

# The Furtherance of Autonomous Engineering via Reinforcement Learning

Doris Antensteiner<sup>a</sup>, Vincent Dietrich<sup>b</sup> and Michael Fiegert<sup>c</sup>

*Siemens Technology / Siemens AILab, Munich, Germany*

**Keywords:** Autonomous Engineering, Reinforcement Learning, Artificial Intelligence, Industrial Robotics, 6D Pose Estimation, Computer Vision.

**Abstract:** Engineering efforts are one of the major cost factors in today's industrial automation systems. We present a configuration system, which grants a reduced obligation of engineering effort. Through self-learning the configuration system can adapt to various tasks by actively learning about its environment. We validate our configuration system using a robotic perception system, specifically a picking application. Perception systems for robotic applications become increasingly essential in industrial environments. Today, such systems often require tedious configuration and design from a well trained technician. These processes have to be carried out for each application and each change in the environment. Our robotic perception system is evaluated on the BOP benchmark and consists of two elements. First, we design building blocks, which are algorithms and datasets available for our configuration algorithm. Second, we implement agents (configuration algorithms) which are designed to intelligently interact with our building blocks. On an exemplary industrial robotic picking problem we show, that our autonomous engineering system can reduce engineering efforts.


## 1 INTRODUCTION


Continuously increasing need for autonomous and dynamic industrial production lines leads to an ongoing spread of robotic solutions as well as an increasing requirement for autonomous robotic interactions. State of the art industrial robotic systems are configured manually by experienced and trained engineers. Our goal is to solve robotic engineering tasks automatically, by learning optimized solutions for each environment. We do this by implementing learning algorithms and utilizing common interfaces such as the OpenAI Gym toolkit (Brockman et al., 2016) to interact with our environment in a standardized way. This enables an effortless exchange of learning algorithms independent of the environment. To evaluate our robotic perception system, we are utilizing the publicly available and commonly used BOP benchmark (Hodaň et al., 2020).


As an applicable precedent for industrial robotic engineering tasks, we designate robotic picking problems to be well suited. Finding optimal 6D object

poses (consisting of the object's position and orientation in a 3D space) in a scene for industrial robotic picking applications is essential and has been developed rapidly in recent history. Not only does this technology offer a vast field of applications, it has also been shown to be successful enough to lift future burdens in industrial settings. Nevertheless, today such systems still bear a major challenge, which is the need of adaption and configuration to the task by a trained engineer. This process is time consuming and expensive. Therefore, we are introducing a novel approach for the furtherance of autonomous engineering. By utilizing learning techniques, we instigate the robots autonomous adaption to new and challenging situations, such as different environments, objects, reflection types, occlusions, shadows, illumination or object surface structures.

Applications for our approach lie in the field of robotic picking in industrial environments of multiple ordered or unordered objects with challenging surface structures (textureless / highly reflective), object handling from industrial linear transport lines as well as assembly tasks.

<sup>a</sup>  <https://orcid.org/0000-0003-2083-0135>

<sup>b</sup>  <https://orcid.org/0000-0003-0568-9727>

<sup>c</sup>  <https://orcid.org/0000-0002-6371-6394>

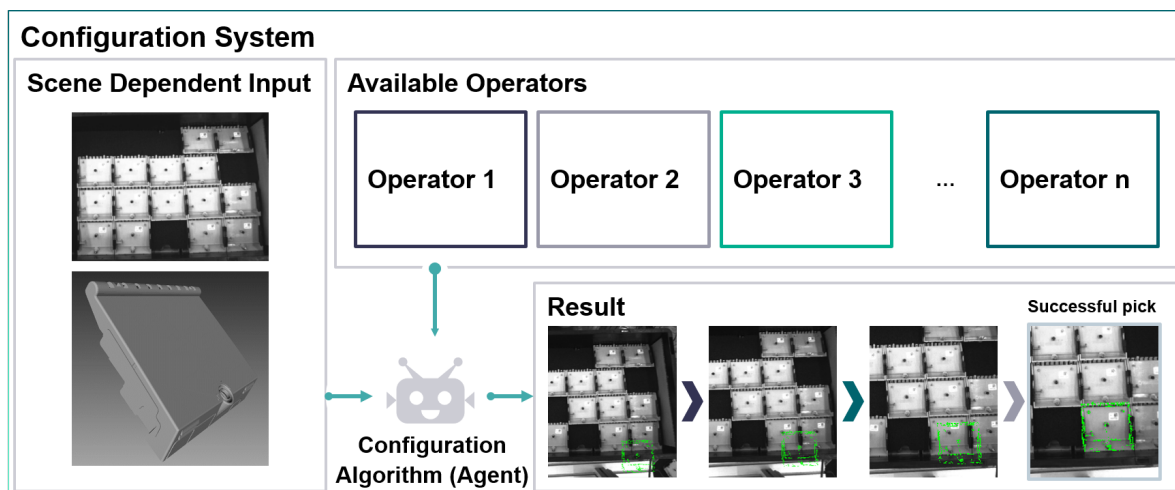


Figure 1: Illustration of the configuration system on an exemplary task. The configuration algorithm (agent) chooses operators out of a given set, considering the scene input data (input images and object models) and generates a resulting pipeline. Green lines on the result images indicate the currently estimated position of the object. In this pipeline, operators are chosen to refine the estimated 6D object pose, until a successful pick is achieved by the robot.

## 1.1 Contribution

Our approach combines two elements. First, we transfer the operative elements (in our case the 6D pose estimation, refinement and scoring functions, as well as the datasets) into formally modeled building blocks. This is needed for generating well-designed interfaces, which are essential for our configuration algorithm to function properly. Second, we apply the configuration algorithms (being categorized as: hypothesis generation, hypothesis refinement and hypothesis scoring). We demonstrate a set of learning approaches (including reinforcement learning and an evolutionary method), which interact with our formally modeled blocks, utilizing the standardized interfaces as defined by the OpenAI Gym toolkit. The main contributions of this paper comprise:

- A systematic organization of our procedural knowledge in the form of building blocks as well as suitable interfaces for perception tasks.
- The evaluation of configuration algorithms which interact with our building blocks.
- A thorough investigation of the interaction of our configuration algorithms with our building blocks and the evaluation of all interacting elements with a focus on the task of 6D pose estimation for industrial robotic applications.

## 2 RELATED WORK

In this work, three essential elements are combined together. First, for our specific demonstrative application, 6D pose estimation algorithms are utilized to solve robotic picking tasks in simulation. Second, the configuration elements, including the 6D pose estimation algorithms as well as different datasets, are described as formally modeled building blocks in order to allow for a standardized interaction. Third, learning algorithms interact with those building blocks through a standardized interface (where we chose OpenAI Gym). In this section we present the related work of each of those elements.

### 2.1 Robust 6D-Pose Estimation

Fast and robust image-based object detection algorithms are essential for industrial robotic picking pipelines. Various algorithms were introduced in the past, each having individual benefits and drawbacks. Some algorithms show high robustness, but only for textured and Lambertian objects (Lowe, 2004; Sun et al., 2011; Yeh et al., 2009). Lambertian objects exhibit ideal diffuse reflection and are well-studied. More complex challenges can be targeted in various ways. A popular approach is the use of depth sensors and RGB-D cameras (Choi and Christensen, 2016; Tang et al., 2012; Song et al., 2017) to find a stable solution. Using multi-view imaging, a non-Lambertian reflection can be suppressed during computation by enforcing a Lambertian estimation, which is robust

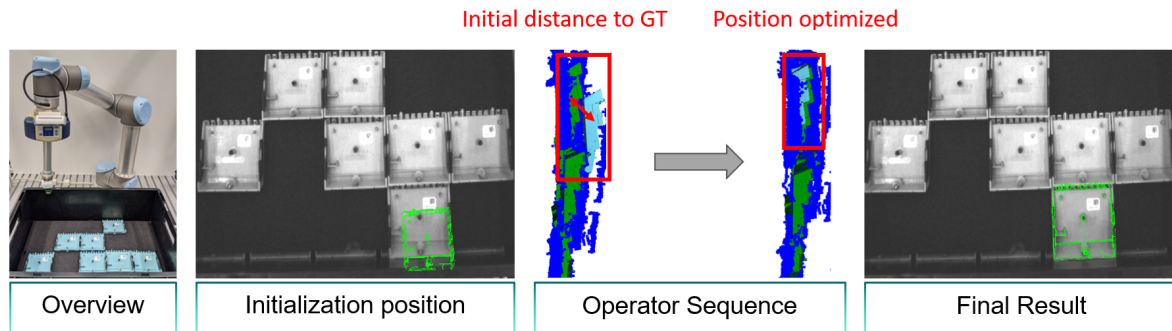


Figure 2: Picture from the view of the robotic arm and a 3D point-cloud illustration from a side view. Shown from both the initial position as well as the optimized position which results in a successful pick.

to outliers (Hailin Jin et al., 2003). This approach got special attention in the field of light field imaging (Wanner and Goldluecke, 2013). Photometric stereo can be used to model higher order reflections by placing multiple light sources in a systematic position towards the scene (Chen et al., 2020).

Many such approaches are complementary to each other and are applicable in special environments (e.g. high depth variations, symmetries, surface reflections). We demonstrate a formal modeling structure, which allows for dynamic combinations of various algorithms and hence enables the selection of the optimal combination of algorithms for each individual problem (e.g. unordered and highly reflective objects in a bin together with symmetric matte samples).

## 2.2 Hierarchical Modeling

Human engineers perform hierarchical tasks given their prior experience and ability to reason by taking a sequence of actions. This is a complex process and enables taking abstract decisions even for unknown environments. To achieve this with a configuration algorithm, abstract models are required to translate these sets of actions to well-defined executable routines.

For these action spaces, hierarchical planning can be used to find optimal results. While for small action spaces, a simple search or brute-force approach can find a sufficient solution, hierarchical planning can be used to find improved solutions for large action spaces and/or time consuming action evaluations. In such cases a simple search would not succeed. Therefore, a large problem is factorized and abstracted with a self-defined model. For our problem, we reduce the action space by a different approach. Instead of using self-defined models, we learn models for an abstraction layer in order to take successful steps in large or costly action spaces.

Prior knowledge of human engineers can be interpreted as strategically developed intuition about the behavior and use of building blocks, which are easy and cheap to simulate. We connect such blocks in the best prior estimated way (e.g. use the most efficient sequence of optimizers) in order to fulfill a specific goal (e.g. pick an object with an industrial robot). By utilizing the hierarchical modeling structure demonstrated in (Kast et al., 2019), we provide standardized interfaces for a structured management of 6D hypothesis generation, refinement and scoring operations. This permits us to use and combine different approaches (blocks) for planing, learning as well as finding optimal solutions.

## 2.3 Learning Algorithms

We utilize machine learning approaches on the example of robotic picking solutions, where the learning algorithms automatically configures and refines parameters for a given scene. Industrial tasks of such nature are today still solved by human engineers. Our approach will allow for a faster deployment and reduce the required ongoing engineering efforts. Such directions of industrial robotic automation were previously discussed in e.g. (El-Shamouty et al., 2019; Kleeberger et al., 2020).

Our configuration algorithm decides which action (e.g. a specific optimization algorithm) to perform next in a given situation (e.g. pick a metallic item from an unordered box). We are demonstrating this by implementing configuration algorithms such as reinforcement learning algorithms and evolutionary approaches.

Previously, automatic decision making for large action spaces in unknown environments was targeted by approaches such as AutoML (Hutter et al., 2018). AutoML makes machine learning more accessible by reducing the need of human expertise. It takes a dataset, optimization metric and constraints as input

and finds a suitable machine learning model for the task. Another approach for predicting the best success in specific areas is matrix factorization. This was successfully employed for recommender systems (Koren et al., 2009). The goal is to predict the best ratings for a specific task (or user) in a matrix of options. A method for parameter optimization in large spaces was presented by (Hutter et al., 2011) (“SMAC”), which aims to solve general algorithm configuration problems.

We approach the problem by forming our action space based on building blocks of procedural knowledge. These building blocks are 6D pose (estimation / refinement, and scoring) algorithms, which perform differently in varying environments (e.g. difficult surface structures or object shapes). Additionally, our used and acquired datasets are also formulated in building blocks. Our configuration algorithm learns how to pick the best 6D pose algorithm for a given task (e.g. pick a shiny object) under a defined state (e.g. current position evaluation).

### 3 PERCEPTION PIPELINE ENGINEERING

We implement an operator library which facilitates the standardized and dynamic selection and parameterization of the formally modeled 6D pose algorithmic elements. We implemented three types of such elements, namely, blocks for hypothesis generation for 6D object poses, hypothesis refinements as well as hypothesis scoring elements. The latter are used to evaluate the current position. Each operator element is modeled using a hierarchical modeling and planning structure as discussed in Sec. 2.2. The formally modeled algorithmic elements will be described in the following conceptually. The specific implementations for our exemplary application (industrial robotic picking) will be described in Sec. 5.1 for each element. Additional formally modeled but non-algorithmic elements are datasets, which will be described in Sec. 5.2.

#### 3.1 Hypothesis Generation

In our work we deal with a subproblem of robotic picking tasks, namely the 6D pose estimation of objects. The initial step for our 6D pose estimation is the generation of a hypothesis  $h_0 \in \mathbb{R}^6$  with a generation operator  $O_G$ :

$$h_0 = O_G(\mathbf{I}),$$

where  $\mathbf{I} \in \mathbb{R}^{m \times n \times c \times k}$  is a set of rectified images of size  $m \times n$  with  $k$  observed viewing angles and  $c$  color and

depth channels (1 for black and white, 3 for RGB and 4 for RGB with depth). Additional parameters are required for specific implementations, as described in Sec. 5.1.1. Our configuration algorithm can either initialize with a single shot pose estimation algorithm or, for systematic evaluation, a randomized initialization, where the degree of randomization is controlled via a parameter.

#### 3.2 Hypothesis Refinement

After a successful initial hypothesis generation, we offer a set of pose refinement operators. Such operators take a pose and infer an optimized pose hypothesis vector  $h$  by applying an operator  $O_R$  at the iteration  $t$  as follows:

$$h_{t+1} = O_R(h_t, \mathbf{I}, \rho),$$

where the extrinsic camera parameters  $\rho \in \mathbb{R}^{6 \times k}$  and the set of rectified images  $\mathbf{I}$  are required as input.

#### 3.3 Hypothesis Scoring

Our hypothesis scoring operators  $O_S$  take a 6D pose  $h_t$  as well as the input image stack  $\mathbf{I}$  and infer the accuracy of the current estimation by returning a score value  $\Psi$  as follows:

$$\Psi = O_S(h_t, \mathbf{I}).$$

## 4 CONFIGURATION ALGORITHMS

In order to combine the created building blocks (described in Sec. 3), we model suitable configuration algorithms (in the literature often referred to as agents). For comparison purposes we model several configuration algorithms to demonstrate their performance. As a baseline model we design an agent to take a random choice from a given set. Other used approaches (e.g. policy gradient, evolutionary method, simple agent) have more complex interaction patterns. In the following we will first define terms and approaches used throughout the paper, then describe our configuration algorithms in more detail.

The learning process  $(\mathcal{S}, \mathcal{A}, f, r)$  consists of a state space  $\mathcal{S}$  described as a vector, a discrete action space  $\mathcal{A}$ , a state transition function  $f: \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ , to map a state  $s_t \in \mathcal{S}$  at iteration  $t$  to the next state  $s_{t+1} \in \mathcal{S}$ . A reward  $R: \mathcal{S} \mapsto r$  is followed by each action. We define the trajectory (sequence of states  $s$ , actions  $a$  and rewards  $r$ ) as:

$$\tau = \{s_t, a_t, r_t\}_{t \in \{t_0, \dots, t_H\}}, s_t \in \mathcal{S}, a_t \in \mathcal{A}, r_t \in \mathbb{R}, \quad (1)$$

where  $t_0$  denotes the first iteration and  $t_H$  the terminating event (e.g. terminal iteration with pick operation). Our goal is to maximize the reward:

$$\max_{\tau} R(\tau),$$

by finding the optimal trajectories.

In our operative environment, the configuration algorithm receives a state input, which is a vector of scores, at every iteration  $t$ . These scores are determined by various algorithms to evaluate the current situation (e.g. depth accuracy, position accuracy of the current 6D-position estimate). The agent learns an interpretation of this vector and infers the optimal next step using a policy. A neural network can function as such a configuration algorithm in order to infer the appropriate next choice (best expected result in terms of accuracy and computation time for the current score values). Our implemented configuration algorithms are described in the following.

#### 4.1 Random Agent

A random agent is a basic approach which takes a uniformly distributed random choice under all possible actions. The agent receives a set of  $c$  action choices at iteration  $t$ :

$$a_t = (a_{t,k})_{k \in \{1, \dots, c\}} \in \mathbb{N}^c. \quad (2)$$

We choose an action by determining  $k$  randomly (from a uniform distribution). The action  $a_{t,k}$  results in a reward  $r_t \in \mathbb{R}$ .

#### 4.2 Simple Agent

A simple agent has a basic memory system and chooses either a random action or the best memorized one. Just as the random agent, this agent also receives a set of  $c$  action choices at iteration  $t$  (see Eq. (2)). The action is chosen by a  $\xi$ -weighted choice of  $k$ . With a probability of  $p = \xi$ , where  $\xi \in \{0, \dots, 1\}$  is a uniform distributed random number, the action element with the index  $k$  is chosen. With a probability of  $p = 1 - \xi$  the best memorized value is taken. The memory is updated for all possible choices at each iteration  $t$  by the received reward. This is achieved using a running average:

$$m_{at} = \frac{1}{n_{a_t}} \cdot (m_{at-1} \cdot (n_{a_t} - 1) + r_t),$$

where  $n_a$  denotes the current number of update calls for each action. Note with  $\xi = 1$  the simple agent would behave as our random agent.

### 4.3 Policy Gradient Reinforcement Learning

With a policy gradient reinforcement learning method, we utilize a learning approach, where the agent (policy network) takes a state vector  $s_{t,k} \in \mathbb{R}$ , where  $k \in \{1, \dots, n\}$  represents the  $n$  input elements, and returns a probability distribution over actions  $P(\mathcal{A}|\mathcal{S})$ . We are sampling this probability distribution to retrieve our next action. The state  $s_{t,k}$  holds a score, which reflects the quality of the current state. Our neural network model consists of two linear layers, the first followed by a tanh activation function and the second by a softmax function.

### 4.4 Evolutionary Reinforcement Learning

We also adapt an evolutionary strategy, where we are maximizing the fitness of a set of  $n$  agents  $G_{t,i}, i \in \{1, \dots, n\}$ , at iteration  $t$ , by maximizing the reward  $r$  of the agent over the episode length  $l$ :

$$\max_{r_{G_{t,i}}} \frac{1}{l} \sum_{j=1}^l r_{G_{t,i,j}}.$$

An agent with superior neural network weights (large  $r_{G_{t,i}}$ ) bears better traits to solve the task and will show high performance in the given environment (e.g. picking shiny objects from an unordered bin). The best performing agents are used to create new populations of neural networks, by breeding and mutating. We choose the best two agents to breed the next population of agents  $G_{t+1,i}$  by  $n$  random combinations of their weights. More specifically, weights are either picked from the first or the second best agent, by an equal division of 50% for each. It is randomly determined which weights are chosen from which agent:

$$\mathcal{X} \left( 1, \dots, \frac{n}{2} - 1 \right) w_{r_1} + \mathcal{X} \left( \frac{n}{2}, \dots, n \right) w_{r_2},$$

where  $\mathcal{X} \in \mathbb{N}^{k,m}$  is a matrix of randomized index values of the size  $k \times m$  of the best performing weights  $w_{r_1}$  and  $w_{r_2}$ . The neural network for each agent consists of three linear layers, the first two are followed by a tanh activation function and the last by a softmax function.

## 5 EXPERIMENTAL BUILDING BLOCKS

For our experimental evaluation we implemented and utilized two base types of building blocks. First, algo-



rithmic building blocks for 6D pose hypothesis generation, refinement and hypothesis scoring. Second, dataset blocks, which provide access to both benchmark datasets as well as our real world dataset collection.

## 5.1 Algorithmic Building Blocks

In the following we describe the implementation of our algorithmic building blocks. These formally modeled blocks are implemented in a way to enable a standardized interaction. Each block belongs to one of the following three categories: hypothesis generation, hypothesis refinement or hypothesis scoring. Hypothesis generation algorithms generate initial 6D pose hypotheses using object models as well as input image data (e.g. RGB image + depth image). Hypothesis scoring algorithms take a 6D pose hypothesis as well as input image data and return a value, which reflects the estimated accuracy of the current position. Hypothesis refinement algorithms take such initial object poses and refine them, utilizing optimization algorithms and taking scoring algorithms into account.

### 5.1.1 Hypothesis Generation

In this section we describe a set of utilized algorithms which generate an initial hypothesis  $h_0 \in \mathbb{R}^6$ , as introduced in Sec. 3.1.

**Single Shot Pose Estimation Algorithm.** The algorithm, presented in (Tekin et al., 2017), predicts the 6D pose of an object from an input image using an end-to-end CNN (Convolutional Neural Network) architecture.

To achieve this, we apply an operator  $O_{G_S}$  at the first iteration on a set of rectified images  $\mathbf{I} \in \mathbb{R}^{m \times n \times k}$ :

$$h_0 = O_{G_S}(\mathbf{I}, \rho).$$

Where  $k$  defines the viewpoints and the image grid size is  $m \times n$ . Extrinsic camera parameters are defined as  $\rho \in \mathbb{R}^{6 \times k}$ , consisting of translation and rotation from the camera to the world coordinate system.

This method can handle occlusions and allows for real-time processing. Fast computational speeds can be achieved due to omitting the need for additional post-processing, which most other algorithms of this category require. This algorithm is using a pre-trained network. For tailoring it to a specific dataset or set of objects, another training sequence is suggested for updating the weights.

**Simulated Initialization.** For strategic evaluation purposes we implemented a simulated initialization

for our generated hypothesis  $h_0 \in \mathbb{R}^6$ . The simulated initialization algorithm which takes our ground truth hypothesis:

$$\hat{h} = (\hat{h}_p, \hat{h}_o)$$

as input and adds random Gaussian noise on the position  $\hat{h}_p$  and orientation  $\hat{h}_o$  components. The degree of randomization can be varied systematically with the parameter  $\lambda \in \mathbb{R}^+$ . Hence our operator  $O_{G_I}$  is called at the first iteration as follows:

$$h_0 = O_{G_I}(\hat{h}, \lambda),$$

where  $\lambda = 0$  initializes with the given ground truth.

### 5.1.2 Hypothesis Refinement

In this section we describe a set of utilized hypothesis refinement algorithms which function as such operators  $O_R$ , as introduced in Sec. 3.2.

**ICP Refinement.** First, we model an Iterative Closest Point (ICP) algorithm as presented by (Rusinkiewicz and Levoy, 2001). ICP is a well-established (Arun et al., 1987) procedure for iteratively fitting 3D models and 3D point clouds when an initial hypothesis prediction exists. We formulate the open source implementation from Open3D (Zhou et al., 2018) as a building block, which we will call PCD-ICP (Point Cloud ICP).

**D2CO Refinement Methods.** Then, we utilize the Direct Directional Chamfer Optimization (D<sup>2</sup>CO) method, presented in (Imperoli and Pretto, 2015), which refines 3D object positions. It processes gray-level images and promotes using a 3D distance called Directional Chamfer Distance (Liu et al., 2012). The method targets handling textureless and partially occluded objects at a high processing speed while not requiring an offline learning step. To achieve this, it is employing non-linear optimization procedures.

The D<sup>2</sup>CO framework (Imperoli and Pretto, 2015) additionally provides a set of comparison algorithms. The set comprises the Directional Chamfer Distance ICP (DC-ICP), Simple Chamfer Matching optimization, an ICP implementation that exploits the Chamfer Distance (C-ICP) and a direct optimization procedure (a simple coarse-to-fine object registration using Gaussian pyramids of gradient magnitudes images). DC-ICP showed state of the art results but required many iterations to converge and hence is quite slow. Within this framework, other algorithms (such as LM-ICP or direct optimization) were shown to perform weaker but with a significant speed improvement. C-ICP showed a lower registration rate as well as a high computational time for the tested objects.

**Depth Refinement.** Additionally, we implemented a depth refinement algorithm, which was first mentioned in (Dietrich et al., 2019). It is taking a given pose hypothesis  $h_t$  and refining it along the ray from the optical frame to the pose hypothesis. The amount of the shift is defined by the distance between the input depth image and the rendered depth image.

In different environments (objects can be e.g. shiny, matte, symmetric, unordered), our hypothesis refinement building blocks can exhibit varying strength and weaknesses, which affect their performance. We show that combining such algorithmic building blocks in a strategic way allows for the utilization of the strength of each algorithm in an optimal way. This combination leads to a superior 6D pose estimation result for a given scene and state. To achieve this, we set up and implement configuration algorithms in order to learn the best combination dynamically for various environments.

### 5.1.3 Hypothesis Scoring

In this section we describe a set of utilized algorithms which generate scoring values for a current 6D pose hypothesis, as introduced in Sec. 3.3

**D<sup>2</sup>CO Scoring.** This method (as described in Sec. 5.1.2) comes with a scoring function, which is based on local image gradient directions. It uses an L1 penalty and allows for outliers.

$$\Psi_C = \frac{1}{l} \sum_{p=1}^l |\cos(G_{I_p} - N_{J_p})|$$

Here,  $I_p$  denotes the gray scale image at the position  $p = (x, y)$  on a discretised surface with a size of  $m \times n$ ,  $l$  defines the number of cloud elements, and  $J_p$  denotes the projected point cloud element. The gradient of the image is defined as:

$$G_{I_p} = (G_{I_{p,x}}, G_{I_{p,y}}, 1)$$

while the normal direction  $N_J$  is calculated for the projected point cloud  $J_p$  with:

$$N_{J_p} = (N_{J_{p,x}}, N_{J_{p,y}}, N_{J_{p,z}}).$$

**Depth Scoring.** We implement a depth scoring metric  $\Psi_D$ , which is combining two depth scores:

$$\Psi_D = (\mathcal{M}_1 + \mathcal{M}_2).$$

For the computation of the score a depth image of the object hypothesis is rendered and compared with the input depth image. The operator computes the following scores:



(a) LM

(b) Our industrial dataset

Figure 3: Dataset examples from the BOP Benchmark (Hodaň et al., 2020) (LM) and our industrial dataset (covers). The latter depicts an example of one of our typical industrial dataset types.

- $\mathcal{M}_1$ : Percentage of measured points on the object with a depth distance between the depth map  $D$  to the given depth map  $\hat{D}$  lower than a parameterizable threshold (0.005 meter).
- $\mathcal{M}_2$ : Percentage of measured NaN points on the depth map  $D$  of the observed object.

This metric computes a likelihood of  $D$  holding accurate depth values.

### 5.1.4 Pick Operator

In our simulated environment, we additionally implemented a “pick” operator. This operator measures the distance of the current 6D pose to the 6D ground truth pose. If the distance is below a defined threshold of accuracy, the pick is defined as “carried out successfully”, otherwise it failed. This threshold was defined from experience and set to a value so that only a small deviation from the ground truth is allowed both in position and rotation. In a real world scenario the success of the “pick” is defined by whether the object was grabbed and held by the robotic arm.

## 5.2 Dataset Blocks

We combine our algorithmic building blocks (Sec. 5.1) using the previously defined configuration algorithms (Sec. 4). Our goal is to find the optimal trajectory  $\tau$  (Eq. 1) for our robotic picking application. Note that the trajectory  $\tau$  defines our sequence of perception operators. For every scene, a less time intensive trajectory, which ends with a successful pick operation is superior to a trajectory, which consumes more time until the successful terminal operation. For this evaluation we are using both, datasets with ground truth information from the BOP benchmark as well as self-acquired real world datasets. In the following we describe the implementation of our dataset building blocks. By formally modeling our dataset blocks we enable a standardized interaction.

### 5.2.1 BOP Benchmark

The BOP benchmark provides scenes for 6D pose estimation, where multiple objects are present. The benchmark contains 11 public datasets which come with RGB-D images, 3D object models, 6D ground truth object poses as well as intrinsic camera parameters. Additionally, the datasets comes with a toolbox, which can be utilized to handle the data. We are using the Linemod (*LM*) benchmark, illustrated in Fig. 3a, to evaluate our setup and algorithms. It contains 15 different scenes with a total of 18273 images, comprising textureless as well as industry-relevant objects, which are two important categories for our industrial robotic picking task. We are using this dataset for evaluating our configuration system.

### 5.2.2 Our Real World Dataset

This dataset was acquired in our lab. It consists of several acquisitions of covers (see Fig. 3b), which are placed in a bin. The dataset was acquired with diffuse illumination and a roboception visard 160 color camera and consists of 436 acquired scenes. Throughout the project we used this dataset for internal quantitative and qualitative comparison of the standardized BOP benchmark with our industrial real world acquisitions. The ground truth was manually annotated.

## 6 EXPERIMENTAL RESULTS

We evaluate the use of our formally modeled building blocks (see Sec. 3) by an configuration algorithm (see Sec. 4). To achieve this, we are utilizing the defined datasets (see Sec. 5.2) to demonstrate the performance of our proposed algorithms. Our dynamic framework for procedural knowledge (building blocks) plays an integral part by enabling a structured use and providing clear interfaces to our algorithmic models.

### 6.1 Setup

Our test setup takes a specific input dataset and initializes the first hypothesis  $h_0$  with a hypothesis generation algorithm. At each iteration  $t$ , the configuration algorithm gets a set of possible actions  $a$  and chooses from that set (as described in Sec. 4). If the configuration algorithm arrives at the pick action and is successful, the process is terminated. We measure success in time until the successful pick for each test-case. A failing pick operation would result in a return of a negative time penalty reward and the algorithm continues with the next chosen operator.

We are splitting the LM BOP dataset in two disjoint test- and training-sets, for all algorithms which have a training sequence included. All image IDs are randomly assigned to one of the two sets.

### 6.2 Evaluations

Quantitative evaluations are presented using the LM samples from the BOP Benchmark dataset (see Sec. 5.2.1). We compare four configuration algorithms (random agent, simple agent, policy gradient and evolutionary approach) using three metrics (number of operations, computational time, % of successful first pick operations), as shown in Fig. 5. All evaluations are carried out using an increasing noise level  $\lambda$  for the initial hypothesis generation operation. Specifically, we used noise levels up to  $\lambda = 1.0$  (where  $\lambda = 0$  would evaluate the initialization with the ground truth position), illustrations of initialization examples with each noise level are shown in Fig. 4. Note, that the noise both on the position and orientation is normally distributed and varies in extend by this random factor. In the following we describe each metric and discuss the results.

#### 6.2.1 Number of Operations

The number of operations a configuration algorithm requires before a successful pick operation (object was localized with the specified precision around the ground truth, such that a pick would be possible) is a time-independent quality measure. Ruling out the optimization of the computation time of each algorithm, this measure focuses solely on the number of steps (hypothesis refinement algorithm calls), which were required to reach a successful pick operation.

An evaluation is shown in Fig. 5a, where the quartile range of the result values on the dataset at each observation ( $\lambda$  noise values) is indicated with a vertical line. The mean value is at the position of the interception with the dotted line (interpolated values between  $\lambda$  level observations). A configuration algorithm which takes fewer operations to arrive at a successful pick operation is considered superior. Over all noise levels, the random agent showed the worst results. This serves as a baseline for our other algorithms. The simple agent performed significantly better in situations with low noise, where few operation steps are sufficient, but increasingly worse with higher  $\lambda$  values. The evolutionary reinforcement learning strategy showed stable, but not optimal, results over different noise levels. Because of the high computational efforts (several days of execution) of that algorithm, we used a small number of agents and



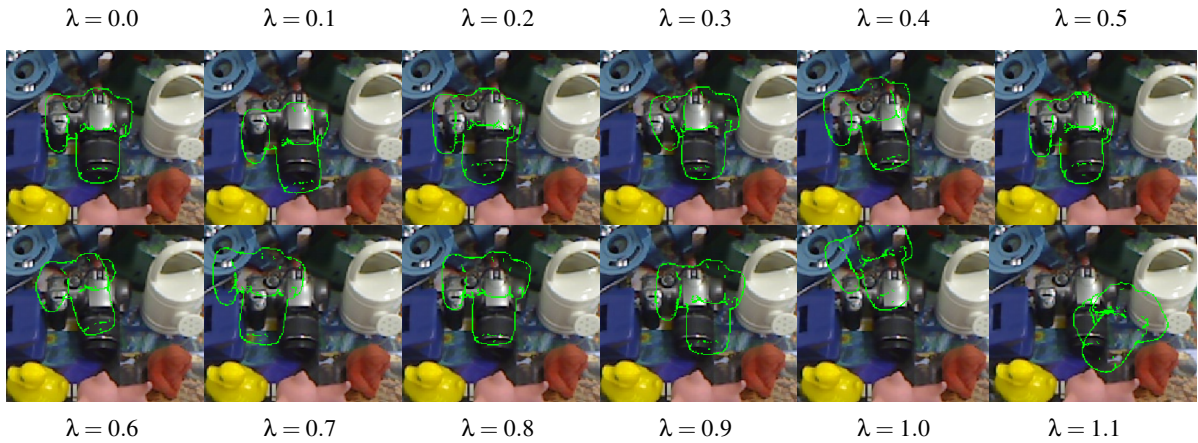


Figure 4: Examples of randomized noise levels from no noise ( $\lambda = 0.0$ ) to noise level  $\lambda = 1.1$  (as described in Sec. 5.1.1). Noise is added on the 3-dimensional rotation and translation of the object position.

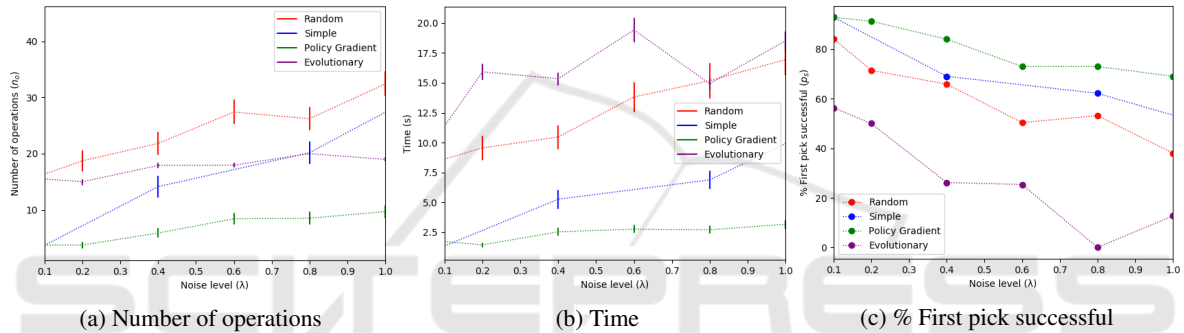


Figure 5: Comparison of methods on the BOP LM dataset (see Sec. 5.2.1). Three types of metrics (see Sec. 6.2) were used with noise levels  $\lambda \in \{0.1, \dots, 1.0\}$  (see Sec. 5.1.1).

iterations. This can be overcome by optimizing / parallelizing algorithmic components.

### 6.2.2 Computation Time

In Fig. 5b we evaluated the algorithms by the total computation time required to reach a successful pick averaged over all samples. The graph type is the same as described in Sec. 6.2.1, where the quartile range for each configuration algorithm and  $\lambda$  noise value is indicated by a vertical line. The evolutionary reinforcement learning strategy shows the worst computational performance, followed by the random agent. While the simple agent shows a better performance over all noise level, the best performing strategy in total time until a successful pick per sample is the policy gradient algorithm. Note, that the configuration agent takes the computation time into account, as the time is returned as a negative reward component of the reward created for each action. Future work will cover the improvement of computation time through parallelization as well as optimal data handling.

### 6.2.3 Successful First Picks

This metric evaluates how often a running algorithm is correctly choosing the “pick” operator, when chosen the first time in the sequence. Consider our operational elements: hypothesis generation ( $a_{gen}$ ), hypothesis refinement ( $a_{ref}$ ) and pick operator ( $a_{pick}$ ). A configuration algorithm which chooses the following sequence (0 indicating that the simulated pick failed and 1 a successful pick operation):

$$\{a_{gen}, a_{ref}, a_{pick} \rightarrow 0, a_{ref}, a_{pick} \rightarrow 1\}$$

would arrive at a successful pick after 5 operations (value 5 in “number of operations”, as shown in Fig. 5a). In the metric of successful first picks, it would receive the value 0. Contrary, this sequence would receive the value 1 (with value 5 in “number of operations”):

$$\{a_{gen}, a_{ref}, a_{ref}, a_{ref}, a_{pick} \rightarrow 1\}.$$

We are expressing the results as percentage of successful first pick operations. The evaluation is shown in Fig. 5c, where a higher value indicates a better

performance. Note that, although the overall performance of the evolutionary reinforcement algorithm is superior to the random agent, it shows fewer successful first picks. This indicates, that a higher improvement can be achieved by allowing for more agents and iterations (which requires a compensation of the run-time by parallelization). The simple agent shows a better successful pick performance than the random agent over all noise levels. The best performing method in this metric is our policy gradient agent.

We evaluated the performance of all three metrics additionally on our industrial dataset, which showed comparable results in all categories.

## 7 CONCLUSIONS

Industrial production processes have a continuously increasing need for flexible and dynamic robotic solutions. Today, this often requires long and tedious configurations by well-trained engineers. Such processes are both costly and time consuming. We target this problem by learning optimized solutions, applicable for a wide range of industrial tasks and environments. As an applicable precedent for such industrial robotic engineering tasks, utilize the field of robotic picking.

We demonstrated our systematic approach of formulating procedural knowledge in building blocks and creating standardized interfaces. We evaluated specific configuration algorithms which were tasked to choose such building blocks in an optimal order. This was enabled by another standardized interface for learning algorithms, namely the utilization of the OpenAI Gym interface.

We showed, that an improvement of performance with respect to a random or simple approach (as could be performed by an engineered pipeline) can be achieved for the task of 6D pose estimation for industrial robotic picking for various scenarios (datasets). From all evaluated configuration algorithms, the policy gradient approach achieved the most superior performance. We successfully demonstrated the general feasibility of our approach on the public bop-benchmark.

We demonstrated the setup and use of configuration algorithms and formally modeled building blocks, both utilizing standardized interfaces (the OpenAI Gym interface and a framework for hierarchical modeling respectively). This standardization enables the dynamic connection of a wide range of formally modeled building blocks and configuration algorithms. The more elements are available through these frameworks, the more powerful our solution be-

comes. This will allow for a dynamic adaption to a vast range of environments and objects with complex shapes, surface reflection behaviors and textures. Hence, one aspect of future work will lie in dimensional scaling, such that our system holds numerous elements (building blocks and configuration algorithms). Other aspects of future work will cover the improvement of computational time of different algorithmic components or the increase of computational power (e.g. by parallelization, using services such as server clusters), enabling the evaluation of a wider range of learning algorithms, as well as the transfer of our algorithmic ideas to different areas of industrial robotic applications.

## ACKNOWLEDGEMENTS

This work was carried out within the Siemens AI Lab Residency Program.

## REFERENCES

- Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym.
- Chen, G., Han, K., Shi, B., Matsushita, Y., and Wong, K.-Y. K. (2020). Deep photometric stereo for non-lambertian surfaces.
- Choi, C. and Christensen, H. I. (2016). Rgb-d object pose estimation in unstructured environments. *Robotics Auton. Syst.*, 75:595–613.
- Dietrich, V., Kast, B., Fiegert, M., Albrecht, S., and Beetz, M. (2019). Automatic configuration of the structure and parameterization of perception pipelines. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 312–319.
- El-Shamouty, M., Kleeberger, K., Laemmle, A., and Huber, M. (01 Nov. 2019). Simulation-driven machine learning for robotics and automation. *tm - Technisches Messen*, 86(11):673 – 684.
- Hailin Jin, Soatto, S., and Yezzi, A. J. (2003). Multi-view stereo beyond lambert. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I.
- Hodaň, T., Sundermeyer, M., Drost, B., Labbé, Y., Brachmann, E., Michel, F., Rother, C., and Matas, J. (2020). BOP challenge 2020 on 6D object localization. *European Conference on Computer Vision Workshops (ECCVW)*.

- Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization, LION'05*, page 507–523, Berlin, Heidelberg. Springer-Verlag.
- Hutter, F., Kotthoff, L., and Vanschoren, J., editors (2018). *Automated Machine Learning: Methods, Systems, Challenges*. Springer. In press, available at <http://automl.org/book>.
- Imperoli, M. and Pretto, A. (2015). D<sup>2</sup>CO: Fast and robust registration of 3D textureless objects using the Directional Chamfer Distance. In *Proc. of 10th International Conference on Computer Vision Systems (ICVS 2015)*, pages 316–328.
- Kast, B., Dietrich, V., Albrecht, S., Feiten, W., and Zhang, J. (2019). A hierarchical planner based on set-theoretic models: Towards automating the automation for autonomous systems. In Gusikhin, O., Madani, K., and Zaytoon, J., editors, *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2019 - Volume 1, Prague, Czech Republic, July 29-31, 2019*, pages 249–260. SciTePress.
- Kleeberger, K., Bormann, R., Kraus, W., and Huber, M. F. (2020). A survey on learning-based robotic grasping. In *Current Robotics Reports*, page pages239–249.
- Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.
- Liu, M.-Y., Tuzel, O., Veeraraghavan, A., Taguchi, Y., Marks, T. K., and Chellappa, R. (2012). Fast object localization and pose estimation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, 31(8):951–973.
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110.
- Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152.
- Song, K.-T., Wu, C.-H., and Jiang, S.-Y. (2017). Cad-based pose estimation design for random bin picking using a rgb-d camera. *Journal of Intelligent and Robotic Systems*, 87.
- Sun, M., Kumar, S., Bradsky, and Savarese, S. (2011). Toward automatic 3d generic object modeling from one single image. In *3DIM-PVT*.
- Tang, J., Miller, S., Singh, A., and Abbeel, P. (2012). A textured object recognition pipeline for color and depth image data. In *ICRA*, pages 3467–3474. IEEE.
- Tekin, B., Sinha, S. N., and Fua, P. (2017). Real-time seamless single shot 6d object pose prediction. *CoRR*, abs/1711.08848.
- Wanner, S. and Goldluecke, B. (2013). Reconstructing reflective and transparent surfaces from epipolar plane images. In Weickert, J., Hein, M., and Schiele, B., editors, *Pattern Recognition*, pages 1–10, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Yeh, T., Lee, J. J., and Darrell, T. (2009). Fast concurrent object localization and recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287.
- Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.