one of the assembly connections $m$ between two parts $k_{m1}$ and $k_{m2}$, we use $\Delta\mathbf{p}_{m,e} = (\mathbf{R}^{k_{m1}})^T(\mathbf{p}^{k_{m2}} - \mathbf{p}^{k_{m1}})$ and $\Delta\mathbf{p}'_{m,e} = (\mathbf{R}^{k_{m2}})^T(\mathbf{p}^{k_{m1}} - \mathbf{p}^{k_{m2}})$ for the *estimated* relative position of $k_{m2}$ to $k_{m1}$ and of $k_{m1}$ to $k_{m2}$ respectively, while $\Delta\mathbf{p}_{m,d}$ and $\Delta\mathbf{p}'_{m,d}$ denote the corresponding *desired* relative positions.

The probability of a connection being established, given the poses of the assembly parts, is taken to be

$$p(c_m = 1|\mathbf{r}) = \frac{1}{2}d(\Delta\mathbf{p}_{m,d}, \Delta\mathbf{p}_{m,e}) + \frac{1}{2}d(\Delta\mathbf{p}'_{m,d}, \Delta\mathbf{p}'_{m,e}) \tag{1}$$

where

$$d(\mathbf{p}_1, \mathbf{p}_2) = \min\left\{1, \max\left\{0, \frac{\|\mathbf{p}_1 - \mathbf{p}_2\| - \beta)}{\alpha - \beta}\right\}\right\} \tag{2}$$

is a truncated affine transformation of the Euclidean distance between two vectors $\mathbf{p}_1$ and $\mathbf{p}_2$. The parameter $\alpha$ controls the allowed margin of difference between the two vectors, while $\beta$ is the largest distance at which to consider a connection possible. We chose $\alpha = 4$cm and $\beta = 8$cm. Therefore, through (1) we can estimate the assembly state, given the assembly part pose estimates that result from the tracker described in the next section.

## IV. 3D TRACKING

The objective of 3-dimensional visual tracking is to continually estimate the pose of one or more objects in a 3D space, given a stream of images depicting the objects. As the objects' paths are continuous, the previous pose estimates are combined with the latest color and depth images in order to produce the newest estimate, thus differentiating the problem from object *detection*. If we only use the latest estimate, then the tracking problem involves the estimation of

$$E\left[p(\mathbf{r}_t^k|\mathbf{r}_{t-1}, I_{ct}, \mathbf{z}_t)\right] \tag{3}$$

where p(.) denotes a probability density function and E[p(.)] its expected value. In other words, we wish to calculate the expected value of each pose probability distribution, given the previous pose estimates and latest images.

In this work, we split the problem of 3D tracking into two tasks. The first is a 2-dimensional tracking-by-detection problem using solely the RGB stream. This is well-researched and can be performed effectively with little computational cost, using a number of different methods. The second task involves inferring the objects' 3D poses from the depth stream, using a particle filter. This is facilitated by incorporating the 2D tracker output into the algorithm, as explained in section IV-B. The proposed solution results in a two-stage tracker, that is both efficient and robust to rapid movements and occlusions. The idea is that while the second stage is highly accurate, it loses track of the objects when they move too fast. The first stage, on the other hand, is less accurate but able to rediscover objects after temporary failures. Thus, the overall tracking algorithm possesses the benefits of both stages. In the following subsections we describe the two stages in detail.

### A. 1st Stage: RGB-based Tracking-by-Detection

Tracking-by-detection involves detecting some objects in a series of images, based on a set of distinguishing features, and then associating the detections with the objects that are being tracked. In the general case, given a detection algorithm that operates on an RGB image $I_c$, we can produce a probability map $p(\omega(i) = \omega_k|I_c)$ for each object $k$, where $\omega(i)$ is the class of the object visible in pixel $i$ and $\omega_k$ is the $k$-th object class. Thresholding this map gives us a set of candidate regions within which each object might be depicted. Note that the number of detected regions for each object may be larger than one, if there are artifacts of similar appearance in the image, or zero, if the object is occluded or has left the image boundaries.

If the objects are easily distinguishable based on the features used by the detector, then we can simply choose the detected region with the largest area for each object. Otherwise, we must assign the detected regions to the tracked objects. To solve the data association problem in such cases, we opted for an approach based on the Hungarian algorithm [24]. Specifically, for each set of similar objects, we retain the detected regions with the largest area. Specifically, for a set of $N_s$ similar objects, we retain the $N_r$ detected regions with the largest area. If there are enough such regions, then $N_r = N_s$, otherwise all the detected regions are retained ($N_r < N_s$). We then calculate the pairwise distances of each region's center from each object's previous known position. These distances represent the costs of the assignment problem. The Hungarian algorithm matches the regions to the objects in such a way as to minimize the total cost. Finally, a confidence score $s_k$ is produced for each object's estimate:

$$s_k = \frac{A_k}{A_k + \sum_j A'_j} \tag{4}$$

where $A_k$ is the object's respective region area, and $A'_j$ are the areas of the detected regions that don't correspond to any object (background artifacts, noise, etc.).

In the experiments of Section V we used colored bricks, for which a simple color thresholding sufficed. Therefore, the detector simply produced a value of 1 over the pixels whose HSV color values lay within a specified range and 0 otherwise. Note, however, that a wide range of more complex detectors could potentially be used, depending on the difficulty of the scene, provided they are capable of real-time performance.

### *B. 2nd Stage: Depth-based 3D Tracker*

For the depth-based tracking algorithm we chose to follow the method described in [23], because it is efficient, capable of running in real-time, and can deal with occlusions, both partial and complete.

We made a few alterations to the algorithm, in order to better deal with multiple connecting objects. Namely,

- we introduced a collision penalization factor to reduce pose intersections.
- we modified the input variable to allow the exploitation of the color tracker estimates of Section IV-A.
- we imposed connection constraints on the pose estimates when the connections appear likely, to reduce the number of variables that require estimation and to stabilize the output.

The changes are described in detail below.

*1) Bayes model:* As shown in [23], the recursive nature of the tracking problem allows us to use a Bayes filter, whose hidden state corresponds to the object poses and the observed state is the received depth image. Specifically for the hidden state, the vector $\mathbf{x}_t = (\mathbf{r}_t, \mathbf{v}_t)$ is used. The state is augmented with a set of binary variables $o_t^i$ that model the occlusions at each pixel $i$, with $o_t^i = 0$ meaning an object is visible in that pixel and $o_t^i = 1$ denoting an occlusion, and an input variable $\mathbf{u}_t$ which is used to inject prior knowledge into the process model.

The tracking problem is thus formulated as the inference of the probability distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$. This also demands the estimation of $p(o_t^i|\mathbf{x}_t, \mathbf{u}_t, \mathbf{z}_t)$ over all pixels. Due to the non-linear nature of the problem, optimal analytic solutions cannot be found. Instead, a particle filter is used to sample the desired probability distribution. As explained in [23], the particles need only be distributed over the space of $\mathbf{x}_t$, since the occlusion variables $o_t^i$ can be integrated out of the equations.

In this work, we employ the same Bayesian network, with the addition of the binary vector $\mathbf{c}_t$, which represents the apparent connections between objects, as explained in Section III. The independence assumptions between the different variables can be seen in Fig. 2.

*2) Process and observation models:* The process model of [23] is used, which includes a velocity term, a system input, and a noise vector drawn from a Gaussian distribution, to model uncertainty. The occlusion variable transitions are considered to be random events, described by constant probabilities that are set offline.

The original likelihood function $p(\mathbf{z}_t|\mathbf{r}_t, \mathbf{o}_t)$ of [23] compares the measured depth values at each pixel with the expected values produced via rendering techniques. Here, we introduce a penalty factor, which penalizes object collision. When two objects are close to one another, it can be difficult for the tracker to distinguish them. Particularly for objects of similar shape, we observed that the estimated poses often lead to invalid configurations, with one object's representation intersecting another's. To avoid this, we changed the final observation model to $p^*(\mathbf{z}_t|\mathbf{r}_t, \mathbf{o}_t) = \kappa(\mathbf{r}_t)p(\mathbf{z}_t|\mathbf{r}_t, \mathbf{o}_t)$, where $\kappa(\mathbf{r}_t)$ is a factor that penalizes collisions in the pose estimates. We used the simple function

$$\kappa(\mathbf{r}_t) = \begin{cases} 0.01, & \text{if any intersections exist} \\ 1, & \text{otherwise} \end{cases} \tag{5}$$

We found that such a function suffices to reduce the col-

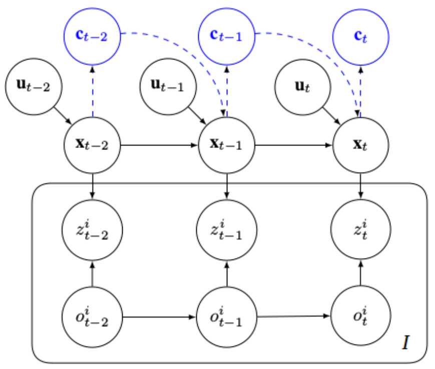

Fig. 2: Independence assumptions made by the filter. The box is a plate which represents all $I$ pixels. Figure modified from [23], with the addition of the connection variables $\mathbf{c}_t$, shown in blue.

lisions between relatively simple shapes. A question for future work is how this penalization could be improved for more complicated object forms, while maintaining a low computational cost.

*3) Input:* In Section IV-A we provided a method of producing crude position estimates on the image plane for each object $k$. Using these estimates, coupled with the depth values at the respective pixels, we can calculate the corresponding 3D coordinates $\mathbf{P}_t^k$ though the camera's inverse perspective mapping. These coordinates are then used to form the object's input vector $\mathbf{u}_t$, as follows

$$\mathbf{u}_t^k = \left(\mathbf{P}_t^k - \mathbf{p}_{t-1}^k\right) \cdot s_k \tag{6}$$

Here $s_k$ is the confidence score of the $k$-th object's 2D position estimate (equation (4)).

Effectively, the input acts as a correcting force on the position estimates, pushing them closer to those provided by the first stage of the tracker. Having more reliable 2D estimates means the control input will have a greater influence on the particles. If an object was undetectable in the first stage, then $s_k = 0$ and the input will have no effect.

*4) Connections:* Since our goal is to recognize connections between the tracked parts, we found it beneficial to enforce connection constraints upon the pose estimates when the respective connections appear likely enough. In doing so, we reduce the number of pose variables that need to be estimated, thus achieving a smoother and more accurate output. The connections that define the assembly task are defined offline and known to the system at runtime. Therefore, we have a finite and rather small set of connections which may be enforced at any time.

In Fig. 2 we introduced the binary vector $\mathbf{c}_t$ which encodes the connections state at time $t$. Specifically, each element $c_{m,t}$ describes whether or not the $m$-th connection has been established. As shown, $\mathbf{c}_t$ depends on the current state estimate and influences the next estimate.

From the previous time step we can compute a set of probabilities $p(c_{m,t}|\mathbf{r}_{t-1})$ from equation (1). Each connection is enforced on a fraction of the total $N$ particle estimates, proportionate to the connection's previous probability. What this means is that the connected objects will be represented as a single object by the chosen particles. In this manner, we approximate the marginalization of the connection variables. We found that using a cap of $\sim 60\%$ of the total particles achieved the desired results.

*5) Algorithm:* Having formulated the process and observation models, the algorithm of [23] is used to produce the object poses. While not stated in the article, the code provided by the authors allows their algorithm to be extended to multiple objects, by dividing the samples into blocks, applying the transition model to each object sequentially and re-sampling for each block, if necessary. In this way, the high-dimensional state can be estimated accurately, despite the limited number of particles.

The parts' initial poses are determined as follows. The tabletop is detected in the depth image by taking the largest smooth area. A least-squares approach is used to find the plane that best fits the points belonging to the detected area. Thus, the point cloud can be filtered to exclude any points below the detected plane. The remaining points are then clustered using the DBSCAN algorithm [25], so each cluster of points belongs to a different part. Using the estimated positions given by the 1st stage of the tracker (Section IV-A), the part representations are positioned in the center of their corresponding point clusters. A set of random rotations is produced, and each rotation is applied to the parts before performing the ICP algorithm to refine the poses. For each object, we choose the best final fit over all rotations in the random set.

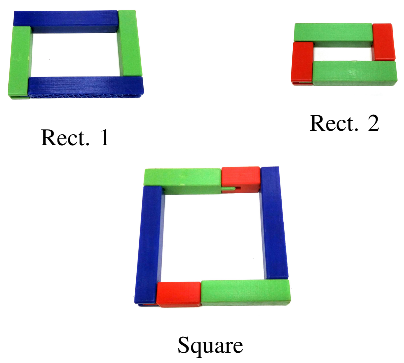

Fig. 3: Three structures used in the assembly experiments.

### C. A Word on Combining Modalities

As explained previously, we chose to combine the depth and color data by infusing the detected pixel estimates into the process model of the particle filter. This idea of treating certain data as system input instead of actual measurements can be found in other works, particularly in those using Inertial Measurement Unit (IMU) data to estimate the state of unmanned vehicles, eg. in [26]. Other options considered here were (i) to incorporate the color-based information in the filter's update stage by changing the observation model accordingly, or (ii) to use two independent trackers whose results are then weakly fused to produce the final estimates.

The problem with (i) is that when an object is lost, the particles will remain centered around an invalid point in the search space and therefore the tracker will not be able to rediscover the object. Method (ii) also suffers from the same issue, while at the same time it ignores the correlation between the two modalities, in effect solving two problems as though they were independent of one another. On the other hand, our approach alleviates the need for re-initialization, and provides an inherent robustness when either one of the two stages fails. Also note that the system's modularity allows us to easily replace one of the two subcomponents with a more complex alternative, if necessary.

## V. EXPERIMENTS AND RESULTS

Having established the core tracking algorithm, we now demonstrate how this can be employed towards our initial goal of robot-to-child assembly guidance. We first illustrate the advantages of our tracking algorithm over other methods proposed in the literature, and then describe the setup used and evaluate the developed system's overall performance.

### A. Tracking Evaluation

In Section II we provided a non-exhaustive list of recent works focused on 3D tracking algorithms. To the best of our knowledge, there has yet to have been an extensive comparative study of such algorithms, making it difficult to identify a single state-of-the-art method. This is further complicated by the fact that each algorithm is designed for a different purpose, depending on the shape, texture and symmetry of the objects, the expected level of occlusion and the processing rate and scalability requirements. Nevertheless, we chose to compare our proposed algorithm against the SDF tracker of [15], due to its accuracy and availability of code. Other methods may produce similar or better results, but we expect most of them to exhibit the same pitfalls as the SDF tracker, which we show below. A more thorough comparison of different trackers across multiple datasets would be an interesting topic for future research.

Since obtaining ground truth for 6-DoF tracking is difficult without the use of intrusive means such as fiducial markers, we conducted a similar experiment to one of those proposed in [15], where the objects are placed in a fixed configuration while the camera is moved around. Two of the bricks shown in Fig. 3, one blue and one green, were placed on a table at a small distance ($\sim$ 10cm) from one another. In order to demonstrate our trackers robustness, we also placed an obstacle on the table, to introduce both partial and full occlusions at times. We also moved the camera at varying speeds, with sudden jolts equivalent to rapid object movements in a real-world setting. If the tracking is accurate, the estimated distance between the two bricks should ideally remain steady for the duration of the experiment.

The results are shown in Fig. 4. We plot the absolute difference between the estimated distance and the ground truth for both trackers, across two different runs. While the SDF tracker is initially more accurate, it fails against rapid movements and full occlusions. The proposed method, on the other hand, retains a relatively low output error under such conditions. Therefore, at the price of slightly reduced tracking accuracy, the algorithm is very robust and is able to recover from temporary failures, without the need for reinitialization techniques. Note that for fairness, both algorithms were run on a single CPU core, and the image data were fed to the trackers in real-time.

### B. Assembly Guidance Setup

The chosen structures consisted of up to six 3D printed bricks which could be connected to one another: two long blue bricks, two medium-length green bricks and two shorter red bricks. The assemblies requested of the child were two rectangular structures and one square, as shown in Fig. 3. For the rectangles, three connections needed to be completed (whereby the fourth connection was established automatically), while the square required five connections.

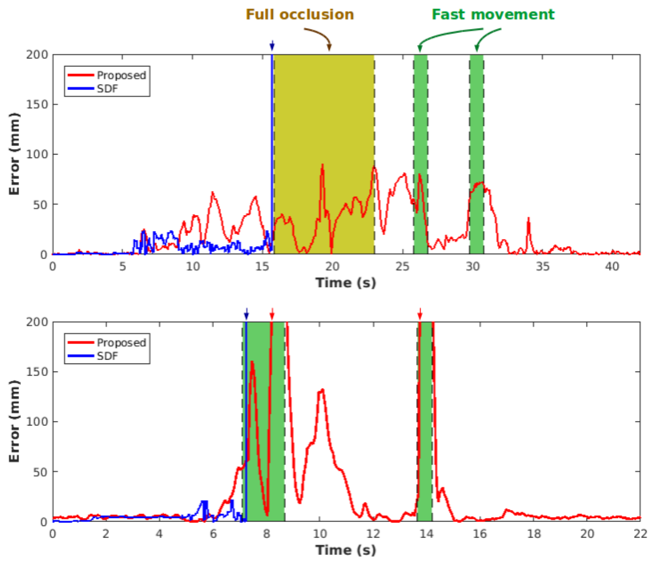

Fig. 4: Comparison of the distance error between two objects across two different runs (top and bottom), estimated by the proposed algorithm (red) and the SDF tracker (blue) of [15]. The yellow region signifies a full occlusion of an object, while the green regions mark camera jolts. The blue and red arrows show the points where the objects were lost. Note that the blue curve remains out of sight from the points where the objects are lost, meaning that the tracker never recovers.

The bricks were placed on a table in front of a Microsoft Kinect device. A Nao robot [27] was positioned to the side of the table, facing the assembly parts. The child sat in front of the table facing the Kinect device, ready to manipulate the bricks as he/she wished. The setup is shown in Fig. 5(a).

The set of desired connections were predefined, with each one described by a necessary relative pose between the two corresponding parts. A discrete number of wrong connections were also considered, such as having the two green bricks together or at a wrong angle. The tracking algorithm described in Section IV was used to determine the brick configuration in each new frame. Equation (1) was used to check which connections had been established. Since the same-colored bricks were identical in form, they could be used interchangeably in the task. For this reason, care was taken to check all feasible combinations for each connection.

Based on the assembly state, the system then chose from a list of responses to aid the child towards their final goal. The chosen response was carried out by the robot, and aside from verbal instructions, also included pointing gestures and head movements, to make the messages clearer. There were three types of responses available: *explanatory*, *congratulatory* and *corrective*. The former were used when introducing the task, when describing each individual step and when the child didn't complete the last given instruction in a given time. The congratulatory messages were given after each correct action, while the corrective ones were given when a mistake was identified.

### C. Experimental procedure

The setup was placed in a Greek primary school, where a total of 21 children, aged 9-10, were invited to interact

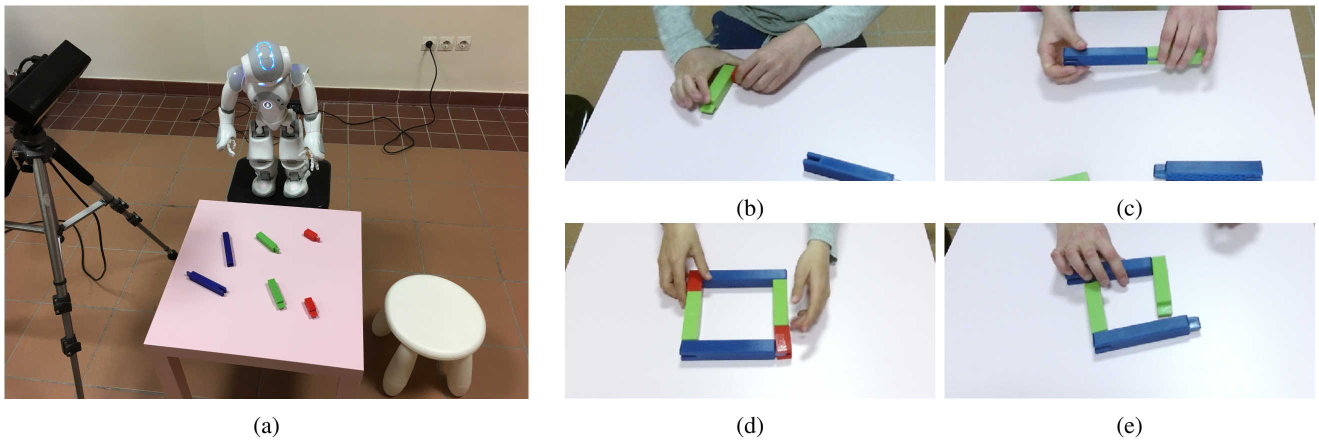
Fig. 5: Setup (a) and four snapshots (b)-(e) taken from the brick assembly experiments, showing (b) a correct connection that was successfully recognized, (c) an incorrect action successfully recognized, (d) a correct action not recognized, and (e) a false alarm. (Child not shown for privacy reasons.)

with the robot. Children of this age group were chosen for experiments, because the Greek school curriculum introduces basic geometric concepts such as right angles and basic shape distinctions at around this age, meaning that the tasks were educational and relatively challenging. Six of the children played with the robot on their own, while the remaining children sat in groups of five. In each trial, the child or group was presented with the setup described above, and was guided by the robot in the creation of one or more of the structures shown in Fig. 3.

## D. Results

In Table I, a number of statistics extracted from the experimental data are presented. The ground truth timestamps when each connection was completed were noted during each trial. Comparing these with the estimated times, we are able to evaluate the system's ability to understand the assembly's state. The first row shows the median difference between the actual connection timestamps and the timestamps estimated by the system, excluding the connections that were never identified. The second and third rows contain the percentage of connections that the system successfully recognized within a time margin of 5s and 20s respectively, including both correct connections that the child completed and mistaken connections. In the fourth and fifth rows the success rate is noted only for the correct connections required to complete the assembly.

| | | Rectangles | Square |
|---|---|---|---|
| (a) | **Median connection time error (s)** | 3.42 | 7.02 |
| | **Total connections identified within 5s (%)** | 50.00 | 43.24 |
| | **Total connections identified within 20s (%)** | 56.25 | 59.46 |
| | **Correct connections identified within 5s (%)** | 70.00 | 39.39 |
| | **Correct connections identified within 20s (%)** | 80.00 | 57.58 |
| (b) | **Average number of mistakes/trial** | 0.86 | 0.67 |
| | **Completion time (s)** | 48.49 | 110.09 |

TABLE I: Statistics related to the System's accuracy (a) and the children's performance (b), for the two rectangles and the square of Fig. 3, averaged over all trials.

| No. | Question | MOP |
|---|---|---|
| 1 | Were you comfortable working with the robot? | 1.81 |
| 2 | Would you play with a robot again, sometime? | 1.81 |
| 3 | Was the robot helpful? | 1.10 |
| 4 | Did the robot make a lot of mistakes? | 0.95 |
| 5 | Did the robot behave like a human? | 1.10 |
| 6 | Were the robot's instructions clear? | 1.95 |
| 7 | Were the tasks easy? | 1.95 |

TABLE II: Questions and results of the questionnaire presented to the children, following their interaction with the robot. The responses were mapped to a scale of 0-2 (0: "no", 1: "quite"/"maybe", 2: "yes"), from which the Mean Opinion Score (MOP) was calculated.

The remaining rows of Table I are related to the children's performance in building the requested structures. The sixth row contains the average number of mistakes the children made per trial, including using the wrong brick and placing the correct brick at a wrong angle. The seventh row shows the average time it took the children to complete each shape.

The results show that the system can correctly estimate the assembly state to a satisfying degree, despite a multitude of difficulties, including severe occlusions, very fast movements, and temporary absences of parts from the field of view of the camera. The rectangles were generally easier for the system to handle, because there were fewer parts that required tracking, but also because the red bricks were a lot harder to track, due to their smaller size. The children made less than one mistake per trial on average, proving that the robot's instructions were helpful and clear. Note that because the children were not familiar with the bricks or the robot at the beginning of the experiments, and as the rectangles were requested before the square, the number of mistakes for the rectangles was higher on average. The completion time is of course higher for the square, though, as it is more complex.

The majority of failure cases were caused when the two stages of the tracker disagreed on the identities of two identical bricks. For example, in Fig. 5(d) the two submodules disagreed on which red brick was which. This led to a large corrective input value for the particle filter, causing erroneous position estimates. This was further complicated by the fact that the children chose the wrong brick at times. Other failures occurred when two bricks were close but not connected, leading to false alarms, as shown in Fig.5(e), and when the bricks were outside the camera's field of view.

Following the experiments, the children were given a questionnaire, as shown in Table II. Each question could be answered "yes", "no" or by some intermediate response ("quite", "maybe" etc.). The answers were mapped to a scale of 0-2, with 2 being the most positive. The mean opinion scores are shown in the third column of Table II. The answers demonstrate that the children viewed the method of robotic supervision very positively, finding the interaction enjoyable and the instructions comprehensible. The fact that they found the tasks easy, while also realizing that the robot made a fair number of mistakes, shows that they were able to distinguish the robot's mistakes from their own, and therefore were not discouraged by the incorrect responses given by the robot. Finally, their willingness to play with a robot again in the future also provides an inspiring takeaway from this work.

## VI. CONCLUSION

In this work we proposed a RGB-D based method of 3D object tracking in an assembly task. The proposed method extends the state-of-the art in two ways: by incorporating both RGB and depth information, and by the ability to robustly track multiple object even in the presence of significant hand occlusions. In addition, a major contribution of the paper is the integration of the tracking system in a Child-Robot Interaction use case, using a Nao robot. The system was evaluated according to the proposed educational scenario, in which 21 children from a primary school were guided to form geometrical shapes. The objective and subjective evaluation confirms the success of our system both in terms of performance and enjoyability. As future work we intend to extend our system to track more complex objects, i.e., toys, and design more demanding and educational tasks for the use of robotic agents as supplementary material in primary schools.

## ACKNOWLEDGMENTS

The authors would like to thank the Nea Genia Ziridi school for their collaboration in the experimental procedure and Georgia Chalvatzaki (NTUA, Greece) for her insightful comments on the overall direction of the work.

## REFERENCES

[1] J. Zhang and A. Knoll, "A two-arm situated artificial communicator for human-robot cooperative assembly," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 4, pp. 651–658, 2003.

[2] C. Burghart, C. Gaertner, and H. Woern, "Cooperative solving of a children's jigsaw puzzle between human and robot: First results," in *Cognitive Robotics: Papers from the AAAI Workshop*, M. Beetz, K. Rajan, M. Thielscher, and R. B. Rusu, Eds. The AAAI Press, 2006, pp. 33–39.

[3] M. Rickert, M. E. Foster, M. Giuliani, T. By, G. Panin, and A. Knoll, "Integrating language, vision and action for human robot dialog systems," in *International Conference on Universal Access in Human-Computer Interaction*. Springer, 2007, pp. 987–995.

[4] M. Fridin and Y. Yaakobi, "Educational robot for children with ADHD/ADD, architecture design," in *International Conference on Computational Vision and Robotics*, Bhubaneswar, India, 2011.

[5] B. Mutlu, A. Terrell, and C.-M. Huang, "Coordination mechanisms in human-robot collaboration," in *Proc. Workshop on Collaborative Manipulation, 8th ACM/IEEE International Conference on Human-Robot Interaction*, 2013, pp. 1–6.

[6] J. M. Molineros, "Computer vision and augmented reality for guiding assembly," Ph.D. dissertation, Dept. CSE, The Pennsylvania State University, 2002.

[7] A. Nishihara and J. Okamoto, "Object recognition in assembly assisted by augmented reality system," in *Proc. SAI Intelligent Systems Conference (IntelliSys)*, 2015, pp. 400–407.

[8] D. Cakmakov and E. Celakoska, "Estimation of curve similarity using turning functions," *International Journal of Applied Mathematics*, vol. 15, pp. 403–416, 2004.

[9] A. Gupta, D. Fox, B. Curless, and M. Cohen, "Duplotrack: a real-time system for authoring and guiding duplo block assembly," in *Proc. 25th Annual ACM Symposium User Interface Software and Technology*, 2012, pp. 389–402.

[10] B. M. Khuong, K. Kiyokawa, A. Miller, J. J. La Viola, T. Mashita, and H. Takemura, "The effectiveness of an AR-based context-aware assembly support system in object assembly," in *Proc. Virtual Reality (VR)*, 2014, pp. 57–62.

[11] R. Radkowski, "Object tracking with a range camera for augmented reality assembly assistance," *Journal of Computing and Information Science in Engineering*, vol. 16, no. 1, pp. 011 004–1–8, 2016.

[12] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[13] K. Pauwels, V. Ivan, E. Ros, and S. Vijayakumar, "Real-time object pose recognition and tracking with an imprecisely calibrated moving RGB-D camera," in *Intelligent Robots and Systems (IROS)*, 2014, pp. 2733–2740.

[14] N. Kyriazis and A. Argyros, "Scalable 3D tracking of multiple interacting objects," in *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3430–3437.

[15] C. Y. Ren, V. Prisacariu, O. Kaehler, I. Reid, and D. Murray, "3D tracking of multiple objects with identical appearance using RGB-D input," in *Proc. 2nd International Conference on 3D Vision (3DV)*, vol. 1, 2014, pp. 47–54.

[16] D. J. Tan and S. Ilic, "Multi-forest tracker: A chameleon in tracking," in *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1202–1209.

[17] D. J. Tan, F. Tombari, S. Ilic, and N. Navab, "A versatile learning-based 3D temporal tracker: Scalable, robust, online." in *Proc. International Conference on Computer Vision (ICCV)*, 2015, pp. 693–701.

[18] S. Akkaladevi, M. Ankerl, C. Heindl, and A. Pichler, "Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5644–5649.

[19] M. Garon and J.-F. Lalonde, "Deep 6-dof tracking," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 11, pp. 2410–2418, 2017.

[20] P. Hebert, N. Hudson, J. Ma, T. Howard, T. Fuchs, M. Bajracharya, and J. Burdick, "Combined shape, appearance and silhouette for simultaneous manipulator and object tracking," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2012, pp. 2405–2412.

[21] O. S. Gedik and A. A. Alatan, "3-D rigid body tracking using vision and depth sensors," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1395–1405, 2013.

[22] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal, "Depth-based object tracking using a robust gaussian filter," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2016, pp. 608–615.

[23] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic object tracking using a range camera," in *Proc. International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 3195–3202.

[24] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, vol. 2, no. 1-2, pp. 83–97, 1955.

[25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96, no. 34, 1996, pp. 226–231.

[26] S. Leutenegger and R. Y. Siegwart, "A low-cost and fail-safe inertial navigation system for airplanes," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2012, pp. 612–618.

[27] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "Mechatronic design of NAO humanoid," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2009, pp. 769–774.