Reinforcement-Learning Based Robotic Assembly of Fractured Objects Using Visual and Tactile Information

Xinchao Song*, Nikolas Lamb*, Sean Banerjee, Natasha Kholgade Banerjee
Department of Computer Science, *Clarkson University*
Potsdam, New York, USA
{xisong, lambne, sbanerje, nbanerje}@clarkson.edu

***Abstract*—Though several approaches exist to automatically generate repair parts for fractured objects, there has been little prior work on the automatic assembly of generated repair parts. Assembly of repair parts to fractured objects is a challenging problem due to the complex high-frequency geometry at the fractured region, which limits the effectiveness of traditional controllers. We present an approach using reinforcement learning that combines visual and tactile information to automatically assemble repair parts to fractured objects. Our approach overcomes the limitations of existing assembly approaches that require objects to have a specific structure, that require training on a large dataset to generalize to new objects, or that require the assembled state to be easily identifiable, such as for peg-in-hole assembly. We propose two visual metrics that provide estimation of assembly state with 3 degrees of freedom. Tactile information allows our approach to assemble objects under occlusion, as occurs when the objects are nearly assembled. Our approach is able to assemble objects with complex interfaces without placing requirements on object structure.**

***Index Terms*—Reinforcement Learning, Fractured Shape Repair, Machine Learning, Robotic Assembly, Sim-to-Real**

I. INTRODUCTION

Common household objects are often damaged during normal use. Objects may be dropped, fracturing off large pieces of the object or chipping the edges, and creating complex fracture surface profiles. Reassembling a fractured object requires carefully maneuvering the fractured part into the area that the part detached from. Several approaches have shown success at generating repair parts for fractured objects that have had parts lost or destroyed using symmetric reflection [1], [2], using high-resolution 3D scans of complete counterparts [3], which may be combined with a database search [4], or using machine learning [5], [6]. Though these approaches may be used to generate a repair part, theoretically enabling restoration of damaged objects at scale, they do not physically assemble the repair part and fractured object, and thus are not fully automated. We provide an approach to automate the assembly of a fractured object with its corresponding repair part using a pair of robotic manipulators, enabling existing object restoration approaches to be used at scale for industrial recycling and cultural heritage object restoration.

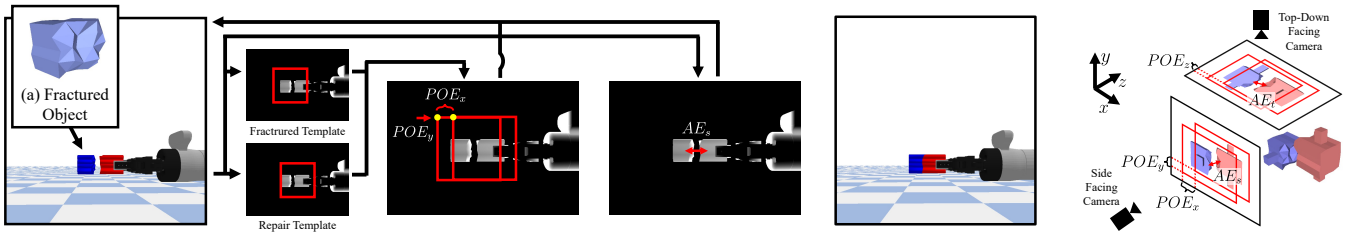
This work is supported by the National Science Foundation (NSF) under grant IIS-2023998.

*These authors contributed equally to this paper.

Assembly of repair parts to fractured surfaces is a challenging task due to the geometric complexity of the fractured region, shown in Figure 1(a). The surface irregularity of the fractured region prevents using a traditional controller to automatically assemble a repair part to a fractured object, as precise assembly of often requires approaching from a precise angle and 'settling in' the repair part. Controllers also rely on accurate pose estimation of the target object, which is challenging to acquire for fractured objects due to their complex and irregular geometry. Most existing approaches to perform automated assembly using conventional controllers are only tested on a single highly specific task and on objects with comparatively simple geometry that allow for using traditional computer vision techniques to estimate the pose of the objects with high accuracy, e.g. for assembly of peg-in-hole [7]–[9], smartphone backs [10], electrical contacts [11], and metal pegs [12], which typically do not generalize to new domains and are sensitive to parameter selection. These approaches cannot be used for fractured object assembly, as, due to the complexity and randomness of object fractures, the pose of the fractured object cannot be accurately estimated.

Recently, reinforcement learning (RL) approaches have been explored, e.g. for assembly of peg-in-hole [13]–[17], metal or wooden beams [18], [19], a stick of RAM [20], or electrical component enclosures [21]. These approaches often assume that the precise 6 degrees of freedom (DoF) pose of the manipulated or target object can be obtained in real time with high accuracy, which is challenging for fractured objects due to their complexity, especially with a single modality, e.g., either visual information [15] or tactile information [16]–[19]. Complex assembly tasks can benefit from combining multiple sensing modalities [13], [14], [20], [21], which allows control to be maintained when the assembly surface is occluded. However, current approaches typically perform assembly of objects with easily identifiable assembly states, e.g., a peg is assembled when it has been inserted into a hole, or a smartphone back is assembled when it snaps into place. Fractured objects do not have easily identifiable assembly states. Instead, their assembly must be estimated visually based on the proximity of the broken object to the repair part.

We propose a reinforcement learning based approach to assemble fractured objects using visual and tactile information.



(b) Simulation Environment (c) Matched Templates (d) Compute Pixel Offset Error (POE) (e) Compute Assembly Error (AE) (f) Assembled Repair Part (g) Error Observations From Side and Top-Down Perspectives

Fig. 1: Overview of our approach from the perspective of the side facing camera. (a) Fractured object (b) in simulation environment. (c) Templates extracted for fractured and repair objects. (d) We use the depth image to compute pixel offset error (POE) in x and y directions (POE_x and POE_y), and (e) side view assembly error (AE_s). Errors are fed as observation to (f) enable assembly. (g) We compute POE_x, POE_y, and AE_s with the side camera and POE_z and AE_t with the top camera.

We contribute two new visual metrics using template matching and deep learning to estimate the error of assembly of a fractured object and a repair part. We incorporate tactile information to allow our approach to assemble objects when the contact area may be occluded, i.e. when the objects are very near to being assembled. We compare our approach to one baseline approach and five ablated approaches, and demonstrate that our approach has the highest assembly success rate in simulation and on a real robot. We show assembly results on a real robot using a policy trained in simulation. Our approach achieves 52.33% assembly success rate on a real robot.

II. ESTIMATING ASSEMBLY

We consider the task of assembling a repair part to a fractured object, e.g. the fractured object in Figure 1(a). We assume that the repair part is coarsely aligned to the fractured object, as shown in Figure 1(b). Coarse alignment may be obtained by lifting each object in front of a depth camera, rotating the objects to obtain a coarse scan, using the scans with a pose estimation algorithm, e.g. OpenGR [22], to align a 3D model obtained from a repair approach to the scan, and using the estimated pose to inform a controller to coarsely align the objects. After the initial coarse alignment, we assume no knowledge of pose, i.e. that the final pose is not known.

We train a reinforcement learning (RL) algorithm to perform assembly of a repair part to a fractured object. Our approach has the following design goals: (a) our approach cannot place constraints on object geometry, and (b) our approach must be robust to sensor noise present in real systems. We propose two visual based metrics to estimate the error in assembly of the repair part to the fractured object, the pixel offset error (POE), and the assembly error (AE). We provide these metrics, along with proprioceptive force and torque information, as observations to the RL agent.

A. Pixel Offset Error (POE)

The POE uses the template matching approach of Cao et al. [23] to estimate if the objects are assembled in the image plane. To generate the image templates, we obtain the assembled 3D models for the fractured object and repair part. We translate the objects 500 mm from the camera and render the meshes separately in depth over a range of rotations from -10 to 10 degrees with a step size of 2 degrees in each dimension to obtain 1000 template images. We perform template matching, shown in Figure 1(c), to obtain the horizontal and vertical coordinates of the matched template for the fractured and repair meshes in each image. For an input image, we perform template matching with a sliding window of 100×100 pixels. When assembled, the objects will have overlapping templates. We compute the horizontal and vertical error as the distance between the templates in each dimension, as shown in Figure 1(d). The POE values are high if the objects are not closely assembled. For training and testing our approach, we use one side view camera and one top view camera to observe the scene. For ease of reference, we refer to POE values in the frame of the side view camera, shown by the axes in Figure 1(g). We compute the POE in the horizontal and vertical dimensions (POE_x and POE_y) as the difference between the matched template locations in the side facing camera, and in the in-plane dimension as the difference between the matched template locations in the top-down facing camera, POE_z, shown in Figure 1(g).

B. Assembly Error (AE)

For the assembly error (AE), we train a neural network based on the RexNet [24] architecture to predict the distance between two objects during assembly. To train the network, we collect depth images of assembled and nearly assembled fractured and repair object pairs. We use synthetically fractured object pairs provided by Lamb et al. [6]. To generate depth images that observe the fractured region, we fit a plane to the fractured region vertices, offset the repair part in random direction selected from a uniform distribution, and record a depth image from a depth camera located between 400 mm and 600 mm from the object. To prevent the repair part from being offset into the fractured object, we offset the repair part along a vector that is on the repair side of the plane fit to the fractured region vertices. To ensure the depth image can see the fractured and repair objects, we place the depth camera such that it lies on the plane fit to the fractured region vertices, and points towards the center of the fractured region vertices. We rotate the camera on the plane around the fracture region

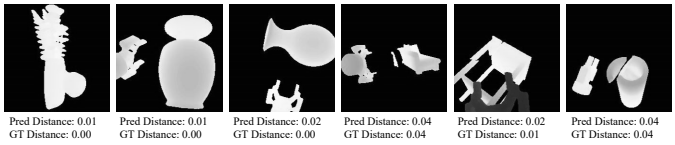


Fig. 2: Assembly error for fractured objects. Left three images are assembled, i.e. have a ground truth (GT) assembly error of 0. Right three are not assembled. Distance in meters.

vertices by μ degrees, i.e. so that the camera may observe the fracture and repair objects from multiple angles. We randomly scale the objects such that the union of the two objects is between 250 mm and 300 mm in the longest dimension.

We perform a 70%/10%/20% train/validation/test split over 24,227 fractured objects. To obtain additional data, we collect multiple depth images by rotating the camera around the fracture region for $\mu \in \{0, 45, 90, 135, 180, 225, 270, 315\}$, yielding a total of 193,816 depth images. To make our approach robust to other objects in the frame, i.e. the robot gripper, we generate 1024 depth images of the robot gripper in random orientations, and augment the training, validation, and testing images i_n by computing $i_n = \min(g, i)$, where g is the gripper augmentation image and i is the object render. We train RexNet for 252 epochs. We obtain a mean assembly distance prediction error of 0.90 mm on the test set. We show predicted assembly errors for example objects in Figure 2.

When training and testing our RL agent, we use the trained model to predict the assembly error, as shown in Figure 1(e). We compute the AE for the side view camera, AE_s, and the top view camera, AE_t, as shown in Figure 1(g), and feed the values as input to the RL agent.

III. PROBLEM FORMULATION

We model our problem as a Partially Observable Markov Decision Process (POMDP). At every timestep t, an agent executes an action $a_t \in \mathcal{A}$, causing the environment state $s_t \in \mathcal{S}$ to transition to $s_{t+1} \in \mathcal{S}$ with a conditional transition probability $\mathcal{T}(s_{t+1}|s_t, a_t)$. The agent receives an observation $o_t \in \Omega$ determined by a conditional observation probability $\mathcal{O}(o_t|s_t, a_t)$, and a reward $r_t = \mathcal{R}(s_t, a_t)$, where $\mathcal{R}$ is a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The agent attempts to maximize the expected discounted reward

$$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{1}$$

with a discount factor $\gamma \in [0, 1]$, to learn an optimal policy π_θ. We solve the POMDP problem using Proximal Policy Optimization (PPO) [25], which optimizes a surrogate objective function using multiple epochs of stochastic gradient ascent.

We define the observation space of the RL agent as

$$\Omega = [\ F_x, F_y, F_z, M_x, M_y, M_z, E\ ]^T \tag{2}$$

where F and M represent the force and torque in the x, y, and z axis applied at the end effector (EE) to the fractured object, and $E = [POE_x, POE_y, POE_z, AE_s, AE_t]$. We define a discrete action space for the RL agent. At each timestep, the RL agent takes a translational step $\pm\delta$ or a rotational step $\pm\epsilon$ along the three possible axes, resulting 6 possible rotations or translations, for 12 total actions. We define the RL agent action $a = [T_x, T_y, T_z, R_x, R_y, R_z]^T$, where T corresponds to a translation and R corresponds to a rotation in the robot's coordinate frame. We use $\delta = 0.5$ mm and $\epsilon = 1^\circ$.

For training, we define the assembly goodness g of the fractured object and repair part using a weighted sum of the Euclidean distance e_t and geodesic distance e_r between the assembled target pose and the current pose of the EE. We define the goodness as

$$g = (1 - \mu)(1 - e_t) + \mu(1 - e_r), \tag{3}$$

where μ is a hyperparameter that defines the importance of rotational error relative to translational error. We compute e_t as the norm of the vector between the current EE pose and the target pose, clamp the values between 0 and 50 mm, and normalize the values to lie between 0 and 1. We compute e_r as the geodesic distance between the current EE pose and the target pose, clamp the values between 0 and 0.16, and normalize the values to lie between 0 and 1. We use $\mu = 0.5$. A perfect assembly has a goodness $g = 1$. During training, the RL agent is tasked with maximizing goodness g.

We define the reward function $\mathcal{R}(s_t, a_t)$ at time step t as

$$r_t = \begin{cases} g + 1000 & \text{if } e_d < \eta_{sd} \text{ and } e_r < \eta_{sr}, \\ g + 10 & \text{else if } e_t < \eta_{ct} \text{ and} \\ & \quad e_r < \eta_{cr}, \\ g - 2 & \text{else if out of bounds,} \\ g & \text{otherwise,} \end{cases} \tag{4}$$

where e_d is the maximum translation error in any dimension. If e_d and e_r are less than a small threshold η_{sd} and η_{sr}, the objects are assembled and a large positive reward ($+1000$) is given. To provide intermediate rewards, we also give a closeness reward ($+10$) when e_t and e_r are less than a threshold η_{ct} and η_{cr}. We apply a penalty of (-2) if the robot moves outside of the workspace bounds. We set the boundaries to be $x_b = (-5 \text{ mm}, 35 \text{ mm})$, $y_b = z_b = (-15 \text{ mm}, 15 \text{ mm})$. If the robot applies too much force or torque, we end the episode with no penalty. We use a force threshold of 60 N and a torque threshold of 10 N $\cdot$ m. We use $\eta_{sd} = 1$ mm, $\eta_{sr} = 0.031$, $\eta_{ct} = 12$ mm, and $\eta_{cr} = 0.031$.

IV. EXPERIMENTS AND RESULTS

A. Training Setup

We perform training in simulation. We use the PyBullet physics engine and the OpenAI Gym framework to implement the simulation environment. The assembly system in simulation is equipped with two Robotiq 2F-85 adaptive grippers, illustrated in Figure 3. We insert a synthetically fractured object and repair part into the scene. We fix the fractured object in place, and task the robot to assemble the repair part to the fractured object. The robot gripper that holds the repair part starts at a random pose approximately 20 mm from

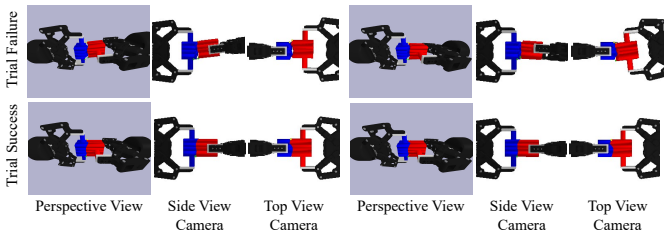


Fig. 3: (Top) trial failure and (bottom) trial success in simulation. For each trial, the right two images show the scene as observed from the side view and top view cameras. Failures are often caused by a misalignment of the repair part (top). Fractured object shown in blue, repair part shown in red.

TABLE I: Testing success rate and standard deviation of our approach and all baselines.

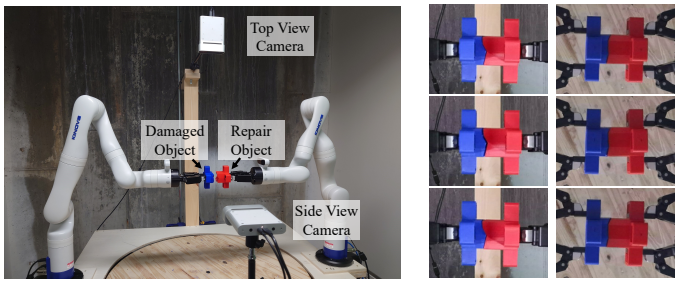
Observation	Sim Success Rate	Real Success Rate
f/t + pose	1.33% ± 2.31%	3.67% ± 6.35%
f/t + AE	43.33% ± 7.51%	3.33% ± 1.15%
f/t + $2AE$	39.00% ± 6.93%	3.00% ± 0.00%
f/t + POE	87.67% ± 5.13%	12.33% ± 2.89%
f/t + $2POE$	92.67% ± 8.50%	44.67% ± 10.79%
f/t + POE + AE	82.33% ± 2.08%	31.67% ± 7.37%
f/t + $2POE$ + $2AE$	98.33% ± 1.53%	52.33% ± 11.59%

the fractured object. We generate a set of four randomized fractured objects for training.

The PyBullet simulator requires collision objects to be convex. To generate convex fractured objects and repair parts, we create a grid of points and apply a uniform random translation to the points to simulate the complexity of the fractured surface. We apply a random translation along the grid dimensions of 0 mm to 3 mm and a random translation along the dimension tangent to the grid of 0 mm to 5 mm. We use Delaunay triangulation [26] to create a mesh from the grid of points. For each triangle, we create a second triangle offset in the dimension tangent to the grid by 40 mm. We generate a convex hull around the two triangles, and export the object as the union of all convex hulls, such that the object is a union of exclusively convex shapes. For the fractured object, we offset the second triangle in the positive direction, and for the repair part, we offset the second triangle in the negative direction. We add a bar to the synthetic fractured object and repair part to improve template matching performance. Template matching for the fractured object is performed only once at the beginning of training, as the fractured object is static. We train on a server with two Intel Xeon E5-2660 CPUs, 256 GB of RAM, and two NVIDIA RTX 3090 GPUs.

B. Testing Setup

For real-robot testing, we use two Kinova Gen3 lightweight robots with Robotiq 2F-85 adaptive grippers. We place two Azure Kinect sensors 500 mm away from the robots such that they can observe the two robots during assembly. The system is controlled by a PC with an Intel Core i7-7700K CPU, 64 GB of RAM, and an NVIDIA GTX 1080 Ti GPU. The system



(a) Real Experimental Setup (b) Trial Success

Fig. 4: (a) Our experimental setup using two Kinova Gen3 robots. (b) Three real objects successfully assembled, viewed from the side and top view cameras.

setup is shown in Figure 4(a). We collect proprioceptive force and torque information from the robot arm holding the fractured object. We perform assembly by moving the arm holding the repair part. To ensure the objects do not move in the grippers during assembly, we 3D print the fractured object and repair part with holders. The ground truth pose for the repair part is obtained by manual assembly. On the real robot we use a relaxed success threshold $e_t < \eta_{st}$ and $e_r < \eta_{sr}$, where $\eta_{st} = 3$ mm and $\eta_{sr} = 0.031$. We mask out background pixels in the image for POE and AE to improve accuracy. As for training, template matching for the fractured object is performed once at the beginning of each episode. When transferring force and torque sensing in simulation to reality, we scale the force and torque observations by manually calibrated factors.

C. Results

We perform training and testing with our approach, with one baseline approach with force, torque, and pose information observations (f/t + pose), and with five ablated approaches. The configurations we use are shown in Table I. In Table I, pose refers to EE pose relative to the start location, f/t refers to EE force and torque information, and POE and AE refer to pixel offset error and assembly error from a side view camera. For $2POE$ and $2AE$, we use both the side view camera and the top view camera. We train for a maximum of 1,000,000 steps with three different seeds for all approaches. Our approach takes 24 hours to train. We perform simulated and real testing over 100 episodes using a fractured object not present in the training set with a different seed than is used for training. Each real testing episode takes 80 seconds on average. We show the mean success rate and standard deviation in Table I.

Our approach with both cameras (f/t + $2POE$ + $2AE$) outperforms the baseline and all ablated approaches with a success rate of 98.33% in simulation and 52.33% on the real robot. Our approach consistently shows high assembly success in simulation, with a standard deviation of 1.53%, though the standard deviation increases to 11.59% on the real robot, likely due to outliers during template matching on real depth images. We observe that POE is essential to obtain consistent assembly on the real robot, as using AE only gives a success

rate of $\approx 3\%$, as shown in rows 2 and 3 in Table I. Though using two views for POE in simulation performs similarly to using one view, on the real robot using two views substantially improves performance, as shown by the 12.33% success rate for one view and 44.67% success rate for both views on the real robot. Combining the POE and AE with both views yields the best performance, as the AE provides an additional source of information in case template matching fails.

We show two successful assemblies in simulation on the bottom Figure 3, and three successful assemblies using the real robot in Figure 4(b). As shown, our approach is able to closely mate a repair part to the fractured object in both environments. Repair parts join continuously to the fractured object, and serve to repair the fractured object to its original physicality. As we use relatively low-end lightweight robots, we observe that robot holding the fractured object may be pushed from its stationary position during the assembly process. The proprioceptive force and torque observations may also be noisy. However, we show that our approach is robust to movement of the fractured object, and successfully assembles the repair part more than half the time. Replacing our low-end robots with more advanced industrial robots with higher-precision tactile sensors may additionally improve performance.

V. CONCLUSION

In this paper, we propose a reinforcement learning approach using two novel visual metrics, which we term pixel offset error and assembly error, to assemble complex fractured objects. Our approach does not place constraints on object geometry and estimates the assembly state of the constituent objects in real time. We show assembled fractured and restored pairs using our approach in simulation and on real objects.

As we rely on depth cameras to estimate object location, our system cannot be used to repair objects made of reflective material such as metal or glass. Though we train and test our approach on synthetically generated objects to accelerate training, our approach is not limited to assembling synthetic objects. As part of future work, we will perform training and testing on CAD models such as mugs, bowls, and jars, and will test our approach on physically fractured objects. We will also explore assembling objects with multiple repair parts by assembling repair parts piece by piece. By combining it with existing object repair approaches [3], our approach may be used to fully automate the repair of damaged objects.

REFERENCES

[1] G. Papaioannou, T. Schreck, A. Andreadis, P. Mavridis, R. Gregor, I. Sipiran, and K. Vardis, "From reassembly to object completion: A complete systems pipeline," *Journal on Computing and Cultural Heritage*, vol. 10, no. 2, pp. 1–22, 2017.

[2] R. Gregor, I. Sipiran, G. Papaioannou, T. Schreck, A. Andreadis, and P. Mavridis, "Towards automated 3d reconstruction of defective cultural heritage objects." in *GCH*. EUROGRAPHICS, 2014, pp. 135–144.

[3] N. Lamb, S. Banerjee, and N. K. Banerjee, "Automated reconstruction of smoothly joining 3d printed restorations to fix broken objects," in *Proc. SCF*. ACM, 2019, pp. 1–12.

[4] N. Lamb, N. Wiederhold, B. Lamb, S. Banerjee, and N. K. Banerjee, "Using learned visual and geometric features to retrieve complete 3d proxies for broken objects," in *Proc. SCF*. ACM, 2021, pp. 1–15.

[5] R. Hermoza and I. Sipiran, "3d reconstruction of incomplete archaeological objects using a generative adversarial network," in *Proc. CGI*. ACM, 2018, pp. 5–11.

[6] N. Lamb, S. Banerjee, and N. K. Banerjee, "Deepjoin: Learning a joint occupancy, signed distance, and normal field function for shape repair," in *ACM Trans. Graph.* ACM, jul 2022.

[7] T. Jiang, H. Cui, X. Cheng, and W. Tian, "A measurement method for robot peg-in-hole prealignment based on combined two-level visual sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.

[8] S. Liu, D. Xing, Y.-F. Li, J. Zhang, and D. Xu, "Robust insertion control for precision assembly with passive compliance combining vision and force information," *IEEE/ASME Transactions on Mechatronics*, vol. 24, pp. 1974–1985, 2019.

[9] A. S. Morgan, B. Wen, J. Liang, A. Boularias, A. M. Dollar, and K. E. Bekris, "Vision-driven compliant manipulation for reliable, high-precision assembly tasks," *ArXiv*, vol. abs/2106.14070, 2021.

[10] W.-C. Chang, "Robotic assembly of smartphone back shells with eye-in-hand visual servoing," *Robotics and Computer-integrated Manufacturing*, vol. 50, pp. 102–113, 2018.

[11] M. Hajduk, J. Semjon, and J. Varga, "Utilization of scara robots in the assembly of electrical contacts," *International Journal of Mechanical Engineering and Robotics Research*, vol. 6, no. 5, pp. 360–365, 2017.

[12] Y. Ma, X. Liu, J. Zhang, D. Xu, D. Zhang, and W. rong Wu, "Robotic grasping and alignment for small size components assembly based on visual servoing," *The International Journal of Advanced Manufacturing Technology*, vol. 106, pp. 4827–4843, 2020.

[13] Y. Wang, L. Zhao, Q. Zhang, R. Zhou, L. Wu, J. Ma, B. Y. Zhang, and Y. Zhang, "Alignment method of combined perception for peg-in-hole assembly with deep reinforcement learning," *J. Sensors*, vol. 2021, pp. 5 073 689:1–5 073 689:12, 2021.

[14] P. A. Zachares, M. A. Lee, W. Lian, and J. Bohg, "Interpreting contact interactions to overcome failure in robot assembly tasks," in *2021 IEEE ICRA*. IEEE, 2021, pp. 3410–3417.

[15] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *2020 IEEE/RSJ IROS*. IEEE, 2020, pp. 5548–5555.

[16] J. Xu, Z. Hou, W. Wang, B. Xu, K. Zhang, and K. Chen, "Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks," *IEEE Trans. on Industrial Informatics*, vol. 15, pp. 1658–1667, 2019.

[17] T. Inoue, G. D. Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," *2017 IEEE/RSJ IROS*, pp. 819–825, 2017.

[18] J. Luo and H. Li, "A learning approach to robot-agnostic force-guided high precision assembly," in *2021 IEEE/RSJ IROS*. IEEE, 2021, pp. 2151–2157.

[19] A. A. Apolinarska, M. Pacher, H. Li, N. Cote, R. Pastrana, F. Gramazio, and M. Kohler, "Robotic assembly of timber joints using reinforcement learning," *Automation in Construction*, vol. 125, p. 103569, 2021.

[20] Y. Shi, Z. Chen, H. Liu, S. Riedel, C. Gao, Q. Feng, J. Deng, and J. Zhang, "Proactive action visual residual reinforcement learning for contact-rich tasks using a torque-controlled robot," in *2021 IEEE ICRA*. IEEE, 2021, pp. 765–771.

[21] R. Song, F. Li, W. Quan, X. Yang, and J. Zhao, "Skill learning for robotic assembly based on visual perspectives and force sensing," *Robotics Auton. Syst.*, vol. 135, p. 103651, 2021.

[22] N. Mellado *et al.*, "Opengr: A c++ library for 3d global registration," https://storm-irit.github.io/OpenGR/, 2017.

[23] Z. Cao, Y. Sheikh, and N. K. Banerjee, "Real-time scalable 6dof pose estimation for textureless objects," in *2016 IEEE ICRA*. IEEE, 2016, pp. 2441–2448.

[24] D. Han, S. Yun, B. Heo, and Y. Yoo, "Rethinking channel dimensions for efficient model design," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 732–741.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[26] D.-T. Lee and B. J. Schachter, "Two algorithms for constructing a delaunay triangulation," *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.