Learning a High-Precision Robotic Assembly Task Using Pose Estimation from Simulated Depth Images

Y. Litvak, A. Biess*, A. Bar-Hillel*

***Abstract*— Most of industrial robotic assembly tasks today require fixed initial conditions for successful assembly. These constraints induce high production costs and low adaptability to new tasks. In this work we aim towards flexible and adaptable robotic assembly by using 3D CAD models for all parts to be assembled. We focus on a generic assembly task - the *Siemens Innovation Challenge* - in which a robot needs to assemble a gear-like mechanism with high precision into an operating system. To obtain the millimeter-accuracy required for this task and industrial settings alike, we use a depth camera mounted near the robot's end-effector. We present a high-accuracy three-stage pose estimation pipeline based on deep convolutional neural networks, which includes detection, pose estimation, refinement, and handling of near- and full symmetries of parts. The networks are trained on simulated depth images by means to ensure successful transfer to the real robot. We obtain an average pose estimation error of 2.14 millimeters and 1.09 degree leading to 88.6% success rate for robotic assembly of randomly distributed parts. To the best of our knowledge, this is the first time that the *Siemens Innovation Challenge* is fully solved, opening up new possibilities for automated industrial assembly.**

I. Introduction

Robots in manufacturing and assembly tasks are often lacking perception, making it necessary to place the parts to be manipulated in pre-defined positions using fixtures or part feeders. The installation of such structured environments is a time-consuming process, which induces high costs. To enable the next advance in robotic manufacturing, where a large variety of products is likely to be manufactured in small production volumes (Industry 4.0), robots need to be endowed with adaptive and flexible skills. Flexibility here means the ability to work with *arbitrary* initial part conditions. We strive to obtain flexible assembly based on recent advance in machine learning, computer vision and depth sensing.

Many studies involving manipulation of parts are proposing methods for grasping from arbitrary initial conditions [1],[2],[3],[4],[5],[6],[7],[8],[9]. Some consider kitting and basic assembly tasks [2],[10],[11], like stacking of Lego bricks [11] or peg-in-the-hole problems [10]. Most of the grasping studies consider grasping of *general* objects, which is considerably more difficult than grasping of a known specific part. In addition, most of these studies consider RGB images obtained with a camera in a fixed position covering the entire scene, and some of them train mappings directly from images to actions. This creates highly diverse image distribution, leading to a general and difficult learning problems. The cost of learning in such a large input space is often reduced accuracy (the ability to bring a part exactly to a target pose) and reduced robustness (the probability to accomplish the task successfully). When all the parts of an assembly task are rigid and known a-priory as CAD models, a different approach can be used, which we pursue here. First, the task can be completely defined in terms of start and end target poses of the parts. The target poses are naturally defined as relative part poses (pose of one part in the reference frame of the other), but can trivially be translated into the robot reference frame once the poses of all parts are estimated. In addition, the end effector pose required for a successful grasp of each part can be stated a-priory (in the part's reference frame), as part of the task definition. This implies that an assembly task can be primarily reduced to pose estimation, provided that motion control is sufficiently accurate. While this view is somewhat naïve (as it assumes perfectly accurate motion control) it is still useful, since errors arising from inaccuracies in pose estimation are often much larger than those arising from motor noise.

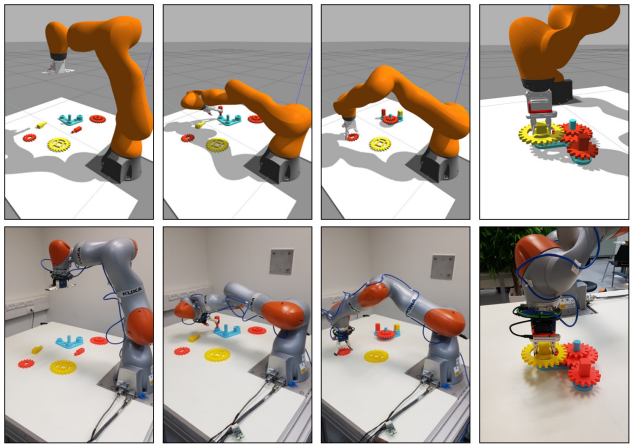

Figure 1: The KUKA LBR IIWA robot performs the *Siemens Innovation Challenge* successfully in simulation (top row) and in reality (bottom row).

An appealing approach for improving pose estimation is by making the learning problem easier: use a small input space, and a large training sample. A large sample is difficult to obtain on a real robot [3], but can be easily generated in a simulated environment [4],[5],[6],[7],[10],[12]. When a pose estimation algorithm is trained in a simulator, adapting it for execution on the real robot may be challenging, especially for RGB images and wide scene view. In recent years techniques for bridging the simulation-reality gap in RGB images were proposed like domain randomization [4],[5] or images

* These authors contributed equally, same affiliation for all authors.
Research supported by the ABC Robotics Initiative, Helmsley Charity Foundation.
Y.L. is with the Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Be'er Sheva 8410501, Israel. (corresponding author, phone: +972-52-523-4050 email: litvaky@post.bgu.ac.il)
A.B. (email: abiess@bgu.ac.il), A.B-H. (email: barhille@bgu.ac.il)

synthesis with GANs [4]. However, transfer is far from perfect, and bridging the simulation–reality gap is simpler for synthetic depth images, which are more similar to their real counterparts obtained by a depth camera [7]. These considerations led us to the following working assumptions in our approach: First, it is assumed that exact 3D CAD models of all rigid parts of interests are available. Second, we rely exclusively on depth sensing. Third, the depth sensor is mounted close to the robot's end-effector and pose estimation involves placing the effector in a canonical pose near the part of interest. These factors allow for extensive training in a simulator environment, enable smoother adaptation to the real robot and lead to higher accuracy due to the small distance between camera and manipulated parts. In addition, since we assume having a CAD model of all parts, and all training occurs with simulated data, a policy may be trained for a new task in several days of computation time without training on the real robot. Hence, the approach – if successful – has the potential to be adaptive by enabling fast migration to new tasks.

In this study we test the suggested approach on a single assembly task and try to solve the *Siemens Innovation Challenge*. In the challenge a robot needs to assemble a gear-like system composed of a base plate, two different shafts and three different gears into an operating mechanism (Fig. 1). The task can be broken into 5 sub-tasks, each requiring grasping and assembly of a single part. The sub-tasks are highly challenging since they require millimeter accuracy in positioning of the parts. The initial poses of the base plate and the parts to be assembled are not constrained and arbitrarily placed without overlap in the workspace of the robot. In addition, some parts have rotational symmetries, or near-rotational symmetries, of different orders. The *base plate* has no symmetry (or symmetry of order $n=1$). *Gear 2* has symmetries of order $n=4$ (i.e., rotations of 360/4 degrees lead to the same shape) and the *compound gear* has symmetry of order $n=12$. *Gear 1, shaft 1 and shaft 2* have near-symmetries, meaning rotations of the part resemble each other, but are not identical (Fig. 2 and 3).

For a rigid part, the pose in 3D is fully defined by 6 parameters (three translational and three rotational degrees of freedom). However, for many robotic applications, and specifically for the task we consider, it is sufficient to describe the pose of a part in the plane, which requires only three parameters (two translational and one rotational degree of freedom [13],[14]). The main component in our approach is the pose estimation pipeline, composed of three stages implemented as Convolutional Neural Networks (CNN). In stage one, several images covering the scene are taken from a pre-defined height above the workspace surface. These are processed by a RetinaNet [15] based detector and pose estimator, providing detection and initial pose estimation for each part. In stage two, each part is cropped from the image, virtually rotated to a canonical pose and processed with a second network to obtain orientation refinements and near-symmetry classification. In stage 3, the robot's end-effector is moved to take a close-up view image of each part separately. The image is taken in canonical pose based on the current pose estimation and is subjected to further pose refinement by a third network. Stages one and two are executed once to provide pose estimations for all parts, followed by stage three at the beginning of assembly of each part.

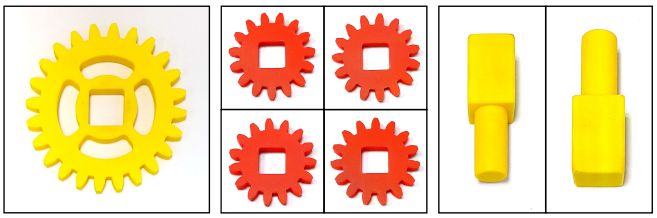

Figure 2: Isometry and near isometry of real assembly parts: (Left): *Gear 2* object with symmetry every 90°. (Middle): near-symmetry views of the *Gear 1* with 4 near-symmetries rotated to canonical pose. Note that while these four rotations of *Gear 1* are very similar, successful classification among them is required for exact pose estimation. When another rotation is used instead of the correct one, the grasp efficiency will be reduced due to lack of contact points and the gear's teeth may not fit well into the teeth of its neighboring gears. (Right): two near-symmetries of *Shaft 2* rotated to canonical pose, which are much easier to discriminate.

Our environment consisted of a KUKA LBR IIWA 14 R820 anthropomorphic robotic manipulator equipped with a SAKE ROBOTICS EZGripper and an Intel RealSense D415 depth camera mounted on the gripper (Fig. 1). For each part, except of the *base plate*, the task definition includes two relative positions: a final required pose relative to the *base plate*, and a grasp pose (of the end effector) defined in the part's reference frame. A script encoding the assembly sequence was written, including a grasping and assembly motion for each part with trajectories computed using standard motion planning. In addition, a force-based feedback routine is employed upon each assembly part. Part misalignments are detected using a threshold on the force sensor reading and assembly is re-attempted in multiple offsets around the expected position.

Our experiments included 30 assembly attempts on the real robot. In these experiments part detection rate was 100%, near isometries were resolved in 100%, the mean translation error of a part was 2.14mm, and the mean rotational deviation was 1.09 degrees. This accuracy enabled successful grasp in 96.6% of the cases, and successful assembly in 88.6% of the cases. While direct comparison is not possible, the obtained accuracy and the scope of successful assembly is higher than previous related work [5],[10]. Most of the failures are related to the grasp instability due to a sub-optimal gripper: the fingers cannot reach a parallel grip in certain cases and the force applied by the gripper is not strong enough. We hence believe the suggested approach is a significant step toward flexible and adaptive CAD-based robotic assembly.

II. Related Work

We briefly discuss topics which are mostly related to our work: flexible assembly and part manipulation, pose estimation, and exporting policies from simulation to reality.

Flexible Assembly: A lot of recent work has focused on grasping from arbitrary initial conditions ([1],[3],[4],[5],[6],[7],[8],[9], are a small subset), but fewer have considered manipulation and assembly tasks [2],[10],[11]. The latter tasks are harder as they involve both grasping and bringing certain parts to tight spatial relations with others. In [2] several manipulation tasks were learnt on the real robot, like placing a coat hanger on a rack or screwing a cap on a bottle. The learning burden is split between two agents: a mixture of linear

Gaussian controllers, each trained for specific known initial conditions and a deep network generalizing across initial conditions. The linear Gaussian controllers are simple to learn as they get the state as input, and they are used to teach the network using a BADMM formulation, with local optima guaranties. In [10] the tasks are most similar to ours, as they use a subset of the *Siemens Innovation Challenge* tasks (two sub tasks), as well as a U-shape fitting task. In this work the pose estimation problem is faced using fiducials (sticking markers on the parts designed for pose estimation). The focus is on enabling assembly actions with complex non-linear trajectories. Training is done using a combination of motion planning and Guided Policy Search (GPS), where the former is used to guide training of the latter. High success rates are reported for the real robot, but only from two known and fixed initial part positions. In [11] a difficult assembly task of Lego brick stitching is handled, but only in a simulation environment. The authors assume known states, including brick positions, and use a DDPG algorithm with several improvements to learn assembly policies.

Approach [2] and [10] as stated above are facing the difficulty arising from the unknown state (pose of the parts) by solving a difficult learning problem of complex observation-to–action maps. Our approach is inherently different: We avoid the need for learning observation-to-action maps by enabling highly accurate state (pose) estimation, from which simple motion planning suffices for task accomplishment. We obtain this accuracy using two main ideas: first, we narrow the distribution to face a relatively easy learning problem by moving the sensor (on the gripper) to the vicinity of the part with a relatively fixed spatial relation and use of near-range depth imaging. Second, we solve this problem in the simulator in the endless sample size limit, enabling a complex pipeline of multiple refining CNNs.

Pose estimation: Pose estimation of an object starts with object detection, for which significant progress has been made in recent years [15],[16],[17]. In the recent vision literature, pose estimation is often performed for non-rigid objects like the whole human body [18] or the human hand [19]. In these cases, the pose is specified using the location of multiple interest points. In contrast, the pose of a rigid object in 3D is fully defined by six parameters, or three if only position on the plane is considered. In the Cornell grasping data set [13] the grasping task is defined using such pose triplets specifying the pose of the gripper (not the object) required for a successful grasp. Several network architectures were suggested for the regression of successful grasp poses, such as, a two-stage pipeline of candidates finding followed by candidate ranking in [13] and regression in multiple spatial cells in [14]. Other CNN architectures for pose estimation in the robotic context include networks with spatial soft-max for feature point finding [3] or convolutional pose machines [20], where multiple layers regress the same heat map for refinement. A possible approach to reduce the complexity of pose estimation is achieved by using depth information ([7],[12]). Specifically [7] uses a depth camera mounted near the gripper as is done in our work. In this study we combine the strengths of [7] with an iterative process of canonization and refinement, similar to [19].

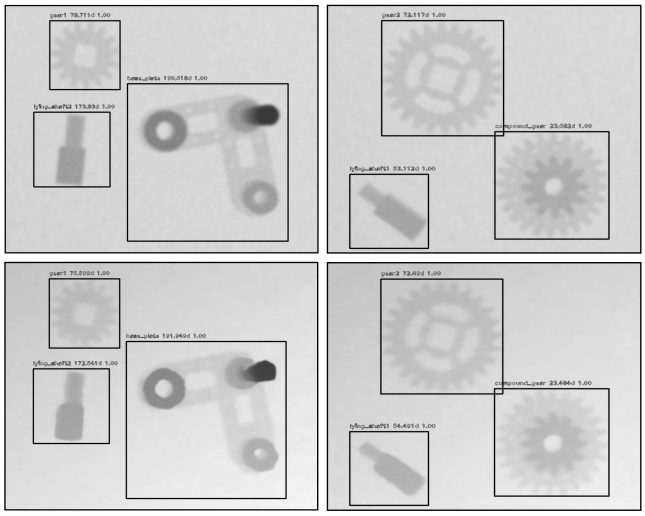

Figure 3: Evaluated depth images, showing all 6 objects, taken from distance of 0.53m in simulation (top row) and in real (bottom row). Parts' classes by order of appearance (left column and then right column, from top object and clockwise): *gear 1, base plate, shaft 2, gear 2, compound gear and shaft 1*.

Simulation to reality transfer: When a predictor is defined over RGB images, transferring between simulated and real images is challenging due to significant differences between simulated and real images in terms of illumination, texture, and background distribution. In domain randomization, [4], [5], scenes are generated under a large variety of visual conditions (e.g. illumination conditions and background textures), object types and robot dynamics characteristics. A predictor trained on such a rich distribution is usually more amenable for transfer to the real world. Other domain adaptation techniques are domain–adversarial neural networks (DANN [21]) and a transfer GAN creating seemingly-real images from simulated ones [5],[22]. When relying solely on depth imaging for perception, the gap between simulation and reality is smaller and easier to bridge [12],[23] – a strategy that we follow in this work. Specifically in [12], a 6D grasp pose detector has been trained in a simulator and transferred successfully to the real robot with 93% success rate without any special mechanisms used for transfer. In [23] it was shown that by enhancing simulated data to mimic the signal processing pipeline of a depth camera transfer can be further facilitated. We follow some of these ideas in this work.

III. METHOD

A. Pose Estimation – General Overview

Our pose estimation pipeline includes three stages: (1) detection and coarse pose estimation, (2) pose refinement in canonical pose and classification of near-symmetries and (3) high-precision pose estimation from close-up views.

Stage 1- Detection and coarse pose estimation: Several depth images covering the entire surface are taken from a distance of 0.53m above the table. These images are analyzed for parts and poses using a fully convolutional neural network (CNN) with architecture adapted from the RetinaNet [15]. For each detected part the network outputs the part's class (one of six known classes), a bounding box around the part, and the orientation angle θ of the part around the Z axis of the world frame. The real position of the part's center on the workspace surface is then calculated by de-projecting the center pixel of

the estimated bounding box. To simplify learning, the range of the orientation angle θ for each part is defined by the symmetry of the part spanning an interval from 0° to $2\pi/n$, where n is the order of the rotational symmetry. For symmetric parts, *compound gear* and *gear 2*, the maximum angles are 30° ($n=12$) and 90° ($n=4$) respectively. For near-symmetric parts, *gear* 1, *shaft 1* and *shaft 2*, the maximum angles are 90°, 180° and 180° respectively.

Stage 2 - Refined orientation estimation in canonical pose and classification of near-symmetries: Each part detected at stage 1 is cropped from the original image according to its estimated bounding box and rotated into a canonical pose by the estimated (negative) orientation angle θ. The network in stage 2 receives as input the canonical image as well as the predicted class of the part from stage 1 and computes a correction for the orientation angle θ. In addition, the network specifies the sub-class for near-symmetric parts in terms of its orientation (Figure 2). For shafts, there are two possible sub-classes: the part is close to 0° (class 0) or close to 180° (class 1). Respectively, for *gear 1* four sub-classes are possible.

Stage 3 - High-precision pose estimation from close-up views: To obtain high-precision pose estimation results, we analyzed images from close-up views. For this purpose an additional depth image of each part is taken from a distance of 0.31m (the closest distance for which the largest part can be fully seen) and rotated into canonical pose according to the estimations obtained from stages 1 and 2. The output of stage 3 is a bounding box with a high-precision estimate of the center position in the world frame (same as in stage 1), a fine-tuned correction for the orientation angle θ and a classification of the sub-classes for the near-symmetric parts (same as in stage 2).

B. *Pose Estimation – Networks Architectures*

Stage 1: All our networks are similar to the RetinaNet architecture [15] (Fig. 4). The networks' backbone in stage 1 is ResNet-50 [24], a residual network pre-trained on ImageNet dataset, followed by a Feature Pyramid Network (FPN) [25], which is effective in detection of multi-scaled parts. The FPN extracts features from three internal layers of ResNet-50 to construct a pyramid of five layers, *P3-P7*. These layers contain similar representations, summarizing features from high and low layers of the ResNet backbone, but in five different spatial resolutions. Even though the images in our environment are taken from fixed heights, RetinaNet's backbone is used without modification. Anchors for object detection were used as in the original network with five octaves and nine anchors at each point, representing different scales and aspect ratios. During training anchored are labeled as positive, negative or 'ignore' based on the intersection-over-union (IOU) between the anchor and ground truth bounding boxes, with original

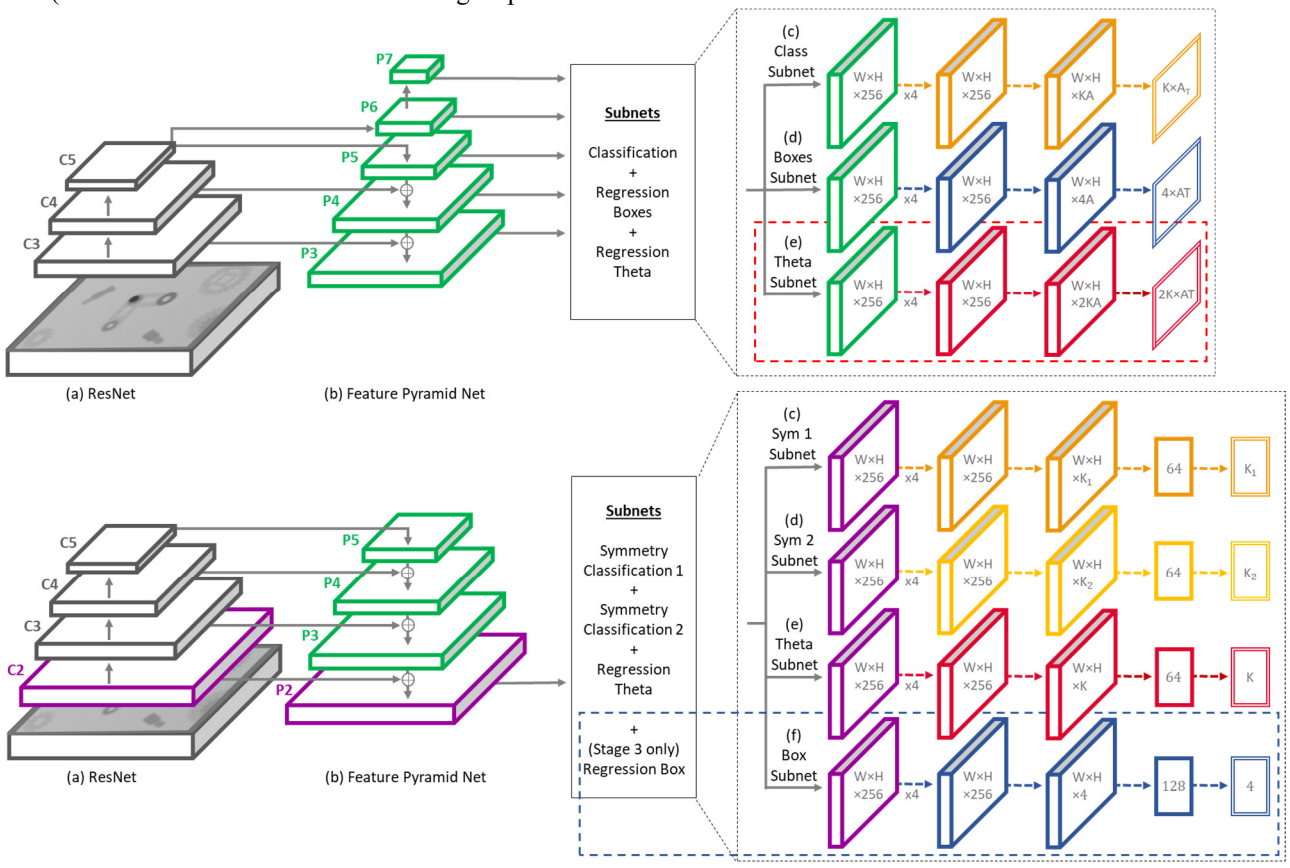

Figure 4: Networks' architecture. (Top graph): Stage 1 network, based on RetinaNet architecture, starting with a backbone of FPN on top of a feedforward ResNet (a-b), followed by three subnetworks. First two subnetworks are classification (c) and regression of bounding boxes (d) and the third and new subnetwork, marked with a red square, is regression of rotation angle θ (e). In this drawing $A = 9$ is the number of anchors per location, A_T is the total number of anchors in the image and $K = 6$ the number of part classes. (Bottom graph): Stage 2 and 3 networks. We use four internal layers from ResNet (a) to produce a high-resolution layer from the FPN (b), followed by 3 subnetworks in stage 2 (c-e) and a fourth subnetwork, marked with a blue square, in stage 3 (f). K_1 is the number of near-symmetries classes learned in Symmetry 1 subnetwork (c), K_2 is the same for (d), and K is the total number of classes ($K_1 + K_2$ + all other objects which don't have near-symmetries). The last layer in each subnetwork is the output layer.

thresholds set to 0.5 and 0.4 for positive and negative overlap, respectively. Since we aim towards more accurate detection, we set the thresholds to 0.7 and 0.6, respectively.

RetinaNet includes two sub-networks: one for classification and one for regression of the bounding boxes. The losses for the first two sub-networks were used without change: Focal loss [15] for part classification, and Huber loss (smooth L_1) for bounding box regression with parametrization of [20]. A third sub-network for regression of the orientation angle θ was added in stage 1. The third subnet of orientation regression outputs for each [anchor, part] the orientation θ parametrized as $(\sin(n\theta), \cos(n\theta))$, where n is the symmetry order of the part (following [8]). Since for most part-anchor pairs no part is detected, no loss is applied in such locations during training – the loss only applies in true detection anchors and only for the relevant part. We use the Huber loss over the two relevant orientation neurons for each class.

Stage 2: No detection is needed in stage 2 and 3, since input images with a single part are provided to the network, so the anchoring system and the focal loss are not in use. We nevertheless kept the same backbone, but instead of creating five pyramid layers, *P3-P7* we use one layer with higher resolution. We used another internal ResNet layer, *C2*, and created a new pyramid layer, *P2*, that combines the information from all the layers above it along with *C2*. *P2* is the input to the subnetworks. Stage 2 contains three subnetworks. The first two classify parts with near-symmetries. Subnet 1 predicts the four possible orientations of *gear 1* and subnet 2 the two possible orientations of *shaft 1* and *shaft 2* leading to a total of four outputs. The loss for these two subnets is the standard categorical cross entropy. Like before, the subclass loss is only used in training for parts in the relevant class (*gear 1* for subnet 1, shafts for subnet 2). The third subnetwork regresses the orientation angle θ, similar to the one in stage 1. Since the part is almost in canonical position and the predicted angular range is within [-10, 10] degrees, no cyclic angle needs to be considered here, so the output neurons directly regressing θ (one neuron for each class).

Stage 3: In stage 3 an additional subnetwork is added to regress the bounding box of the part. Unlike in stage 1, the subnet has only 4 outputs, since there are no anchors. The loss function is the Huber loss as used in stage 2.

In all subnetworks of stage 2 and 3, a fully connected layer with 64 or 128 neurons is added after the last 2D-convolutional layer and before the output layer of the subnet, to improve the accuracy of the network. This is on cost of computation time, which was not optimized in this work. In the classification subnets a *ReLU* activation function was used after the fully connected layer, whereas in the regression subnets no activation function was used.

C. Creating Training Dataset

Our simulated environment was built in Gazebo [26] – a ROS simulation software – and consisted of a KUKA LBR IIWA simulated robot [27], to which we added the gripper, camera, table and parts. Simulated depth images were created with the Gazebo depth camera plugin.

Stage 1: For training in stage 1 and stage 2, depth images of randomly distributed parts were taken from a distance of 0.53m above the table. Due to the limited field of view of the camera only some of the parts are visible in each image, but randomization parameters were adjusted such that the total number of appearances of each part is similar. Each image contains 1 to 5 parts, with an average of 3.2 parts per image. 200,000 images were created for training with proper ground truth. The bounding box ground truth was determined by the projected center of the part in the image and the part's radius. For each part, the bounding box size is fixed and does not depend on the part orientation (Fig. 3). To estimate the ground truth rotation angle in stage 1, all near-symmetric parts were treated as symmetric. For example, if *gear 1* is placed in an orientation of 215 degrees, the ground truth angle in stage 1 would be mod(215,90) = 35. This value is then multiplied by the symmetry order to create the cosine and sine ground truth.

Stage 2: From the 200,000 images of stage 1, about 80,000 images of each part were cropped for training in stage 2. Each sub-class of the parts with near-symmetries was handled separately. When preparing the images for stage 2, the images of the parts are cropped from perturbed image positions $(\hat{x}, \hat{y}, \hat{\theta}) = (x, y, \theta) + (\delta x, \delta y, \delta\theta)$. Here (x, y, θ) is the ground truth position, and $(\delta x, \delta y, \delta\theta)$ is a random noise vector with its components drawn independently from Gaussian distributions with small σ. The cropped images hence contain parts which are slightly mis-centered and with slightly miss-aligned rotation. The ground truth rotation for this stage is $\delta\theta$, hence the network is trained to identify the object misalignment in a population of roughly centered images.

Stage 3: For training the neural network in stage 3, depth images were taken from a distance of 0.31m above the table (in the simulation environment). For each image, only one part was placed on the table, and it was (virtually) pictured with a deviation $(\delta x, \delta y, \delta\theta)$ from its actual location, as for stage 2. The annotations for stage 3 include $\delta\theta$, sub-classes of near-symmetries, and the calculated bounding box around the part.

D. Bridging the simulation-reality gap

To make the simulated depth images more similar to real depth images, the synthetic images were processed with a few steps attempting to mimic the noise and signal processing pipeline of a depth camera. First, an additive Gaussian independent pixel noise of standard deviation σ was applied to all image pixels. For each image σ was chosen uniformly in [0.5,3] cm. Although the reported noise per pixel of the real D415 depth camera is only 1cm, we use this procedure for the system to be resilient to various noise conditions, which may result from variance in illumination, camera-object distance, etc. Second, a Gaussian filter with randomized standard deviation σ was applied to the image. A similar function is often used in the real depth camera processing pipeline, but with an unknown σ parameter. We hence randomize σ uniformly in [2,5] pixels. Following these operations, the minimum and maximum depth values were limited to a certain range and then values were stretched to the range of $[0,255]$. The 2D images were finally expanded to three identical maps creating a 3D RGB greyscale image required as input by the RetinaNet network. In addition to this operation of bringing synthetic images closer to real, we take some actions to bring real images closer to synthetic. See section IV.A for details.

E. Assembly

For each part assembly, a sequence of pick-and-place operations is computed based on the estimated poses. The pick operation requires the grasping pose of the robot's end-effector with respect to the part, whereas the place operation is determined by the part's pose with respect to the *base plate.* The required pose for each operation is computed based on the estimated part and *base plate* positions. Calculation of target poses include simple homogeneous matrix multiplications. For example, the pose of the end-effector in world coordinates is obtained by the pose of the part in world coordinates multiplied by the pose of the end-effector with respect to the part. Based on the current and target pose of the end effector, motion trajectories are computed using Moveit! – a ROS software motion planning package. The planner generates a suitable trajectory in the 7D joint-angle space of the robot considering all of the robot's constraints. Thus, our assembly process consists of estimating the pose of the parts from real world images followed by applying a sequence of part pick-and-place motions calculated with Moveit!.

To handle estimation errors, a force feedback routine is added and applied to the assembly of each part. All the parts are assembled using a descend motion bringing the part to its proper position. If during the last descend of the place operation the robot's force sensor measures a significant force in the opposite direction (i.e. up in the world Z direction), an unexpected contact with the *base plate* or another part is indicated. This, in turn, shows that the part was not placed in the correct position, and the robot stops the descend motion. Such an event triggers a search algorithm for a new descend position starting from positions which are closer to the initial estimate and going outwards in circles until a position with no negative force is found (Fig. 5). To find the array of possible offset locations, we think of the problem as finding a set of positions which enables inserting a disk of radius r (for example: a shaft) into a hole of radius R ($r<R$) located in an unknown position within a squared area. The optimal search locations for this problem are arranged in equal triangles with side lengths $d = 2(R-r)$ and can be found using a known algorithm [28]. The size of the search area and the distance d are the input parameters to the algorithm. For our environment we chose a square of 5mm in all cardinal directions from the initial point (i.e., a square of 10mm x 10mm) and set the distance to d = 1 mm, which corresponds to the difference between the diameter of the central hole of the *compound gear* and the diameter of the shaft attached to the *base plate*.

IV. Experiments

A. Pre-processing

Real depth images have spatial noise effects (in addition to independent pixelwise noise) and zero-value pixels, mostly near large depth gradients where the camera has difficulties to obtain good depth estimates. To remove some of the noise, we always capture ten copies from the camera for each image and calculate the average of all none-zero pixels. All remaining zero-pixels are replaced by the average of their nearest neighbors. Finally, we use the same minimum and maximum depth limits defined during the creation of the simulated images and transform the image to a 3D RGB greyscale image.

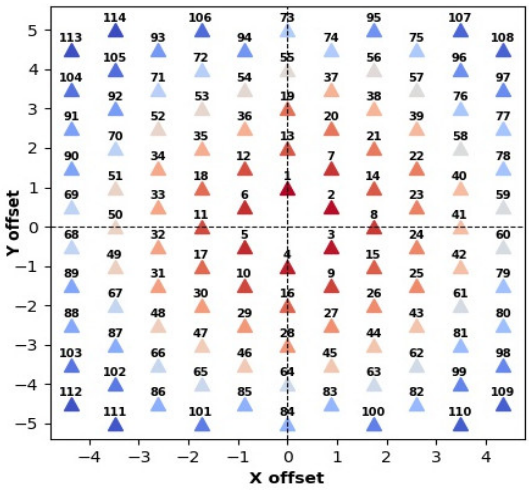

Figure 5: Searching points representing X and Y offsets in units of mm with respect to the calculated placement position, numbered and colored by the order of the points.

The parts of the *Siemens Innovation Challenge* were printed using a 3D printer. For assessing the quality of our pose estimation procedure, we had to create part configurations with known ground truth of the parts' poses on the table. For this purpose, the parts were randomly placed in simulation and their poses were saved to a file. At the beginning of a real experiment, each part was manually placed in the corner of the table – a fixed known position – and then transferred by the robot to its previously saved position on the table.

During preparations, we discovered that our gripper was not centered and aligned to the robot's flange, resulting in bad pose estimations. We measured the offset between the expected pose of the gripper and its actual pose and included this correction into the pose calculations. For similar reasons, we calibrated the position of the camera with respect to the gripper by placing a small cube on the table, taking images from several orientations above the cube and measuring the distance between the cube's center and the center of the image. After these calibrations, our results improved by 150%, but translational ground truth errors of about 1mm remained. In addition, since aligning the parts to the corner of the table was made by human assessment, we estimated a ground truth error for the rotational angle of ~0.5 degree.

B. Experiment

Nine fixed camera positions in the height of 0.53m were chosen to take images of the parts, which were randomly placed on the table in reachable distance to the robot. For each image, stage 1 and 2 of our method were initiated. Since there was overlap between the nine images, some parts were detected in several images, which required some filtering. The first filter removed small objects (shafts) near the edges of images, since objects detected near the image center enable more accurate image-to-world de-projection, leading to more accurate world pose estimates. The second filter found parts from different classes with similar predicted positions and chose the part with the highest stage 1 classification score. In this process, we removed all misclassified parts as they always had lower classification scores than the true parts. The third filter removed duplicates of the same part by choosing the detection with the highest stage 1 classification score. This filter, which implicitly relies on the assumption of a single part

TABLE 1. POSE ESTIMATION ON REAL IMAGES

Part Identity	Phase 1			Phase 2		Phase 3		
	Detection rate	*Translation accuracy (mm)*	*Rotation accuracy (deg)*	*Isometry classification*	*Rotation accuracy (deg)*	*Translation accuracy (mm)*	*Isometry classification*	*Rotation accuracy (deg)*
Base_Plate	100.00%	2.61 (1.76)	1.72 (1.58)	-	1.60 (1.15)	2.01 (1.18)	-	0.98 (0.26)
Shaft_1	100.00%	4.19 (1.90)	-	100.00%	1.33 (0.70)	2.53 (1.00)	100.00%	2.02 (0.51)
Shaft_2	100.00%	4.38 (1.61)	-	100.00%	0.86 (0.67)	2.14 (0.96)	100.00%	1.50 (0.65)
Compound_Gear	100.00%	2.32 (1.24)	0.64 (0.56)	-	0.44 (0.37)	2.25 (1.19)	-	0.87 (0.55)
Gear_1	100.00%	2.44 (1.40)	-	83.33%	1.47 (2.15) *	2.05 (1.25)	100.00%	1.25 (1.30)
Gear_2	100.00%	2.21 (1.27)	0.91 (0.58)	-	0.85 (0.39)	1.85 (0.84)	-	0.82 (0.41)
Average	100.00%	3.02 (1.77)	-	-	**1.09 (1.15) ***	**2.14 (1.09)**	-	1.24 (0.80)

Table 1: Detection and pose estimation for parts of the *Siemens Innovation Challenge*. For pose estimation, standard deviations are reported in parentheses. Isometry-breaking classification is only relevant for *gear 1*, *shaft 1* and *shaft 2*, which have near isometries.
* Calculated without isometry misclassification errors.

from each class, was only applied once when the *base plate* was detected in two different locations. Next, stage 3 of our algorithm was initiated, resulting in high-accuracy pose estimations used for the assembly.

V. RESULTS

A. Robotic assembly in a real environment

We performed a total of 30 real experiments. For each experiment, the six parts had to be detected and subjected to pose estimation, and then 5 of them (all but the *base plate*) were assembled. For cases where the robot failed to assemble a part, the parts were manually assembled, and the robot was then allowed to continue the assembly of the remaining parts. As shown in Table 1, our algorithm achieved 100% detection rate. Pose estimation resulted in an average translational accuracy of 2.14mm and a rotational accuracy of 1.09 degrees.

Out of 150 grasps and attempts to assemble, 133 were successful (88.67%) (Table 2). In 15 experiments, all parts were successfully assembled (50%). 15 assembly attempts (10%) failed due to imperfect grasps using the SAKE EZGripper, which generated too few contact points on *shaft 1*, *shaft 2*, *gear 1* and *gear 2* resulting in too weak grip forces. All the grasps were successful for the *compound gear* for which the gripper generated symmetric contact points. In 5 of the 15 failed attempts, the gripper succeeded to grasp one of the shafts but let it slip while lifting. In the remaining 10 attempts, the grasped part made contact with the *base plate* and the robot sensed a negative feedback force, which triggered the search algorithm for the correct placement position, during which the part slipped. The last 2 failed attempts (1.33%) occurred when the *base plate* was randomly placed in a certain position for which the motion planner could not find a feasible trajectory to place the part. In summary, the angular SAKE EZGripper imposed a severe constraint on the performance. We believe that by using a parallel gripper the grasping and assembly success rates could have been significantly improved, but this needs to be analyzed in future studies.

For comparison of our results with previous works we refer to two related studies. First, we were able to improve pose estimation by an order of magnitude when compared to [5], which also used training in simulation, but used RGB images of more basic parts. Second, we show assembly capabilities on more parts than considered in [10], which studied the subtask of inserting a shaft into a gear within the *Siemens Innovation Challenge* assuming fixed initial conditions.

B. Robotic assembly in simulation

In Table 3 we report pose estimation results on simulated images. The test set for stages 1 and 2 consisted of 30,000 images from a distance of 0.53m (~70,000 parts), whereas 100,000 test images from a distance of 0.31m (100,000 parts) were used for stage 3. As can be seen, our networks achieve near-perfect pose estimation results for simulated images, which are $50\times$ better than the real results for translations and $12\times$ better for rotational accuracy. This discrepancy between simulated and real images reflects the remaining simulation-reality gap, which we were not able to close.

VI. CONCLUSION

Our main contributions in this work are two-fold: First, we show that for an assembly task with given 3D CAD files, an accurate pose estimation algorithm can be trained to achieve high assembly success rates. To the best of our knowledge this is the first time that the *Siemens Innovation Challenge* has been solved with all the parts and with high successful assembly rates. Second, the simulation-to-reality gap can be significantly bridged by using depth images and the assembly task can be learned on simulated images only. Beyond this

TABLE 2. REAL TASK SUCCESS RATE

Part Identity	Grasping Success Rate	Assembly Success Rate
Shaft_1	93.33%	86.67%
Shaft_2	90.00%	80.00%
Compound_Gear	100.00%	100.00%
Gear_1	100.00%	90.00%
Gear_2	100.00%	86.67%
Average	96.67%	88.67%

Table 2: Assembly rate accuracy for parts of *Siemens Innovation Challenge*.

TABLE 3. POSE ESTIMATION ON SIMULATED IMAGES

Part Identity	Phase 1			Phase 2		Phase 3		
	Detection Rate	*Translation Accuracy (mm)*	*Rotation Accuracy (deg)*	*Isometry Classification*	*Rotation Accuracy (deg)*	*Translation Accuracy (mm)*	*Isometry Classification*	*Rotation Accuracy (deg)*
Base_Plate	100.00%	0.12 (0.07)	0.28 (0.37)	-	0.12 (0.10)	0.05 (0.03)	-	0.03 (0.03)
Shaft_1	100.00%	0.18 (0.13)	-	100.00%	0.10 (0.09)	0.04 (0.03)	100.00%	0.04 (0.03)
Shaft_2	100.00%	0.17 (0.11)	-	100.00%	0.11 (0.09)	0.04 (0.02)	100.00%	0.04 (0.03)
Compound_Gear	100.00%	0.11 (0.07)	0.09 (0.07)	-	0.03 (0.03)	0.04 (0.03)	-	0.05 (0.05)
Gear_1	100.00%	0.17 (0.11)	-	99.80%	0.10 (0.27) *	0.05 (0.03)	100.00%	0.17 (0.16)
Gear_2	100.00%	0.11 (0.07)	0.17 (0.15)	-	0.05 (0.05)	0.04 (0.03)	-	0.06 (0.05)
Average	100.00%	0.13 (0.10)	-	-	0.09 (0.13) *	0.04 (0.03)	-	0.09 (0.12)

Table 3: Detection and pose estimation on simulated images. For pose estimation, standard deviations are reported in parentheses. Isometry-breaking classification is only relevant for *Gear 1*, *Shaft 1* and *Shaft 2*, which have disturbing near isometries.

* Calculated without isometry misclassification errors.

work, we believe that the presented three-stage pose estimation algorithm is quite generic and can be applied with some adjustments to general assembly tasks. In adapting to a new task, datasets generation and network training in our method can be performed significantly faster than currently available industrial procedures. It hence has the potential for enabling more flexible manufacturing.

REFERENCES

[1] J. Mahler and K. Goldberg, "Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences", *CoRL*, 2017.

[2] S. Levine, C. Finn, T. Darrell and P. Abbeel, "End-to-End training of deep visuomotor policies," *JLMR*, vol. 17, no. 1, pp. 1334-1373, 2016.

[3] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection", *Int J Robotics Res*., vol. 37, pp. 421-436, 2018.

[4] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine and V. Vanhoucke, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep robotic grasping", http://arxiv.org/abs/1709.07857, 2017.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world". *In Intelligent Robots and Systems (IROS*), 2017.

[6] S. James, A. J and E. J. Davison, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task", https://arxiv.org/abs/1707.02267, 2017.

[7] U. Viereck, A. Pas, K. Saenko and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images", *CoRL*, 2017.

[8] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks", *Robotics and Automation (ICRA),* 2015.

[9] S. Kumra and C. Kanan, "Robotic Grasp Detection using Deep Convolutional Neural Networks*", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2017.

[10] G. Thomas, M. Chien, A. Tamar, J. A. Ojea and P. Abbeel, "Learning Robotic Assembly from CAD", https://arxiv.org/abs/1803.07635v2, 2018.

[11] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez and M. Riedmiller, "Data-efficient Deep Reinforcement Learning for Dexterous Manipulation, " https://arxiv.org/abs/1704.03073, 2017.

[12] M. Gualtieri, A. Pas, K. Saenko and R. Platt, "High precision grasp pose detection in dense clutter," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2016.

[13] I. Lenz, H. Lee and A. Saxena, "Deep learning for detecting robotic grasps," *Int J Robotics Res*, vol. 34, pp. 705-724, 2015.

[14] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural network*", Robotics and Automation (ICRA),* 2015.

[15] T.Y. Lin, P. Goyal, R. Girshick., K. He, P. Dollár, "Focal loss for dense part detection",https://arxiv.org/abs/1708.02002 , 2017.

[16] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[17] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask r-cnn," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[18] S. Wei, V. Ramakrishna, T. Kanade and Y. Sheikh, "Convolutional pose machines," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] E Krupka et al., "Toward Realistic Hands Gesture Interface: Keeping it Simple for Developers and Machines," in *Conference on Human Factors in Computing Systems (CHI)*, 2017.

[20] J. Tremblay, T. To, A. Molchanov, S. Tyree, J. Kautz and S. Birchfield, "Synthetically Trained Neural Networks for Learning Human-Readable Plans from Real-World Demonstrations," https://arxiv.org/abs/1805.07054, 2018.

[21] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand and V. Lempitsky, "Domain-adversarial training of neural networks," JMLR, 2016.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems (NIPS), 2014.

[23] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, T. Chen, A. Hutter, S. Zakharov, H. Kosch and J. Ernst, "DepthSynth: Real-Time Realistic Synthetic Data Generation from CAD Models for 2.5 D Recognition," https://arxiv.org/abs/1702.08558, 2017.

[24] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* , pp. 770-778, 2016

[25] T. Y. Lin, P. Dollár, R.B. Girshick, K. He, B. Hariharan, S.J. Belongie, "Feature Pyramid Networks for Part Detection,".*IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* vol. 1, no. 2, p. 4, 2017.

[26] N.P. Koenig, A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* vol. 4, pp. 2149-2154, 2004.

[27] C. Hennersperger, B. Fuerst, S. Virga, O. Zettinig, B. Frisch, T. Neff, N. Navab, "Towards MRI-based autonomous robotic US acquisitions: a first feasibility study," *IEEE Transactions on Medical Imaging*, 36(2), pp. 538-548. 2017

[28] R. Kershner, "The Number of Circles Covering a Set", *Am. J. Math.*, Vol. 61, No. 3., 1939.