2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Good Grasps Only: A data engine for self-supervised fine-tuning of pose estimation using grasp poses for verification

Frederik Hagelskjær

***Abstract*— In this paper, we present a novel method for self-supervised fine-tuning of pose estimation. Leveraging zero-shot pose estimation, our approach enables the robot to automatically obtain training data without manual labeling. After pose estimation the object is grasped, and in-hand pose estimation is used for data validation. Our pipeline allows the system to fine-tune while the process is running, removing the need for a learning phase.**

The motivation behind our work lies in the need for rapid setup of pose estimation solutions. Specifically, we address the challenging task of bin picking, which plays a pivotal role in flexible robotic setups.

Our method is implemented on a robotics work-cell, and tested with four different objects. For all objects, our method increases the performance and outperforms a state-of-the-art method trained on the CAD model of the objects. Project page available at gogoengine.github.io

I. Introduction

The automation of industrial processes enables much greater output of production at reduced prices. Successful automation is most often obtained for large batch productions, as the large production output allows for the huge development cost. But, small batch productions constitutes a large part of production tasks. And for small batches the huge development cost cannot be tolerated. Flexible set-ups using, e.g. robotics for shorter set-up times, are thus important [1].

One important aspect of flexible set-ups is the feeding process [2]. To allow for fully automatic processes, no manual insertion of objects should be required. While mechanical solutions such as bowl feeders have been created, they all require manual labor for configuration [3].

One solution that does not require any physical configuration is visual pose estimation for bin-picking. However, visual pose estimation often requires a lot of parameter tuning to obtain usable performance [4]. This added set-up time can make the solution unfeasible.

One solution to avoid manual tuning is using deep learning to obtain the parameters. However, manually collecting data is generally time-consuming and limits the usability of solutions. To overcome this, synthetic data generation has been introduced, however, a reality gap exists between the real and synthetic data. This reality gap often decreases the performance of the algorithm [5], [6]. These methods also require simulation of the data, and a subsequent training phase, both of which increase the set-up time.

This project was funded in part by Innovation Fund Denmark through the projects MADE FAST and FERA, and in part by the SDU I4.0-Lab.

All authors are with SDU Robotics, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, 5230 Odense M, Denmark frhag@mmmi.sdu.dk

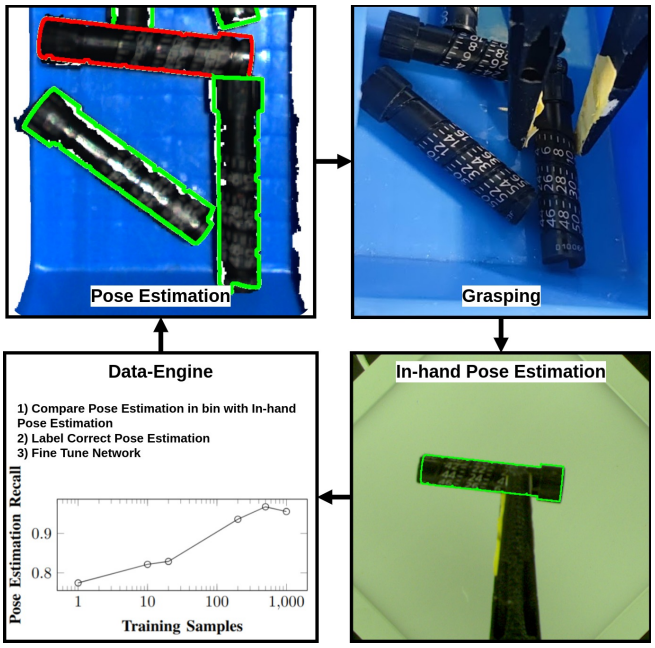


Fig. 1: The pipeline of our developed data engine. First using zero-shot a pose estimation is performed. Then the object is grasped and an in-hand pose estimation is performed. Comparing the two poses, correct pose estimations are sorted, and the network is fine-tuned. A new pose estimation is then performed and the process repeats. As data is gradually collected the network performance increases.

Zero-shot methods, which does not require training for new objects, have shown promising results in this regard [7]. However, methods trained with data of the actual object generally obtain better performance [7], [8]. In this regard self-supervised methods have shown very good results in obtaining training data automatically [6], [9]–[11]. However, the data collection is generally performed in a training phase where the robot is not operable [9]. This training phase is necessary as the data cannot be collected while the system is running. This is either because the system cannot gather data while executing the task, or because the system will not work with the initial performance. As incorrect pose estimations could result in non-recoverable errors. As the system's feasibility depends on the set-up time, this learning phase limits the usability of such systems.

In this paper, a method for online self-supervised learning of pose estimation for bin-picking is presented. The self-supervised method is built into the task, thus allowing the system to start the task immediately and then gradually improve performance. The methodology is visualized in Fig. 1.

Our method uses an existing workcell for bin-picking [12] combined with an existing in-hand pose estimation system [13]. Additionally, we replace the network of the bin-picking pose estimation with a zero-shot method [14].

The presented data engine can work with any combination of pose estimation based bin-picking and in-hand verification, however, the presented system allows several advantages. The bin picking system is able to recover from failed grasps [12], thus allowing the system to run with imperfect pose estimations. This along with using a zero-shot pose estimation method allows the system to run without any configuration. And as the method does not use color information, simple industrial CAD models can be used.

After the object has been grasped, an in-hand pose estimation verifies the object pose. The in-hand pose estimation combines template matching with stable poses. The template matching does not require any configuration or training, while the stable poses makes the solution very robust [13], [15]. This makes the set-up of the in-hand pose estimation very simple.

An additional strategy to improve the abilities of the system when starting out, is to set the initial pose estimation parameters to prioritize precision and disregard the runtime [16]. While a long cycle-time can impact the overall usability of the system, the long cycle-time is only used in the beginning. As data is collected the performance of the pose estimation improves and the cycle-time can gradually be reduced. These parts combined makes the system able to run immediately without tuning the system. We demonstrate that the method is able to work for four different objects, and that the performance is able to increase as the system runs. We also show that the system is able to learn generalities from the scene and improve performance for unseen objects.

The main contributions presented in this paper are:

- A method for creating ground-truth data for bin-picking
- A pose estimation data engine which allow for self-supervised learning during task execution
- Integration of the data engine on a work-cell for bin-picking and in-hand pose estimation
- Experiments demonstrating the effectiveness of the method

The remaining paper is structured as follows: related papers are reviewed in Sec. II. In Sec. III, the work-cell and our developed method are elaborated. In Sec. IV, experiments are performed, showing the validity of the approach. Finally, in Sec. V, a conclusion is given, and further work is discussed.

II. Related Work

Bin-picking is an important tool for flexible robotic systems [2]. As the set-up of bin-picking is generally very time-consuming, many different approaches for automatic set-ups have been developed.

One approach is to avoid the pose estimation altogether, using model-free bin-picking [17]. To improve the performance of such methods many different approaches to self-supervised bin-picking have been created [18]–[20]. An example is shown in [21] where one-shot imitation learning is combined with self-supervised learning. Using a demonstrated goal state the robot automatically learns to grasp the objects correctly. Another approach is shown in [22], where the learning is performed using simulated data. These methods show very good results and the approach seems very promising. However, model-free bin-picking is not a feasible solution to our scenario. As the manipulation of the objects can only be performed from a limited number of grasp poses the grasping needs to be guided by the object's position.

To obtain precise pose estimates for the data engine, we use KeyMatchNet [14], a colorless zero-shot algorithm. While other zero-shot pose estimation methods exist, they require color information and are generally not independent of detectors [7]. Both are requirements for our data engine. For a more in-depth explanation of zero-shot methods see [7], [14].

A. Self-supervised Pose Estimation

Self-supervised learning of pose estimation allows for increased performance without any manual labor. As a result, many different applications have been developed. One such method to automatically collect data for self-supervised learning of pose estimation is presented in [9]. The network is first trained using synthetic data. Similar to our method, this network is then used to obtain the pose estimations for fine-tuning the network. Dissimilar to our method, they use object tracking to create the dataset, where we instead reject false positives. As the initial network cannot handle difficult cases, curriculum learning is employed to start the process with simple scenes of a single object. The complexity of the scenes is then increased as the data is collected. Our method does not require this learning phase and instead starts with the actual task. As our system can automatically remove false positives we robust towards wrong pose estimations.

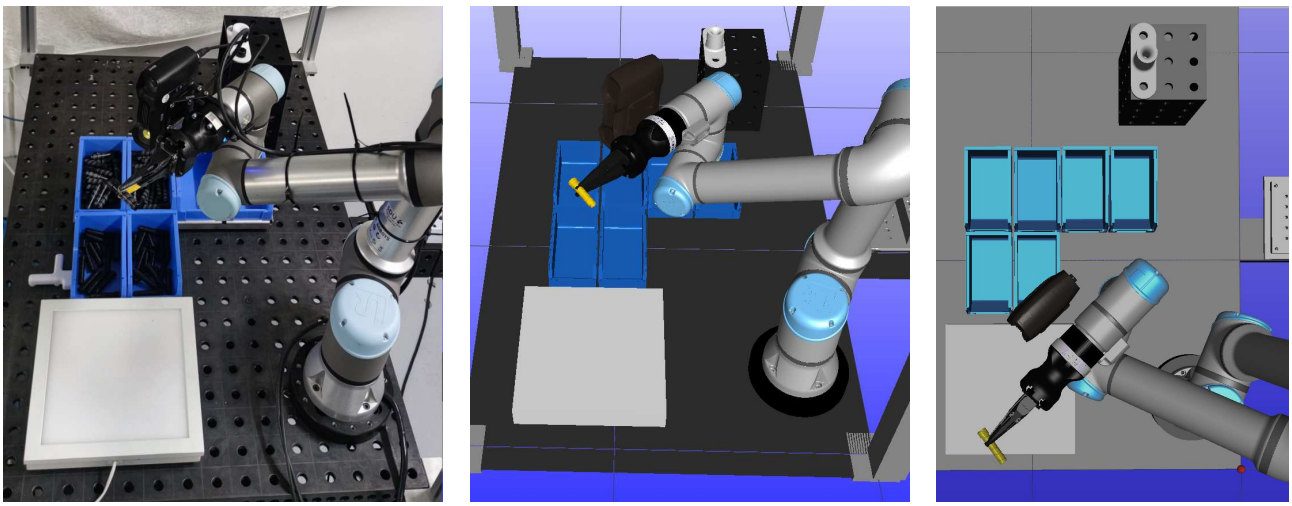
The method presented in [10] is also initially trained on synthetic data, but also adds physical simulation of the object positions. Similar to our method they employ a more expensive pose estimation system for data collection, however, they use multi-view for verification as opposed to our in-hand pose estimation. Similar to [9] the data is collected in a learning phase, and not when performing the task.

Another method pre-trained on synthetic data is presented in [8]. By using neural rendering with real, but unlabeled, images, they improve the pose estimation performance. The method is tested using a benchmarking dataset instead of a robotic set-up, and obtains good performance. However, the performance does not equate to methods trained on the ground truth information from the real data.

Similar to our method SCNet [11], use a PointNet [23] like structure for the pose estimation. However, the focus is on category-level pose estimation, and they disregard the CAD model. However, as the CAD model is available, we utilize this data.

B. Data-Engine

The automatic collection of data is an integral part of our method. Compared with to our method, many differ-



(a) Real workcell. (b) Digital twin. (c) Digital twin from above.

Fig. 2: The workcell used for experiments. The real workcell is shown in Fig. 2a with a digital twin shown in Fig. 2b. The digital twin allows for planning collision-free movements, as robot the movements are dependent on the found object poses. A top view is shown in Fig. 2c. The object bins are placed in the center. At the bottom left the background light for the in-hand pose estimation is located. The fixture is shown at the top right.

ent approaches for data collection and learning have been developed. One solution is human-robot interaction [24]. Here a human assists the robot when any failures occur. The feedback from the human is not only used to solve the occurring failure but also to learn and improve future performance. This approach could potentially be used for our system, but as the set-up should be fully automatic we have not employed any manual correction.

A different method to automatically obtain data is by recording the sensor data directly. This could then be used to predict errors in the system [25]. In the paper, they mount vibration sensors on robots to collect data on faulty motors. Different machine learning algorithms are then tested. They are able to correctly classify faulty motors based on the data. In our current system, we do not record sensor data from the robot, but this could be added in future work. This could potentially be used to classify the success of a grasp without using vision.

Another focus is the handling of collected data. CORE [26] is a recognition engine, where the relevant data is processed in a cloud server. This offloads the computational requirements from the workcell, and data storage, processing, and training are performed in the cloud. We currently perform all processing locally to simplify the set-up. In future work, data storage and network fine-tuning could be performed in the cloud. This could reduce the computational requirements for the robotic set-up, and allow for easier data sharing in a multi-robot set-up.

Another method for handling data is RoboFlow [27]. The architecture of RoboFlow is a central data engine communicating with containerized modules. The modules consist of tasks such as data preprocessing and algorithm development. The containerization makes data reuse and creating new tasks much simpler. Our current focus is on a single workcell performing the task of bin-picking, with different objects. We, therefore, do not containerize the tasks of the system. If our developed system should be diversified to perform different tasks a structure such as RoboFlow could be beneficial. A separate task is to determine how to best train with the collected data. One approach is presented with the Bridge dataset [28]. The paper does not propose a method for data collection, but instead, a methodology to improve performance when only a small amount of training data is available. The Bridge dataset is a large dataset that is included in the training when learning a specific action. The paper demonstrates that performance for the specific task increases when the large dataset is included during training. We use the same technique when fine-tuning the network by including the large dataset from KeyMatchNet [14]. We also test the methodology by including data from the other objects when fine-tuning, and demonstrate the effectiveness of this approach.

III. Method

The developed method is a data engine for self-supervised learning of pose estimation, implemented on a robotics workcell. In the following section, a full description of the developed method is given. Starting with a description of the workcell, and then the data engine is elaborated.

A. Workcell

The workcell is based on the system presented in [12]. The system was tested on a replication of the bin-picking challenge presented at the 2018 World Robot Summit [2] with state-of-the-art results. One of the aspects that allowed the system to obtain good performance is the robustness of the grasping. The system is robust to failed grasps, and can simply retry until a successful grasp is obtained. This allows the system to work, even when some of the pose estimations

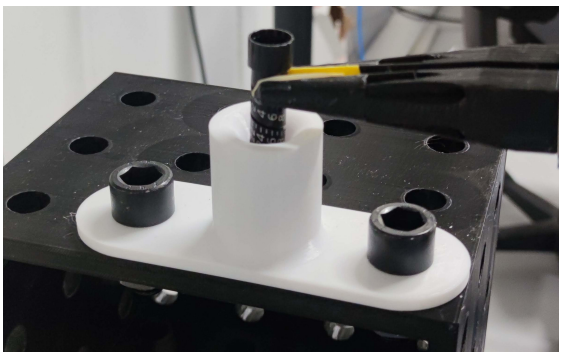


Fig. 3: Insertion of Novo A into the fixture.

fail, which is essential for our self-supervised method. The complete workcell is shown in Fig. 2.

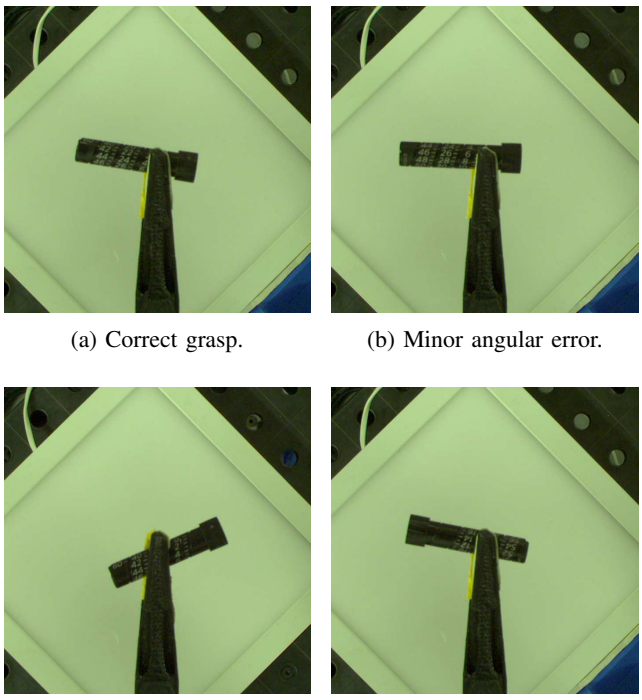
The task is the insertion of objects, and thus the position of the objects needs to be very precise. However, if the pose estimation is incorrect or if the object is moved during the grasping, the resulting grasp pose will be erroneous. An in-hand pose estimation system has, therefore, been added to obtain precise pose estimations after grasping. The in-hand pose estimation system is elaborated in subsection III-A.3.

Additionally, fixtures for the delivery of objects have been added to the workcell, these allow for a final delivery of the objects. An example of an insertion into the fixture is shown in Fig. 3.

1) Pose Estimation: The pose estimation is based on an adapted version of ParaPose [16] used in the original workcell [12]. The pose estimation method uses only depth information as the CAD models seldom have color information. The ParaPose method employs a DGCNN network [29] to compute the pose estimation, but it does not only perform a single pose estimation. Instead, multiple pose estimations are performed. A depth check is then used to remove wrong poses and a non-maximum suppression is used to remove duplicates. Thus even if several pose estimations fail, a successful pose can still be obtained. By increasing the number of pose estimations the possibility of a successful pose is increased, however, the run-time of the algorithm is also increased. When running the workcell the number of pose estimations is set to 48 to increase the possibility of obtaining data. As the self-supervised method increases the performance the number of pose estimations could be reduced, while retaining the performance.

The DGCNN network [29] is, however, replaced with KeyMatchNet [14]. KeyMatchNet is a zero-shot network and thus allows the system to operate immediately without a need for training and generating training data. Initially, the trained model presented in [14] is used, however, as training data is obtained the model is fine-tuned.

2) Expected Grasp Pose: The grasp poses are created as in the original system [12]. By combining the grasp poses with the pose estimations, a huge number of possible grasp solutions are created. By using the robot collision model a large number of these solutions are rejected and only feasible solutions remain [12]. The system then selects the shortest



(a) Correct grasp. (b) Minor angular error.

(c) Large angular error. (d) Wrong orientation.

Fig. 4: Examples of different grasps as shown from the in-hand vision system. Image a) and b) where successfully inserted, while c) and d) could not.

solution in joint space, and a grasping is attempted. This grasping approach is going for the grasp most likely to succeed, a pure exploitation strategy. While this could be changed to give a more diverse set of grasps, it is prioritized that the system runs as effectively as possible from the beginning. As the grasping solution is created by combining a pose estimation with a grasp pose, the expected grasp pose of the object is known. This expected grasp pose is used further in the created data engine.

3) In-hand Pose Estimation: The in-hand pose estimation is using an existing method [13]. This method combines classic template matching with stable poses. The stable pose results from the grasp which limits the possible poses of the object. The object is thus reduced to translation in x and y, and rotation about z, with respect to the camera. The cylindrical shape of the objects allows them to also rotate about their own z-axis while in the fingers. However, this rotation does not impact the insertion or the visual representation of the image. Examples of in-hand inspection after different grasps of Novo A are shown in Fig. 4.

When performing the in-hand pose estimation, the fingers grasping the object are placed 60 cm from the overhead camera, this is a known position ${}_{cam}T^{tcp}$. This known position is combined with the expected grasp pose of the object, ${}_{tcp}T^{obj}_{inhand}$, to create the pose used to generate the templates ${}_{cam}T^{obj}_{inhand}$. The equation is shown in Eq. 1:

$$ {}_{cam}T^{obj}_{inhand} = {}_{cam}T^{tcp}\ {}_{tcp}T^{obj}_{inhand} \tag{1} $$

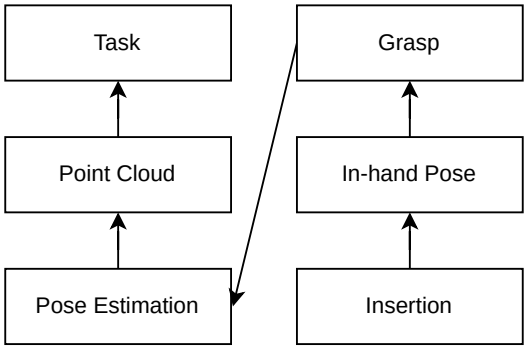


Fig. 5: The layout of the database used for collecting the data. The database structure mimics the flow of the task allowing for a simple relationship between pose estimations and the resulting insertions.

B. Self-supervised learning

The self-supervised learning is enabled by a data engine, consisting of two parts, data collection, and data labeling. As the workcell is running, and data is collected and labeled, the network is concurrently fine-tuned and improved. In the following section, the data collection, data labeling, and fine-tuning are elaborated.

1) Data Collection: The training data for the pose estimation is a point cloud with pose annotations for the object. Instead of manually labeling the poses, we use the poses found from the pose estimation. Every time a pose estimation is performed this data is saved in a database. The database is created following the "tree" pattern [30]. Meaning all the operations are automatically recorded in a parent-child relationship. The database mimics the structure of the task. Starting with the "Task" containing information about the current task, e.g. object, network type, at the top. Then all point clouds were obtained during the task, then pose estimations, grasps, in-hand pose estimations, and finally insertions. The tree structure disallows cyclical relationships simplifying look-ups. Thus an easy relationship can be made between the position of an in-hand pose estimation and the preceding object pose estimation in the bin. The database structure is shown in Fig. 5.

2) Data Labeling: As the system is running, it is continually obtaining pose estimations of objects in the bin. However, many of these pose estimations are expected to be incorrect. As a result the data will not only consist of true samples, but also false samples.

Automatic Data Labeling: While the obtained data could be sorted manually, this would severely hinder the usability of the developed method. Instead, a auto-labeling strategy is employed using the collected data. The auto-labeling strategy uses the in-hand pose for verification. As the in-hand pose estimation use the stable-grasp and the background-light, it is both able to eliminate clutter and occlusion, and limit the pose estimation to 3 Degrees of Freedom. As a result these poses are very precise.

The found in-hand pose, ${}_{tcp}T^{obj}_{inhand}$, is then compared with the expected grasp pose, ${}_{tcp}T^{obj}_{grasp}$. The expected grasp pose is the planned pose for grasping the object. If an object pose is correct, the expected grasp pose should match the found in-hand pose.

The grasp pose comparison is performed using the metric from [31]. As the objects are cylindrical the e_{ADI} score is used [32], [33]. The e_{ADI} score compares the average minimum point distance between the two point clouds. Additionally, to ensure that the orientation of z-axis of the object is correct, an angular check is used. The equation is shown in Eq. 2, where e_θ is the angular error. The distance threshold is set to 2mm, this is much lower than in the original metric [32], [33]. However, this is done to only accept very precise pose estimates.

$$TP(e_{ADI}, e_\theta) = \begin{cases} 1, & \text{if } e_{ADI} < 2.0mm \wedge e_\theta < 15^\circ \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

As many of the errors occur as a result of the object pose being rotated 180° a method to use these poses is implemented. Before calculating the overlap in Eq. 2, if the angular error is larger than 90° the object is flipped 180° about the center. This is performed both for the in bin pose and the expected grasp pose. The equation is shown in Eq. 3, where R is the rotation matrix and R_y is a rotation matrix 180° about the y-axis.

$$f(R, e_\theta) = \begin{cases} RR_y, & \text{if } e_\theta > 90^\circ \\ R, & \text{otherwise} \end{cases} \quad (3)$$

Good Grasps Only: From the automatic labeling the true and false samples are sorted into True Positives (*TP*) and True Negatives (*TN*). But, any movement of objects during grasping will result in False Positives (*FP*) and FN. Additionally, as only a single object in the bin is labeled using the auto labeling system all remaining objects are *FN*.

But, as a result of the bin-picking scenario and the KeyMatchNet [14] algorithm all *FN* can be completely ignored. This is because KeyMatchNet is only trained on positive samples, as it simply performs instance segmentation and keypoint prediction. And as the task is homogeneous bin-picking, there is no need for detection or object class distinction. The position of the bins are known and the contents of the bin is known. The task is simply to estimate the pose of the objects for bin-picking.

Thus we only need to verify *TP* from the training data. And all negatives, regardless of them being *TN* or *FN* can be discarded. The only loss occurring by discarding *FN* is less training samples, and thus potentially a more biased data set.

Any collision during grasping could make a perfect object pose not match the in-hand pose, resulting in a *FN*, which is simply discarded. On the other hand *FP* are very unlikely to occur. They only happen; if by a collision a wrongly pose estimated object is turned into a perfect grasp. The extremely rare occurrence of *FP* makes their contribution negligible. The usage of data instances according to their predicted condition is shown in Tab. I.

3) Network Training: When training data has been obtained the network is fine-tuned using the original KeyMatchNet network as a starting point. The same hyper-parameters

TABLE I: Confusion matrix showing the usage of different instances according to their condition.

		Correct pose estimation	
		True	False
In-hand pose estimation matching expected grasp	Positive	**Training Data**	Very Unlikely
	Negative	Discarded	Discarded

used for training the original network are also used for the fine-tuning.

The general approach for fine-tuning is to freeze the network, except for the last layers which are randomized, and then retrained [34]. However, to retain the zero-shot abilities of the network this method is not applicable. Additionally, as the original network was trained on synthetic data, by including the newly collected real data the features could incorporate information from the scenario. This would allow the method to perform better on new objects in the same workcell. The network is, therefore, trained without freezing the weights, and to avoid over-fitting to the new smaller dataset, the original dataset is included during training [28]. For a fair comparison of the number of training samples obtained from the data engine, each epoch always consists of 2000 samples from both datasets.

As only a few samples of the new object are present, the key-point sampling is randomized during training. For 40% of the training samples, random key-points are sampled instead of using the Farthest point sampling. This allows the key-point positions to be spread more randomly on the object, making the distinction more difficult for the network.

IV. Experiments

To test the effectiveness of the developed data-engine several different experiments have been performed.

A. Test Objects

The test objects are four different cylindrical objects, three Novo Nordisk components, and a screw from the WRS 2018 Assembly Challenge [2]. The objects are shown in Fig. 6. The surface of the Novo Nordisk objects is plastic, while their color differ. One black with white texture, one white and one black, and the WRS screws surface is shiny metallic. These different surface properties are expected to result in unique artifacts in the resulting point clouds. This along with the different sizes of the objects will result in varying results for the pose estimation algorithm.

The objects are all cylindrically shaped, an object type that is very prevalent in the industry. One important task with such objects is the grasping and insertion from unknown poses [2]. For the insertion to be successful the position should be very precise. While small errors in the pose can be compensated by the in-hand pose estimation system, many grasps poses do not allow insertion of the object. If e.g. the object is grasped too close to the bottom or if the object is rotated it cannot be inserted. A good initial pose estimation is, therefore, necessary.

B. Data Collection

In a real application, the data would be collected as the system is executing a task. However, for this paper, a data collection was performed to obtain the data. Data was collected for each of the four objects by running the bin-picking and in-hand inspection until 1000 training samples and 200 test samples had been obtained.

Fig. 6: The four objects used for experiments. From left to right: Novo A, Novo B, Novo C and WRS Screw. The longest object is 61mm and the shortest is 32mm.

During the data collection, the objects are placed in two bins, one with thirty objects and another with ten objects. Starting with the filled bin twenty objects are grasped and moved to the bin with ten objects. This continues until all the samples have been obtained or if the bins become empty. If the bins become empty they are refilled, at the start of the experiment and the data collection continues. The bin can become empty for two reasons, one reason is if the objects are dropped after grasping, as a result of an unstable grasp. Another reason is if the robot pushes objects out of the bin while grasping.

C. Pose Estimation Performance

To demonstrate the effectiveness of the developed method, the pose estimation performance is shown. Performance is shown as a result of the number of training samples used for fine-tuning. Results are shown for zero-shot pose estimation and using 1, 10, 20, 500, and 1000 training samples for fine-tuning. For comparison, we also show results for the ParaPose network, trained on synthetic data [12].

The pose estimation is tested using the same method as in KeyMatchNet [14]. Here a single point cloud is processed by the network, and using Kabsch-RANSAC [35], [36] the pose estimation is found. As the test data was obtained using the full ParaPose [16] method with multiple pose estimations and depth checks, the single pose estimation used in testing is not expected to pose estimate the object perfectly.

The results are shown in Fig. 7. It is seen, that for all objects there is a correlation between the number of samples and the pose estimation recall. It is also interesting how much the performance increases with only a few amount of samples, with even a single sample showing improvement. For all objects, at 1000 samples the performance using self-supervised learning outperforms ParaPose trained on synthetic data. However, the performance for each object differs very much. At 1000 samples the recall for Novo A is 0.98, while Novo C is only 0.67, and WRS Screw is

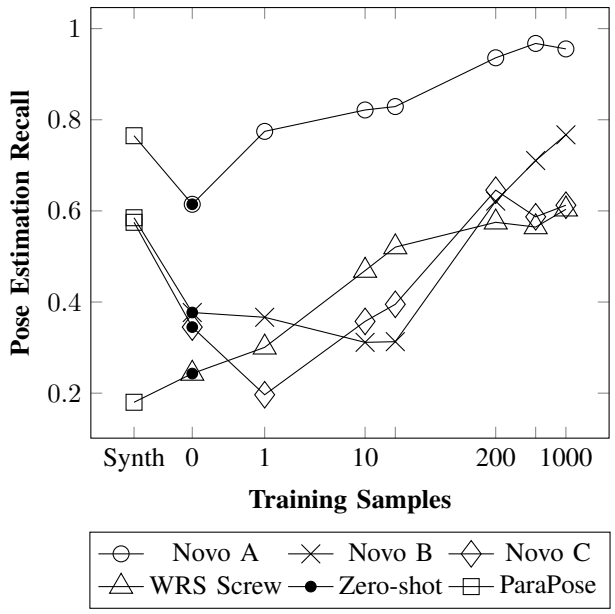


Fig. 7: Pose estimation accuracy as a result of number of training samples, for the four objects. To the leftmost results are also shown for ParaPose [16] trained with synthetic data of the CAD models.

0.64. But, for each the growing number of samples results in growing performance. It is thus seen that the data-engine is a viable method for a set-up to instantly start working, and then gradually increase performance.

D. Cross-Validation Generalizability

A cross-validation experiment is performed to test the method's generalizability to new objects. By learning scenario characteristics with the data engine, it is expected that the performance of novel objects will be improved. In this experiment, we show performance for each object, where the network is fine-tuned using the other objects. We also show performance for the pose estimation when the network is fine-tuned on all four objects. The result of the experiment is shown in Tab. II.

From the results, several interesting observations are seen. By including the other objects during training a better performance is obtained for all objects, except Novo A which shows slightly lower performance. An even more significant finding is the performance when the test object is excluded in the fine-tuning. For all objects, the performance is better compared with the non-tuned network. And, for two objects, the performance is actually better compared with training only on the test object. This indicates that the network generalizes to the scenario and not only learns object-specific features. When introducing novel objects to the workcell, the previous processing of other objects will thus help improve the performance.

E. Grasping and Insertion

The task of the workcell is the correct insertion of the objects. To determine the data engine's ability in this regard, an insertion experiment has been performed. The experiment is performed using the object Novo A. We perform the full pipeline with pose estimation, grasping, in-hand inspection, and insertion. For this test, the full pose estimation method is used with multiple pose estimations and the depth check. We show results for the zero-shot method and with the fine-tuned network using 1000 samples. We also show results for the method using 48 and 6 pose estimations. 48 is the number of pose estimations used during the data collection.

The results are shown in Tab. III. From the results, it is seen that the grasping performance is very similar for all configurations. Only around five percent of found objects are not grasped successfully. However, this does not indicate if the objects are grasped erroneously. When comparing with the success rate of the insertions, the performance varies much more. The self-supervised method vastly outperforms the zero-shot approach, even with a batch-size of only six. Thus by using the data engine the performance of the algorithm could be improved while reducing the run-time. The developed method is thus able to improve the performance of the robotic workcell.

TABLE II: Pose estimation recall as a result of fine-tuning when 500 training samples have been obtained for each object. Results are shown without any fine-tuning, fine-tuning with the test object, fine-tuning with the test object excluded, and with all four objects for fine-tuning.

	Novo A	Novo B	Novo C	WRS Screw
ParaPose [16]	0.77	0.59	0.58	0.18
Zero Shot	0.62	0.38	0.35	0.24
Object Excluded	0.66	0.50	0.62	0.69
Test Object Only	0.97	0.71	0.59	0.57
All Objects	0.95	0.74	0.66	0.64

TABLE III: The success rate of insertions for the Novo A object, as a result of different numbers of pose estimations used. Results are shown without any training data and using 1000 training samples.

	Zero-Shot		**Self-supervised**	
Pose Estimations	48	6	48	6
Grasping	95.9	95.6	95.5	95.4
Insertion	73.1	76.0	86.7	82.0

V. Conclusion

In this paper, we have presented a novel data engine for self-supervised fine-tuning of pose estimation. The data engine allows a workcell to operate without configuration while gradually improving performance. It is demonstrated that the self-supervised method outperforms a state-of-the-art method trained on the object CAD model.

Additionally, tests on four different objects have shown that the self-supervised learning generalizes to novel objects. It is thus possible for the workcell to start with better performance if it has already processed other objects.

In further work, an exploration strategy could be used when selecting objects to grasp. This could improve the variability of the training data. Other labeling strategies could also be integrated to improve the data collection, using e.g. multi-view to verify correct poses instead of only using the grasping and in-hand pose estimation.

The developed framework can be used to gather data from many different objects. This would allow the training

of a more general object pose estimation using the scene information such sensor and object characteristics.

REFERENCES

[1] Y. Yokokohji, Y. Kawai, M. Shibata, Y. Aiyama, S. Kotosaka, W. Uemura, A. Noda, H. Dobashi, T. Sakaguchi, Y. Maeda *et al.*, "World robot summit 2020 assembly challenge–summary of the competition and its outcomes," *Advanced Robotics*, vol. 36, no. 22, pp. 1174–1193, 2022.

[2] Y. Yokokohji, Y. Kawai, M. Shibata, Y. Aiyama, S. Kotosaka, W. Uemura, A. Noda, H. Dobashi, T. Sakaguchi, and K. Yokoi, "Assembly challenge: a robot competition of the industrial robotics category, world robot summit–summary of the pre-competition in 2018," *Advanced Robotics*, vol. 33, no. 17, pp. 876–899, 2019.

[3] S. Mathiesen, L. C. Sørensen, D. Kraft, and L.-P. Ellekilde, "Optimisation of trap design for vibratory bowl feeders," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3467–3474.

[4] F. Hagelskjær, A. G. Buch, and N. Krüger, "Does vision work well enough for industry?" in *VISIGRAPP (4: VISAPP)*, 2018, pp. 198–205.

[5] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "Bop challenge 2020 on 6d object localization," in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 577–594.

[6] G. Wang, F. Manhardt, X. Liu, X. Ji, and F. Tombari, "Occlusion-aware self-supervised monocular 6d object pose estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[7] T. Hodan, M. Sundermeyer, Y. Labbe, V. N. Nguyen, G. Wang, E. Brachmann, B. Drost, V. Lepetit, C. Rother, and J. Matas, "Bop challenge 2023 on detection segmentation and pose estimation of seen and unseen rigid objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5610–5619.

[8] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari, "Self6d: Self-supervised monocular 6d object pose estimation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 108–125.

[9] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3665–3671.

[10] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 545–551.

[11] S. Yu, D.-H. Zhai, and Y. Xia, "Robotic grasp detection based on category-level object pose estimation with self-supervised learning," *IEEE/ASME Transactions on Mechatronics*, 2023.

[12] F. Hagelskjær, D. Kraft *et al.*, "Off-the-shelf bin picking workcell with visual pose estimation: A case study on the world robot summit 2018 kitting task," in *2024 21st International Conference on Ubiquitous Robots (UR)*. IEEE, 2024, pp. 145–152.

[13] F. Hagelskjær and D. Kraft, "In-hand pose estimation and pin inspection for insertion of through-hole components," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 382–389.

[14] F. Hagelskjær and R. L. Haugaard, "Keymatchnet: Zero-shot pose estimation in 3d point clouds by generalized keypoint matching," *arXiv preprint arXiv:2303.16102*, 2023.

[15] F. Hagelskjær, T. R. Savarimuthu, N. Krüger, and A. G. Buch, "Using spatial constraints for fast set-up of precise pose estimation in an industrial setting," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 1308–1314.

[16] F. Hagelskjær and A. G. Buch, "Parapose: Parameter and domain randomization optimization for pose estimation using synthetic data," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6788–6795.

[17] A. Cordeiro, L. F. Rocha, C. Costa, P. Costa, and M. F. Silva, "Bin picking approaches based on deep learning techniques: A state-of-the-art survey," in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2022, pp. 110–117.

[18] L. Berscheid, T. Rühr, and T. Kröger, "Improving data efficiency of self-supervised learning for robotic grasping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2125–2131.

[19] K. Suzuki, Y. Yokota, Y. Kanazawa, and T. Takebayashi, "Online self-supervised learning for object picking: detecting optimum grasping position using a metric learning approach," in *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2020, pp. 205–212.

[20] Q. Shao, J. Hu, W. Wang, Y. Fang, W. Liu, J. Qi, and J. Ma, "Suction grasp region prediction using self-supervised learning for object picking in dense clutter," in *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*. IEEE, 2019, pp. 7–12.

[21] L. Berscheid, P. Meißner, and T. Kröger, "Self-supervised learning for precise pick-and-place without object model," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4828–4835, 2020.

[22] C. Zhao, Z. Tong, J. Rojas, and J. Seo, "Learning to pick by digging: Data-driven dig-grasping for bin picking from clutter," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 749–754.

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[24] G. Boschetti, T. Sinico, and A. Trevisani, "Improving robotic bin-picking performances through human–robot collaboration," *Applied Sciences*, vol. 13, no. 9, p. 5429, 2023.

[25] C. Nentwich, S. Junker, and G. Reinhart, "Data-driven models for fault classification and prediction of industrial robots," *Procedia CIRP*, vol. 93, pp. 1055–1060, 2020.

[26] W. J. Beksi, J. Spruth, and N. Papanikolopoulos, "Core: A cloud-based object recognition engine for robotics," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4512–4517.

[27] Q. Lin, G. Ye, J. Wang, and H. Liu, "Roboflow: a data-centric workflow management system for developing ai-enhanced robots," in *Conference on Robot Learning*. PMLR, 2022, pp. 1789–1794.

[28] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine, "Bridge data: Boosting generalization of robotic skills with cross-domain datasets," *arXiv preprint arXiv:2109.13396*, 2021.

[29] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics*, 2019.

[30] R. A. Haraty and G. Stephan, "Relational database design patterns," in *2013 IEEE 16th International Conference on Computational Science and Engineering*. IEEE, 2013, pp. 818–824.

[31] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*. Springer, 2014, pp. 462–477.

[32] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.

[33] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis *et al.*, "Bop: Benchmark for 6d object pose estimation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 19–34.

[34] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, pp. 1–40, 2016.

[35] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.

[36] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.