Active Assembly Guidance with Online Video Parsing

Bin Wang[1]* Guofeng Wang[2]* Andrei Sharf[3] Yangyan Li[1] Fan Zhong[1] Xueying Qin[1]†
Daniel CohenOr[4] Baoquan Chen[1]

[1]Shandong University [2]ArcSoft Inc. [3]Ben-Gurion University [4]Tel Aviv University

Abstract

In this paper, we introduce an online video-based system that actively assists users in assembly tasks. The system guides and monitors the assembly process by providing instructions and feedback on possibly erroneous operations, enabling easy and effective guidance in AR/MR applications. The core of our system is an online video-based assembly parsing method that can understand the assembly process, which is known to be extremely hard previously. Our method exploits the availability of the participating parts to significantly alleviate the problem, reducing the recognition task to an identification problem, within a constrained search space. To further constrain the search space, and understand the observed assembly activity, we introduce a tree-based global-inference technique. Our key idea is to incorporate part-interaction rules as powerful constraints which significantly regularize the search space and correctly parse the assembly video at interactive rates. Complex examples demonstrate the effectiveness of our method.

Index Terms: Computing methodologies—Computer graphics—Mixed / augmented reality

1 Introduction

Consumer products now often come in parts, to facilitate their packing and transportation. Consequently, assembly guides are required, which are typically in the form of textual descriptions, figurative illustrations, and how-to videos. These are passive means, in which users parse and follow the instructions but lack feedback on their misinterpretations or misunderstandings of the ongoing assembly process. In this paper, we introduce an *active* assembly-assistance means, i.e., an online video-based system that actively guides and monitors the assembly process. The video-based active guidance system facilitates the user assembly tasks and provides feedback on possibly erroneous operations (see Figure 1).

The analysis of video assemblies is challenging since it requires visual identification of the individual parts, understanding their relationships and tracking assembly configurations. Despite the algorithmic progress in the last two decades, such tasks are still extremely difficult due to ambiguity in the identification and tracking of objects as observed in a video. Our method utilizes the participating parts availability to greatly alleviate the problem, mainly by reducing the recognition task to one of identification within a constrained search space. Nevertheless, the identification of parts and particularly their interactions in an assembly from a video remains an ambiguous and ill-posed problem.

To further constrain the search space, and understand the observed assembly activity, we introduce a tree-based global-inference technique. Our key idea is to incorporate part-interaction rules (PIRs) as powerful constraints which heavily regularize the search space and aid in correctly parsing the assembly video at interactive rates.

*Bin Wang and Guofeng Wang are joint first authors.
†e-mail: qxy@sdu.edu.cn

IEEE Virtual Reality 2018
18-22 March 2018, Reutlingen, Germany

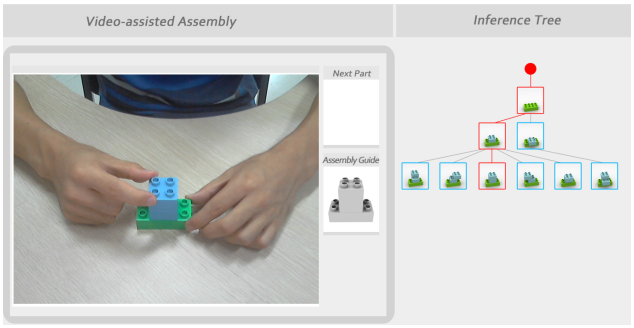


Figure 1: The interface of our system, telling the user the next part to choose and the way to assemble. The inference tree visualizes the parsing process.

The participating parts and their interaction rules are prescribed and represented in a database of 3D shapes and their list of viable interaction rules. A PIR is defined once at a cost that is amortized over the usually massive instances of a manufactured product. It should be stressed that 3D (digital) models of physical objects are now an integral part of the product's design and manufacturing loop and thus are readily available.

We present an inference tree scheme for leveraging the prior knowledge of participating parts and their associated PIRs. The inference tree tracks the assembly process in a stochastic manner and adheres to the correct interpretation in cases of user action ambiguities. Rather than taking a deterministic approach, the inference tree maintains multiple potential interpretations of the assembly configuration at a current observation, and evaluates the likeliness of the different tree paths. As the user proceeds with the assembly process, additional evidence is collected helping to infer the correct interpretation of the assembly state. Thus, the inference tree reflects the system's certainty in the interpretation of the current assembly in the video. As more object parts and assembly actions are introduced, branches in the tree may be pruned if they conflict with the new observations and PIRs (see Figure 2 blue squares). The idea of aforementioned inference scheme is similar to the stochastic activity parsing method introduced in [12], which uses syntactic pattern as constraints for handling ambiguities of temporal events, while our method requires to parse both spatial (part interaction rules) and temporal (order of active parts) configurations.

We demonstrate an active guidance system based on the proposed assembly video parsing method. Our system detects with high probability the current assembly configuration and provides next-step suggestions and error-corrections at interactive-rates. Thanks to the global inference with PIRs as constraints, our system produces faithful instructions with only RGB inputs.

1.1 Related Work

In the following we briefly discuss previous works related to assembly guidance and assembly parsing.

Assembly Guidance Assembly tasks are usually described by

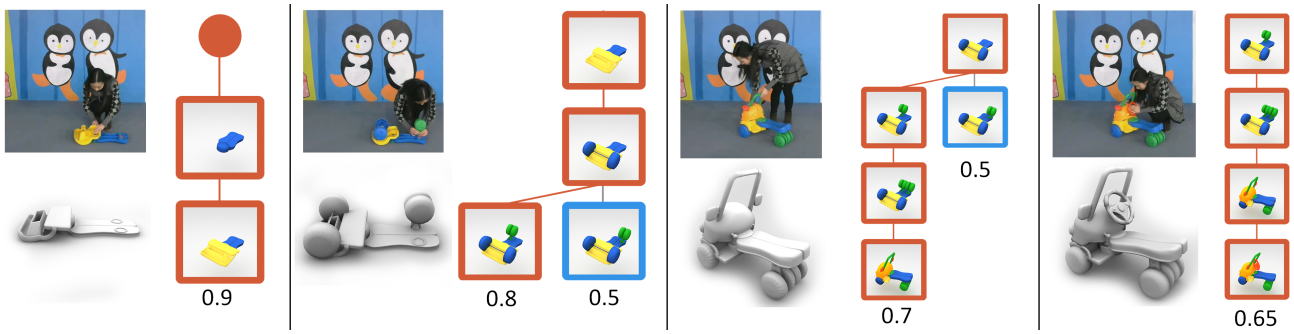


Figure 2: A video-assisted tricycle assembly. Video frames (top left) depict an excerpt of the user actions during an assembly sequence. The system identifies parts, understands their assembly activity and resolves possible ambiguities by employing an inference tree (right). The system monitors and assists the user in the assembly task by generating a 3D model of the assembly sequence (bottom left) .

multiple figures, depicting the current state of the model and where the next part goes. These types of instructions are common for toys (Lego®) and furniture (IKEA®) as well as in "how-to" videos (Youtube®, HowStuffWorks®). We refer to Agrawala et al. [1] for a comprehensive survey on design principles and technical visualization of assemblies.

Tang et al. [24] developed an AR system to aid simple Duplo assembly tasks by displaying 3D object interactions within the user's field of view. Antifakos et al. [3] describe a system which applies sensors to furniture parts and allows the computer to monitor the assembly process and to interactively guide the user. Molineros et al. [19] put markers on parts to track their assembly. They were similar to us, in that they used an assembly graph, (known as liaison graph in robot assembly planning [20]) which represents all feasible assembly operations and reduces the system's degrees of freedom. Nevertheless, these AR systems are mainly concerned with the challenge of displaying the instruction plan to a user. Mohr et al. [17] proposed a system which automatically transfers printed technical documentation to 3D AR. The 3D model is registered to printed documentation with minimal user input, and presents the information from the static documentation in 3D. It analyses assembly information from static documentations, while our system processes assembly videos with PIRs and global inference.

Video tutorials present the user with recorded videos of assembly steps. Pongnumkul et al. [21] present a system that assists the user's work along video tutorials. They detect important events in the video using a simple analysis of Pause-and-Play events and link them to the user application. Jota et al. [13] present StereoBlocks, a system which captures physical objects put together by the user to generate their virtual 3D digital replica. Their system captures simple block assemblies using a Kinect depth camera in a calibrated scene. Similarly, Miller et al. [15] present a system for interactive construction and modification of 3D block models captured by Kinect using a calibrated grid to reduce DOF. Within this line of research, Gupta et al. [8] recently introduced DuploTrack, a system that tracks the assembly process of a snap-together block model using Kinect. The system is able to follow a predefined assembly sequence and provide appropriate feedback. These works share a similar goal of assembly analysis and enhancement. Nevertheless, they are limited to simple Lego-like assemblies or within constrained setups. Due to their widespread use, we focus on casual videos depicting general assemblies.

Assembly Parsing 3D models and their relationships have been explored for object recognition from 2D images since the pioneering work of Brooks [4]. Recent advances in 3D acquisition and the rapid expansion of 3D shape repositories have attracted renewed attention on geometric 3D reasoning for object recognition [11, 26]. In recognition of scenes or complex objects, the spatial layout is often used to regularize the solution [6, 9]. AR applications can benefit a lot from the registered 3D information, for example, Mohr et al. [18] proposed a system to retarget complex video motions into 3D AR tutorials, by registering video object with its 3D model.

Hejrati et al. [10] propose a compositional representation using a small number of local templates for 3D object recognition in images. Desai et al. [7] model human and object interactions using a flexible mixtures-of-parts structure. Their model encodes articulated parts relationships using a flexible spring model. Southey et al. [23] identify the most likely 3D recognition result by accounting for viable 3D spatial relationships. Thus, 3D relationships are utilized to adjust detection confidence scores and to improve the resulting precision. Choi et al. [5] capture 3D spatial relationships using a geometric model for indoor scene understanding. The problem is formulated as an image parsing optimization which yields a parse graph that best explains the image. Similar to these works, we utilize parts interrelations to regularize and reduce the solution space. Nevertheless, our method focuses on assemblies which are typical and therefore can be efficiently governed by a compact set of rules.

CAD models have been extensively explored in the context of assembly analysis [16, 22]. In their work, Mitra et al. [16] infer motions of functional parts from a static assembly by analyzing the geometry and interrelations of parts. Similarly, Shao et al. [22] introduce a part-based assembly analysis which is effective in recovering shape dynamics from sparse 2D sketches. In contrast, our algorithm works the other way: given a video of parts that are assembled, we recognize the individual parts and their interactions. Thus, part recognition turns the video into a parsed dynamic 3D scene, and naturally enables the generation of "how things assemble" visualizations.

2 System Overview

Our principal goal is to understand online and to parse an assembly video. Thus, we may monitor and assist users in their assembly tasks, providing guidance, annotations and a 3D model reconstruction of the sequence.

The input is a raw video footage capturing assembly activities. We assume the participating 3D parts are known a-priori and stored in a part DB, accompanied by a list of simple rules describing viable parts interactions. At the core of our method, is an object-space video analysis algorithm that utilizes the geometry of participating 3D models and their interrelations, defining an inference scheme that helps to interpret the observed activity in the video. The algorithm involves a combination of low-level and high-level video analysis methods based on 3D template fitting for object identification together with 3D object-space assembly processing and inference.

When a new part is presented (denoted *active part*), it is identified by fitting 3D candidates onto the video frames. The identification

of active parts is imperfect. Therefore, each candidate is assigned a confidence value (see Figure 2). For each candidate, we then assess its interaction positions with the current assembled object (denoted *assembly configuration*). Also here, parts target positions and interactions are also ambiguous. By applying part-interaction rules (denoted *PIR*), we reduce the search space and infer with higher confidence the current assembly candidate configurations.

In Section 3, we elaborate on the object-space assembly step, where we apply PIRs from our DB and assemble together 3D object-parts. PIRs prescribe a restricted set of viable orientations and positions which define pairwise interactions between parts. Given a rough guess of the position of the active part in proximity of the assembled object, we explore the PIRs and suggest to potential matching assemblies. In object-space, we then assemble 3D part candidates into a set of valid configurations with respect to the compound 3D object, as suggested by the prescribed rules (see Figures 3 for an example).

The multitude of active part candidates and matching assemblies, yields a set of assembly configurations. We encode these configurations in a tree-like structure (denoted *inference tree*), where different paths correspond to viable configurations. For each configuration, we evaluate its plausibility by registering the respective compound 3D model with the video frames and calculating their confidence scores.

The main challenge stems from the fact that assembly video sequences are typically noisy and may contain substantial occlusions. This results in rather low confidence levels while parsing the video even when parts are correctly identified. Therefore, we cannot trim false positives simply by a confidence threshold. Instead, we make conservative decisions at each step, and keep a set of active part candidates with even low confidence levels. False positives are then pruned by the inference scheme once enough information is accumulated over time. Note that pruning is based on an aggregated assembly confidence rather than making local decisions in each frame independently. By taking the assembly sequence history into account, the robustness of the algorithm increases.

3 Method Details

3.1 DB Creation

We build a database of 3D parts $\{P\}$ which correspond to parts in our video sequences, and a part-rule database $\{R\}$ prescribing the valid inter-relations between parts (Figure 3). These 3D models are now typically an integral part of a product's design and manufacturing loop and readily available. In other cases, we create the corresponding 3D shapes using a modeling tool (e.g. Maya®) or from publicly available repositories. Thus, 3D parts $\{P\}$ serve as the base elements of assembly configurations and may be reused, if applicable, for several videos.

Part-relations are defined as joint position and orientation constraints between 3D parts in space. Note that we do not require the 3D parts to be accurate replicas of the their video counterparts. Instead, 3D parts should merely resemble parts in the videos so that they are discernible and assemble together well.

3D parts in the repository are used to identify *active parts* and estimate their position in the video frames. We associate each 3D part with a set of 2D templates Θ by projecting it onto 2D from 1000 views densely sampled on a 3D sphere centered around it. Each view associates with a pose of six degrees of freedom (DOF) for the 3D part. We also pre-compute the gradient maps G_Θ for the 2D templates using a Sobel filter and keep the top 20 strongest gradient magnitudes in each template. To speed up the template matching, we may store the 2D templates in a two-level hierarchy: a coarse level, using a subset of 200 views uniformly sampling the 3D sphere, and a finer one of 1000 views.

3D parts assemble by connecting together in a common region (denoted *site*). Specifically, for a part P we denote its site S_P (note

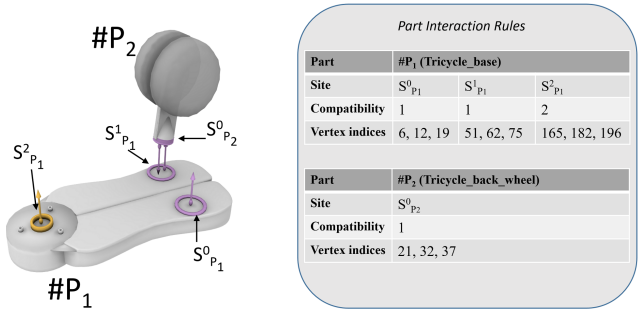


Figure 3: An example of PIRs. Rules define compatible assembly interactions between different sites (interfaces of parts). The same compatibility from different parts can be snapped together. The face vertices determine the transformation between two sites.

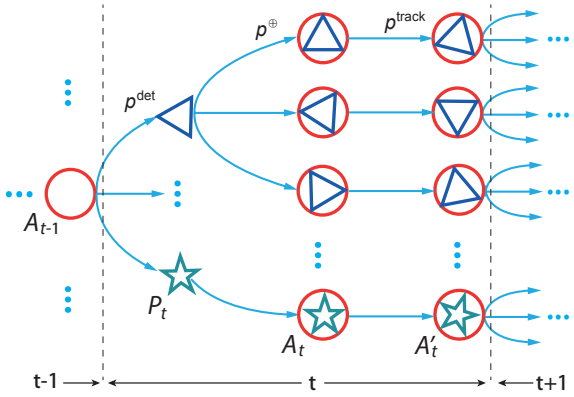


Figure 4: The inference tree. At time t, active part candidates (the triangle and the star) are detected and assembled with the compound model at time $t-1$ (the circle). The new compound model candidates are then reinforced and tracked with video.

that a part may have several sites which can be used to interact with multiple parts), we attribute each site S with a major orientation O_S and an anchor point A_S, for prescribing respectively, the connecting orientation and location of the site in the assembly.

Thus, PIRs define pairs of parts (P_i, P_j) connecting together with respect to specific sites $R_n := (S_{P_i}, S_{P_j})$. To connect two parts w.r.t. their sites, we compute the transformation T_{R_n} that translates and aligns their sites. These rules encode intrinsic relation properties between 3D parts and can now be computed automatically using existing solutions [2, 16, 22]. In our work, we follow commercial guides and extract rules simply by selecting corresponding sites on the 3D objects. In Figure 3, the list was automatically generated from user selections.

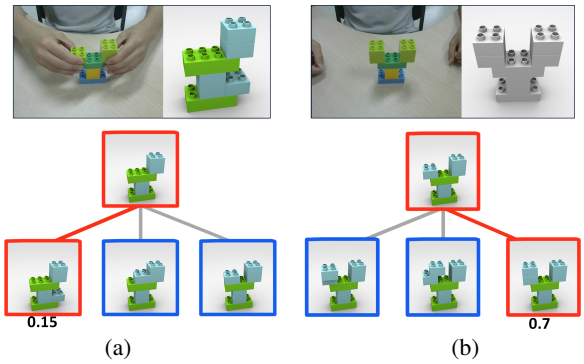
3.2 Online Inference Model

Typically, assemblies are carried out by introducing a part P_t at time t and assembling it with the current model A_{t-1}, resulting in a new assembled model $A_t = A_{t-1} \oplus P_t$. However, accurate recognition of parts and their assembly are error-prone due to occlusions and noise in the input video.

To deal with possible errors, we adopt an online inference strategy that utilizes joint observations from part detection, interaction and tracking in the assembly sequence. Specifically, given a model A_{t-1}, the probability of achieving A_t at time t can be written as:

$$p(A_t|A_{t-1}) = p^{det}(P_t)\, p^{\oplus}(A_{t-1}, P_t|R_n)\, p^{track}(A_t) \qquad (1)$$

where $p^{det}(P)$ measures the detection confidence of a predefined



(a) (b)

Figure 5: Ambiguous assembly configurations. The user's hands occlude the assembled model (a), yielding an incorrect assembly configuration identification in the tree (red branch). The configuration is (typically) of low confidence and pruned out as the following assembly actions disambiguate it (b).

part P, i.e., the probability that P is the *active part* P_t in the input video at time t; $p^{\oplus}$ is the confidence of P_t assembling with A_{t-1} with respect to a predefined rule R_n; p^{track} is the probability of the assembled model A_t in the video, i.e. measuring the detection confidence of A_t in video frame t.

Conditional probabilities in equation (1) can be efficiently represented by an *inference tree* as shown in Figure 4. Nodes in the tree correspond to possible active parts and assembly configurations, while edges correspond to adding a new active part leading from a configuration at level $t-1$ to level t.

By maximizing the conditional probabilities in equation (1), we can infer the model A_t with highest probability. Nevertheless, equation (1) expresses the probability for a single assembly action, generating A_t from A_{t-1} and P_t. This may be too local resulting in an unstable system when video data is ambiguous and noisy.

We extend the probability equation across the whole path of tree in order to avoid local optimal. Thus we solve the configurations $A_t, A_{t-1}, \cdots, A_1$ simultaneously:

$$p(A_t, \cdots, A_1) = \Pi_{i=2}^{t} p(A_i|A_{i-1}) p(A_1) \tag{2}$$

The equation (2) corresponds to a path in the *inference tree* which maximizes the joint probabilities of assembly operations. That means we compute the joint probabilities of whole configuration for probability of current assembled configuration. The advantage of equation (2) is that for a configuration $A_{t-i}, i \geq 1$, its probability is reinforced by considering future observations from assembly steps $t-i+1, \cdots, t$, which is especially helpful for reducing ambiguities as assembly configurations become more complex and discriminative when incoming more parts. For example, in Figure 5, due to user's hand occlusion (a) an erroneous model is inferred. Nevertheless, assembled new parts (b) disambiguate and correct the assembly detection, as an assembly sequence evolves, more parts assemble together reducing the space of viable configurations towards the correct solution.

One big problem of inference is that the tree will grow explosively when more and more active parts come. As active parts appear in the scene, multiple 3D candidates are generated using their probabilities $p^{det}(P_t)$. As parts assemble, we retrieve PIRs with probability $p^{\oplus}(A_{t-1}, P_t|R_n)$ yielding in turn multiple assembly configurations for which we compute the probabilities $p^{track}(A_t)$. The *inference tree* may grow in a combinatorial way in time if the generated assembly hypotheses are not pruned, as shown in Figure 4. Such global computation is unnecessary because of the typical locality of the assembly process. Thus, we need to prune the tree in order to make the computation feasible. In our experiments, we observe that keeping top 5 pathes in *inference tree* achieves a good balance between efficiency and stability.

The prune strategy is straightforward. Once the assembly tracking is finished, we get probability of each term of equation (1), the paths with low confidence are pruned. Note that we do not simply throw out active part candidates with low confidence, as they may be misidentified due to poor active part detection. Instead we consider the whole information of path, if the joint probability of a path is low, we prune it. In this way, it will reduce the ambiguity of active part candidates. In Figure 2, assembled parts which lower the confidence of an incorrect configuration (in blue square) will be pruned.

3.3 Assembly Video Processing

Our system performs as a three-state machine: 1. *active part identification*, 2. *PIR-based assembly* and 3. *assembled object tracking*. When an active part appears in the scene, our system performs *active part identification*. Once an active part is identified and enters the proximity range of assembly configuration, the state machine changes to *PIR-based assembly*, and retrieves PIRs and yields multiple 3D assembly candidates. After that, our system will go into *assembled object tracking* and compute the probability of each 3D assembly candidates. Note that in each state, one of the probabilities in equation (1) is generated. These probabilities are added into the inference tree and paths with low probability are pruned. Thus a compact inference tree is maintained, enabling online running. Then our system returns into *active part identification* for continue.

3.3.1 Active part identification

Our input consists of assembly videos captured by ordinary webcams. The videos may be noisy, captured from a non-static camera, consisting of light variations, occlusions, and camera distortion, etc. In the low-level video analysis step, we coarsely identify and estimate the 6DOF pose of active parts as they appear in the video. For efficiency reasons, we avoid the problem of tracking active parts and instead identify them individually as they appear in the video.

Given a video frame I, we identify active parts by matching 2D templates from our DB. We use a real-time DOT operator (Line2D) [11] to detect if an object represented by a set of 2D templates Θ is in I with 3 levels image pyramids. Note that we skip all pixel locations of the assembly to avoid existing parts in the assembly distorting the active part identification.

We compute the gradient map for I denoted G_I using the Sobel filter and select the $l = 20$ strongest gradient magnitude representatives in each template. We then convolute the gradient map with our pre-computed 2D templates gradients G_Θ of a 3D part in the DB and measure the gradient response using DOT:

$$\Phi(I, \Theta, c) = \text{dot}(G_I, G_\Theta, c)$$

where c is the pixel in I, and dot(.) is the DOT detector measuring the response of I to the template Θ at point c. The specific definition of dot(.) is as equation (2) in [11].

A 2D template is detected in the image at c' if the highest gradient response is above a threshold $\varepsilon = 0.8 \cdot l$:

$$c' = \arg\max_c \{\Phi(I, \Theta, c) > \varepsilon\}$$

Thus, to detect if a 3D part from our DB is in frame I, we match its 2D templates $\Theta_0, \Theta_1, \ldots$ to I utilizing the coarse-to-fine hierarchy and check if any of them gives a gradient response above $\varepsilon = 0.8 \cdot l$.

Typically, a frame I may yield several active parts candidates due to occlusions and ambiguity in the part identification. We consider the DOT gradient response as the identification probability per active part candidate:

$$p^{det}(P_t) \propto \Phi(I, \Theta_{P_t}, c') \tag{3}$$

Once an object is identified in a frame, we refine its 3D pose in object space using its best matching template. Specifically, we

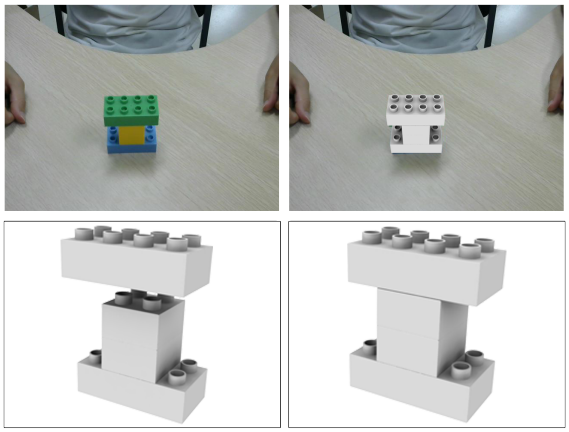


Figure 6: PIR-based snapping. An assembly video (top row), may yield an ambiguous 3D model configuration (bottom row). PIRs regularize the 3D search space of active parts and resolve ambiguities (bottom right) .

can infer the orientation and depth z from the 2D template using its corresponding view direction and scale respectively. The translation in x and y can then be computed by:

$$x = \frac{z}{f_x}(u - c_x), y = \frac{z}{f_y}(v - c_y),$$

where u and v are the pixel horizontal and vertical coordinates, f_x and f_y are the focal lengths of the camera, and c_x and c_y are the center of the camera.

3.3.2 PIR-based guidance

As a new active part P_t approaches the current assembly configuration A_{t-1}, we search the rules DB for part relations that match the current configuration and will guide the assembly process. We apply these rules to the 3D part in object space, and transform it onto the assembled model yielding a new compound 3D model.

The advantages of applying PIRs are twofold. First, rules regularize the 3D pose of active parts by defining a canonical transformation which connects the part to the assembly. These rules regularize the assembly space to be discrete (Figure 6).

Second, by connecting the 3D active part with the assembled model, we can re-evaluate the compound object as a whole and reinforce or prune past decisions. Thus, a poorly detected active part (e.g. due to occlusions) may generate an assembled object of high confidence, while a well detected active part may lead to a low confidence assembly (e.g. due to incorrect PIR selection in Figure 5).

Thus, for an active part candidate P_t, we first check whether any of its assembly sites $\{S_{P_t}\}$ are in the proximity of any of the parts' sites in the current assembled object $\{S_{P_a}\}$: $\|S_{P_t} - S_{P_a}\|_2 < r$, with $P_a \in A_{t-1}$. The distance is measured between the parts' bounding boxes in image space (we use $r = 5$ pixels in all of our experiments). For each pair of parts with sites within proximity distance, we check if their assembly is viable by querying the PIR DB for a rule defining the interaction between S_{P_t} and S_{P_a}. Given such a rule $R_n(S_{P_t}, S_{P_a})$, we apply the corresponding transformation $T_{R_n}(P_t)$ which aligns the active part site S_{P_t} to S_{P_a} in the assembly, and generates a new 3D compound model A_t. Once the active part assembles to the current assembly, the connecting sites will be disabled from connecting to parts in the future.

The probability $p^{\oplus}(A_{t-1}, P_t | R_n)$ is computed with respect to the site distance and the required transformation $T_{R_n}(P_t)$:

$$p^{\oplus}(A_{t-1}, P_t | R_n) = \exp(-\beta \|S_{P_t} - S_{P_a}\|_2) p(T_{R_n}) \quad (4)$$

with $p(T_{R_n})$ measuring the consistency of T_{R_n} w.r.t. the estimated 3D pose of P_t from part detection, β is a balancing parameter which is set as 0.1. We may ignore the a-priori probability of $p(R_n)$ although for specific assembly problems it may reflect the expected order of assembly actions in a sequence and can be trained to reflect their probability.

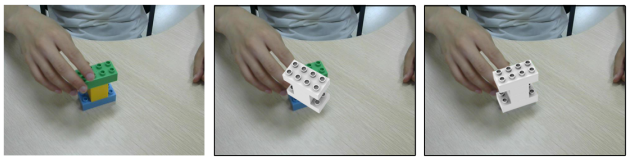


Figure 7: Assembly tracking. A partially occluded assembly (left) leads to incorrect tracking (mid). Taking an hybrid tracking-detection approach, we recover the correct assembly pose (right) by registering the 3D model with the frames.

3.3.3 Assembly tracking

For each new 3D compound model candidate A_t, we compute its probability similar to the active part identification step. Specifically, we project the 3D model onto 2D and compute its gradient response using a DOT operator with the video frame. Note that since we detect and track the compound object's pose in the video, we need to project the 3D model onto 2D from a single view. The score measures the confidence of the 3D assembly configuration with respect to the current video frame:

$$p^{track}(A_t) \propto \Phi(I, A_t^{prj}, c) \quad (5)$$

with A_t^{prj} the projection of A_t on the image, c is the best matching location between A_t^{prj} and I.

As the user interacts and assembles parts, the assembled compound object may be moved and rotated in the scene. Therefore, we take a hybrid tracking-detection approach to locate and track the assembled object in the scene, so that the assembled object keeps registered with video. At the same time $p^{track}(A_t)$ is updated by leveraging observations from different views.

Given a valid assembly configuration, we adopt the technique of [25] to register the 3D assembly with video, and to track the changes when the user interacts with the assembly. [25] solves the optimal 3D object pose by minimizing misalignment between image edges and projected model contours, it is fast and suitable for textureless objects. Nevertheless, this technique is insufficient when object motion is too large or the object has heavier occlusion (see Figure 7).

Therefore, we combine tracking with 2D template detection in a conservative manner. Utilizing the DOT operator, we can infer the 3D pose of the assembled object by registering its corresponding template with the frame. Thus, we generate in realtime 2D templates by projecting the assembled 3D model into 2D using its current pose. We perform such template generations every constant number of frames ($K = 100$). Thus, once tracking of the assembly is lost, we utilize the most current 2D template for identification and re-localization of the assembly object and resume the tracking process.

When performing tracking, $p^{track}(A_t)$ is updated by selecting the maximum over time: $p^{track}(A_t') = \max\{p^{track}(A_t)|t\}$, which prefers the assembly that can get a good match with the video in different views. This is helpful to reduce ambiguous configurations that cannot be identified in specific views.

Table 1: Experiment summary

scene name	no. of parts	duration	no. of PIRs	POV
tricycle (Fig. 2)	7	118 sec	5	1st person
lego (Fig. 5)	13	127 sec	1	1st person
chair (Fig. 10)	6	60 sec	2	1st person
house (Fig. 11)	14	150 sec	11	1st person
robot (Fig. 12)	7	135 sec	5	1st person
wood train (Fig. 8)	5	100 sec	4	3rd person
kitchen (Fig. 12)	6	216 sec	4	3rd person

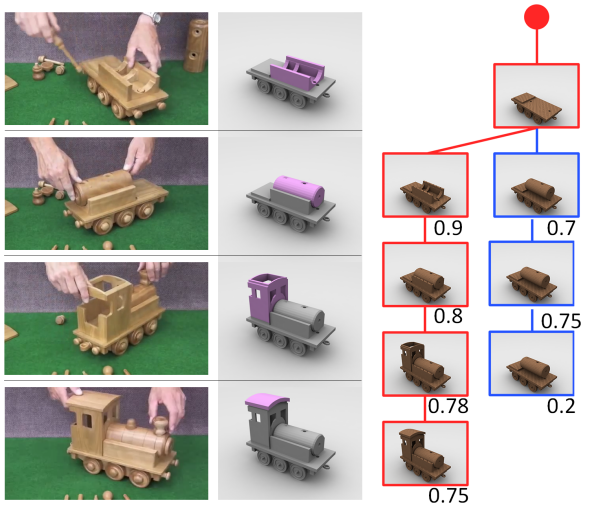


Figure 8: Assembly analysis of a raw Youtube video of a wood toy assembly (left col.) The identified assembly sequence and active parts (in red) are showed in 3D (mid column). Inference tree (right) showing an invalid configuration (blue) is pruned out since assembly rules do not further apply to it.

4 Results and Discussion

We evaluate our system on a variety of assembly videos, with objects from different classes and with different complexity and detail. We capture assembly videos in an unconstrained setup, using an ordinary webcam from either a first- or third-person POV. For first-person POV, the camera is mounted on the forehead of assembly operator, thus the videos are captured under dynamic viewpoints. The parsing results under dynamic first-person POV are provided in the supplementary video. Table 1 summarizes the videos: number of parts, number of PIRs, camera setup and sequence duration.

Figure 2 depicts the analysis results of our system on a tricycle assembly sequence. As parts are introduced in the scene, they trigger our part detection method. As they are attached to the current object, we search the DB for corresponding interaction rules and generate an inference tree. Video frames correspond to detected assembly actions. Due to uncertainty in the video data (e.g. due to occlusions), ambiguity occurs in detecting the tricycle front wheel positions (mid-left). We store the two possible configurations as paths in the tree with different probabilities. Additional assembly actions (mid-right) help disambiguate this case, pruning out the blue path and selecting the red path as the highest confidence configuration (right).

In Figure 8, we process a raw Youtube®video footage of a wooden train assembly process. Since we do not have the physical 3D measurements of the parts, all 3D parts were coarsely approximated. Nevertheless, our algorithm still achieved plausible reconstruction. Note that the ambiguity in the tree is pruned out as parts assemble together and the confidence of the incorrect configuration (blue) finally decreases as assembly rules cannot apply.

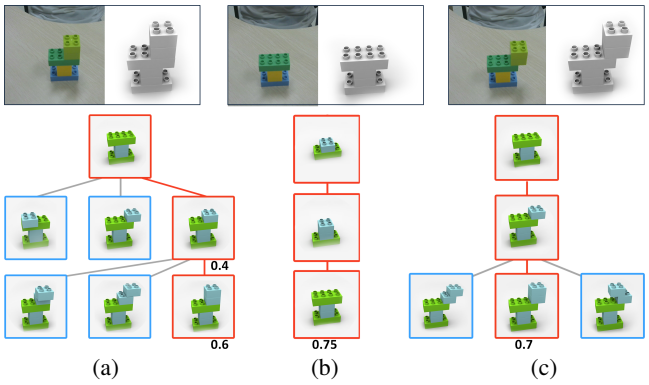


Figure 9: User feedback for assembly actions. Assembly actions identified by our system, may be incorrect w.r.t. the target model (a). Once an incorrect configuration is reliably identified, the system backtracks to a valid configuration (b) and the user is notified to correct the assembly (c).

Our video analysis is independent of the assembly order. While rules merely define pairwise relations, our framework tries to infer part information from a global assembly configuration and use all the information available from other parts in the assembly. In this way, rules progressively constrain the search space, as assembled parts prune down the number of candidates. See the confidence values of tree paths in Figure 5.

4.1 Assembly Guidance

For monitoring and assisting the user in the assembly task, our system tracks the user's assembly actions from a video camera and provides online feedbacks with assembly correction and next-best-part suggestion. For this purpose, in addition to the part-interaction rules, we store in our DB *assembly paths*. Assembly paths are easily achieved by defining PIRs sequences which can be represented by lists or tree like structures. This means that PIRs are additionally indexed to encode their order of application in the assembly.

Assembly correction. As the user performs invalid assembly actions, an appropriate feedback method is issued and the system backtracks to the last valid configuration. For example, in Figure 9, the user positions 2 blocks incorrectly, before the system reliably detects an erroneous configuration (left) and backtracks to the last valid configuration (middle). Therefore, feedback is provided only after the configuration is reliably detected.

In Figure 10, we analyze a relatively simple assembly of a children's chair consisting of a seat, four legs, and a back part. However, this example is still quite challenging as parts are of the same color and the background consists of a significant colored texture which introduces significant noise in the recognition and tracking. Furthermore, the chair legs' are perfectly symmetrical which introduces further ambiguities in the inference tree. The tree stores several leg configurations (possibly symmetric) as candidates (see Figure 10(b,d)) and the best configuration pose is selected. Note that the last level in Figure 10 seems as different viewpoints of exactly the same assembly, they are actually different assembly configurations with the chair back in different sides of the chair base.

Similarly, Figure 11 provides excerpts from a Lego®assembly sequence where the user positions a roof side on the incorrect side (a) and a roof side is misaligned (c). Our system identifies these incorrect configurations (see red paths in corresponding trees) and prompts the user to backtrack and re-assembly again resulting in the correct assembly sequence (in b and d).

Next-best-part suggestion. In Figure 12, we show excerpts of a guided assembly of a complex technical Lego® robot and a kitchen

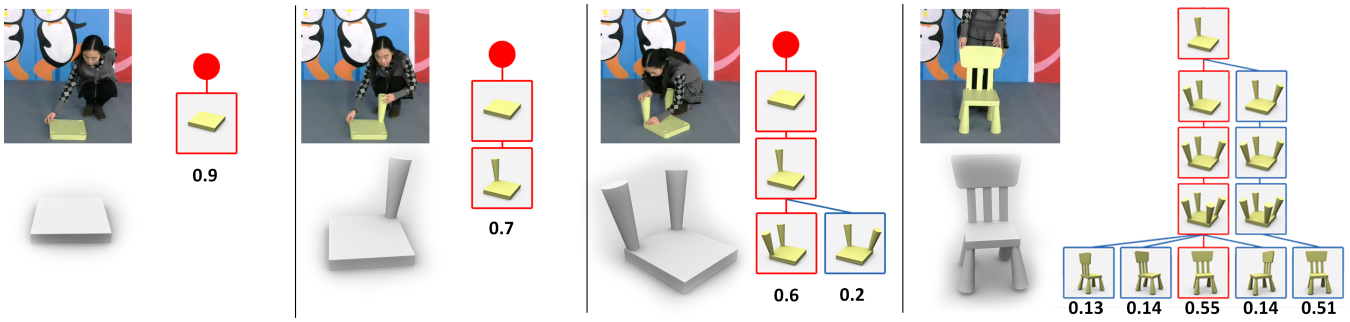


Figure 10: Excerpts from a chair assembly video sequence. As active parts approach the assembled object, part interaction rules infer viable configurations, yielding a tree of assembly hypotheses (on the right). Ambiguous user actions (c), are corrected and pruned by additional assembly actions (d). The correct 3D sequence of the assembly is generated (bottom) to guide the user assembly tasks.

toy. Our system infers the correct assembly actions and allows to the reconstruction of a 3D model counterpart. Thus, the user may inspect the intermediate objects from an infinite number of views and zoom. The system suggests at each step, a possible next part (although typically there may be several). As the user introduces the new active part, the system provides immediate guidance regarding its assembly in the current 3D object. User operations are validated against the predefined set of PIRs and feedback is provided.

4.2 Performance and Scalibility

We evaluate the performance of our system on a PC with a 3.2GHz Intel i5-3470 CPU and 8GB RAM. The frames used for evaluation are of 640×480px resolution, larger resolution is not necessary for processing. Our implementation runs at approximately 10-30fps depending on the number of different parts and the number of PIRs. For a typical case of 15 parts and 10 PIRs, active part identification requires 0.06s, PIR-based guidance requires 0.01s, and assembly tracking 0.02s. Currently we did not use multi-thread and GPU acceleration, with better implementation and hardware, the performance still can be greatly improved. The performance bottleneck of our system is the active part detection, which scales linearly with the number of templates. For the cases with more parts, a more scalable part detection method should be adopted. Fortunately, recent advances in 3D detection provides with potential scalable approaches [14], and our system can exploit these approaches by simply replacing the part detection module.

4.3 More Applications

The proposed assembly parsing method can be used for additional applications. Note that by registering image objects with 3D models, our method actually has reconstructed the assembly process in 3D space, so any 3D operation is enabled. As an example, we demonstrate an application with automatic generation of annotated 3D animations [1]. We utilize the 3D parts, their transformations and assembly snapping transformations to generate novel animations. This is extremely useful if alternate views are required to assist the assembly process. Thus, a user may zoom and further investigate an assembly process, interactively edit it by replacing parts, editing sizes and etc.

5 Conclusions and Limitations

We have presented a method for video analysis which allows the reconstruction of an assembly process and the dynamics of the 3D model that is assembled. The key idea is a tree-based inference framework that links the low-level video analysis with higher level inter-part relational rules. This constrains and guides the system towards the correct detection and registration solution. Instead of making decisions based on individual parts, the assemblies encode inter-part relations and enables the posteriori probabilities of assembly sequences. The decisions are then re-enforced by re-projecting an assembly of 3D parts into the video frame images and their re-evaluation in image space.

[1] Please see the supplemental video for results

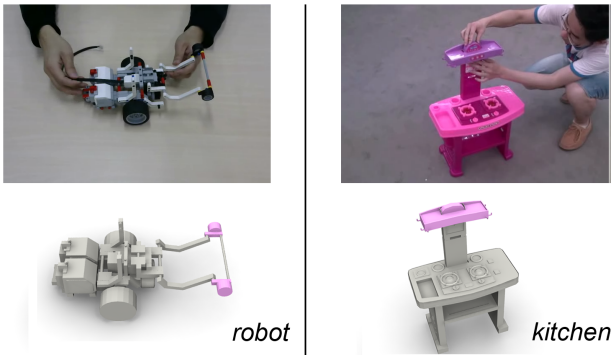


Figure 12: Complex assembly monitoring and guidance. Our system analyzes online the assembly process of a piece of complex technical machinery (top row) and suggests to the user the next part (in red, bottom row) and its position in the assembled object.

Limitations.

Our system just receives one active part at each assembly stage, thus multiple active parts cannot be assembled together at the same time. Our algorithm cannot differentiate subtle differences in video frames. The assembly rules may be able to narrow down some ambiguities, but since the rules do not determine a unique assembly operation, the algorithm can still get confused especially when the participating parts are symmetric or nearly symmetric (see Figure 10). We also cannot handle assembly rules which contain some degree of freedom that span continuous space. It will be very interesting to study how to compute an optimal identification in cases where the assembly rules do not define a discrete space. Finally, some assemblies require meticulous and complex actions, e.g. pushing, pulling and stretching and simultaneous actions. This poses an interesting question regarding their PIR representation and detection of such actions. These issues will be considered in our future works.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments. This work is supported by NSF of China (No. 61572290, No. 61672326), the National Key Research and Development Pro-

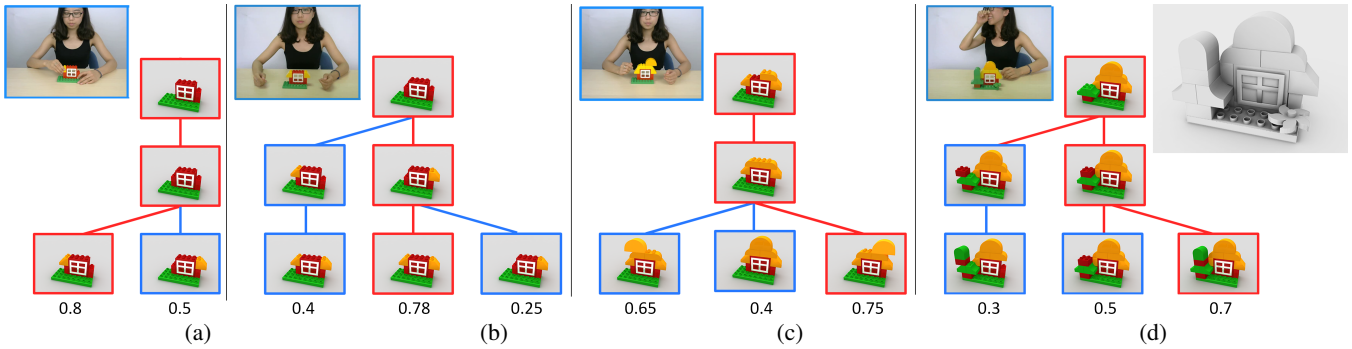


Figure 11: Assembly guidance and feedback, with user errors corrected utilizing the online inference. In (a) roof part is assembled on the incorrect house side and in (c) the house top is misaligned.

gram of China (No. 2016YFB1001501), the Fundamental Research Funds of Shandong University (No. 2015JC051) and the Joint NSFC-ISF Research Program (No. 61561146397) jointly funded by the National Natural Science Foundation of China and the Israel Science Foundation.

REFERENCES

[1] M. Agrawala, W. Li, and F. Berthouzoz. Design principles for visual communication. *Commun. ACM*, pp. 60–69, 2011.

[2] M. Agrawala, D. Phan, J. Heiser, J. Haymaker, J. Klingner, P. Hanrahan, and B. Tversky. Designing effective step-by-step assembly instructions. In *ACM SIGGRAPH 2003 Papers*, pp. 828–837, 2003.

[3] S. Antifakos, F. Michahelles, and B. Schiele. Proactive instructions for furniture assembly. In *Proceedings of the 4th International Conference on Ubiquitous Computing*, pp. 351–360, 2002.

[4] R. A. Brooks. Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence*, 17(1-3):285–348, 1981.

[5] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 33–40, 2013.

[6] R. G. Cinbis and S. Sclaroff. Contextual object detection using set-based classification. In *Proceedings of the 12th European conference on Computer Vision - Volume Part VI*, ECCV'12, pp. 43–57. Springer-Verlag, Berlin, Heidelberg, 2012.

[7] C. Desai and D. Ramanan. Detecting actions, poses, and objects with relational phraselets. In *Proceedings of the 12th European conference on Computer Vision - Volume Part IV*, ECCV'12, pp. 158–172. Springer-Verlag, Berlin, Heidelberg, 2012.

[8] A. Gupta, D. Fox, B. Curless, and M. Cohen. Duplotrack: a real-time system for authoring and guiding duplo block assembly. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, pp. 389–402, 2012.

[9] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1849–1856, 2009.

[10] M. Hejrati and D. Ramanan. Analyzing 3d objects in cluttered images. In *Advances in Neural Information Processing Systems*, pp. 602–610, 2012.

[11] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):876–888, 2012.

[12] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):852–872, 2000.

[13] R. Jota and H. Benko. Constructing virtual 3d models with physical building blocks. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, pp. 2173–2178, 2011.

[14] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit. Hashmod: A hashing method for scalable 3d object detection. 2015.

[15] A. Miller, B. White, E. Charbonneau, Z. Kanzler, and J. J. LaViola Jr. Interactive 3d model acquisition and tracking of building block structures. *IEEE Transactions on Visualization and Computer Graphics*, pp. 651–659, 2012.

[16] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, and M. Agrawala. Illustrating how mechanical assemblies work. *ACM Trans. Graph.*, 29(4):58:1–58:12, July 2010.

[17] P. Mohr, B. Kerbl, M. Donoser, D. Schmalstieg, and D. Kalkofen. Retargeting technical documentation to augmented reality. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 3337–3346, 2015.

[18] P. Mohr, D. Mandl, M. Tatzgern, E. Veas, D. Schmalstieg, and D. Kalkofen. Retargeting video tutorials showing tools with surface contact to augmented reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 6547–6558, 2017.

[19] J. M. Molineros. *Computer Vision and Augmented Reality for Guiding Assembly*. PhD thesis, 2002.

[20] H. Mosemann and F. M. Wahl. Automatic decomposition of planned assembly sequences into skill primitives. *IEEE Transactions on Robotics*, pp. 709–718, 2001.

[21] S. Pongnumkul, M. Dontcheva, W. Li, J. Wang, L. Bourdev, S. Avidan, and M. F. Cohen. Pause-and-play: Automatically linking screencast video tutorials with applications. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pp. 135–144, 2011.

[22] T. Shao, W. Li, K. Zhou, W. Xu, B. Guo, and N. J. Mitra. Interpreting concept sketches. *ACM Trans. Graph.*, 32(4):56:1–56:10, July 2013.

[23] T. Southey and J. J. Little. 3d spatial relationships for improving object detection. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 140–147, 2013.

[24] A. Tang, C. Owen, F. Biocca, and W. Mou. Comparative effectiveness of augmented reality in object assembly. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pp. 73–80, 2003.

[25] G. Wang, B. Wang, F. Zhong, X. Qin, and B. Chen. Global optimal searching for textureless 3d object tracking. *The Visual Computer*, 31:979–988, 2015.

[26] M. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3d representations for object recognition and modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2608–2623, 2013.