# Easy grasping location learning from one-shot demonstration

Laurent Bimont, François Helenon, Eric Nyiri, Stéphane Thiery, Olivier Gibaru

# Easy grasping location learning from one-shot demonstration

Laurent Bimont, François Hélénon, Eric Nyiri, Stéphane Thiery and Olivier Gibaru
LISPEN laboratory, Arts et Métiers, Lille, France.
Corresponding author: laurent.bimont@ensam.eu

***Abstract*— In this paper, we propose a fast learner grasping pipeline able to grasp objects at a specific location few minutes after being taught by an operator. Our motivation is to ease reconfiguration of robot according to a specific task, without any CAD model, nor existing database, nor simulator. We build a CNN pipeline which performs a semantic segmentation of object, and recognizes authorized and prohibited grasping location shown during demonstration. For that we have simplified the input space, created a data augmentation process and proposed a light CNN architecture allowing learning in less than 5 minutes. Validation on a real 7-DOF robot shown good performances (70 to 100% depending on the object), with only a one-shoot operator's demonstration. Performances remain good when grasping similar unseen objects, and with several objects in the robot's workspace using few demonstrations. A video highlighting the main aspects can be found at https://www.youtube.com/watch?v=rYCIk6njBo4**

## I. Introduction

As part of Industry 4.0, collaborative robots are setting up in the factories. They aim to work alongside workers, helping them by carrying heavy loads or performing repetitive tasks. Robotic manufacturers like Kuka or UR have made them easily reconfigurable for simple and deterministic tasks. Thus, these robots are adapted from large companies to small ones, and from large series to mass customisation. The expected benefits are: more attention of the operator on complex tasks, a reduction in repetitive gestures, and an increase in substation productivity. However, for non-deterministic tasks (e.g. when the position is unknown), the use of a robotics engineer is necessary to redeploy the robot, requiring long working hours.

In this paper, we investigated the problem of an operator willing to reconfigure a robot to grasp an unreferenced industrial object at a specific location. This fall under the wide umbrella of task-oriented grasping, different use cases are possible: a) grasping a tool by the handle or head depending on the task the robot has to perform b) grasping an industrial object so that it can be placed in a chosen orientation as part of a pick and place operation or c) grasping a fragile object by a safe location. The use of Convolutional Neural Network (CNN) partly solves this problem since they offer very good results for object recognition and grasping [1] [2] [3]. However, they are accompanied by constraints such as the use of specific databases (Cornell database [4]) or a high training time [3]. Our motivation is to propose a fast learner grasping system, without any need of a database, or CAD model, or simulator, so that it can be easily reconfigure by the operator itself.

Grasping in robotic has been studied for many years resulting in the development of various techniques. Recently, deep learning tools enable the rise of object agnostic grasper following a data-driven approach [5]. We can distinguish two main approaches: 1) the use of labelled data where grasping location are carefully annotated on large amount of images [1] [6], 2) the use of trial and error to learn how to grasp, based on a simulator or a real world environment [3] [7]. The first technique is not satisfactory when no dataset meeting a specific needs is available, and the second is time consuming or can damage fragile objects while learning. To overcome those limitations, the learning from demonstration paradigm is a promising field.

Learning from demonstration can be applied to initialize an object-agnostic grasper to a close form solution and then improving it by trial and error [8] [9]. However, the trial and error stage is a limitation of those methods in an industrial context.

Our contribution is a fast learner grasping system able to achieve relevant task-based grasping from one shot demonstration without a trial and error phase.

## II. Related work

This work gathers task-oriented grasping technique with learning from few demonstrations. In Table I, we provide a brief summary of recent task oriented grasping works.

**Task oriented** uses the concept of affordance introduced by Gibson [15] which describes parts of objects according to their functional utility. In robotics, this concept is used for gripping and handling objects considering the work to perform after [16] [17]. Task oriented grasper can be created using behaviour grounded affordance [11] or spatial maps [16] for instance. Semantic labels technique on images can also be applied [18] [10] [19]: using specific large datasets such as UMD [13] or shape database [12], each pixel is labelled independently according to the part of the object to which it belongs. Our work uses semantic labelization of image without databases, leveraging few examples to learn.

**Demonstration learning** can be used to transfer knowledge from an operator to a system, in our case to teach a robot a precise grasping location. Most of the works address this problem with a trial and error phase through a simulator or directly on the real system. In [20], the authors propose a network architecture and data augmentation pipeline to design a controler to grasp very simple objects (cube, cylinders...) from a single demonstration. However, the controler can not integrate important constraints in task oriented grasping

TABLE I: Brief summary of existing task oriented grasping works

| Ref | Data generation | CAD model & simulator | Observation |
|---|---|---|---|
| [10] | From CAD model | Yes | Training took 6 hours on Titan X GPU |
| [11] | From simulation | Yes | 1.5 M of data are generated for training |
| [12] | ShapeNet and ModelNet40 | Yes | Bayesian Optimisation |
| [13] | RGB-D Part Affordance Dataset | No | - |
| [14] | From few views | No | 20 minutes to reconstruct 3D mapping of object |
| ours | From few demonstration | No | $<$ 5 minutes of training on RTX 2080 GPU |

like prohibited location. Closer to our work, [14] learns a dense descriptors map for objects after building a 3D dense reconstruction model of the object. As a result, they obtain a semantic representation of the object allowing to grasp at the desired location. We have decided to work directly on images without any 3D reconstruction technique.

Different methodologies can be used to **train a CNN with few data**: decrease the input space size, use data augmentation or/and use transfer learning to initialize the network. In [21] [22], the authors reduce their input space size by using only depth from the RGB-D camera. Data augmentation techniques increase the size and variance of the training set, whereas transfer learning techniques accelerate the learning process. For transfer learning, parts of pretrained image classification architecture are used to initialize CNN. The very rich ecosystem of image classification research provides access to a lot of high-performance architectures (VGG [23], Resnet [24], Densenet [25]), pretrained on large image databases (ImageNet [26], coco [27]). Despite being trained on RGB images, we can use those architectures for the creation of CNN processing depth images. In the grasping domain, many works have used this technique to create grasping predictor [28] [29], and achieve up to 89.9% of successful grasping on the Cornell database [1].

## III. Method

We consider the problem of performing an antipodal grasp perpendicular to a planar surface, on known object for which an operator has taught authorized/prohibited grasping location. A RGB-D camera is mounted on the robot's wrist and capture a fixed height top view.

Based on grasping from few demonstrations method, we want to define a pixel wise semantic segmentation pipeline where the input is a depth image of the scene and the output is $\mathbf{g} = (x, y, z, \theta)$ for grasping the object on the demonstrated location. Coordinates $(x, y, z)$ represent the tool center of the gripper, and $\theta$ is the angle of the gripper in the plan. Fig. 1 describes our problem definition. We define a structure of our pipeline allowing a fast training from few demonstrations. This problem imposes constraints that motivated the design of our pipeline: a) learn fast authorized/prohibited grasping locations and b) generalize from few demonstrations.

### A. Inputs & Outputs

**The input** is the 2D shape of object(s) laying on the table, obtained from binarization of a depth image: table's pixels are set to 0 and objects' pixels (above the table) are set to 1. We denote it as $I \in \{0, 1\}^{n \times m}$ where $(n, m)$ is the image

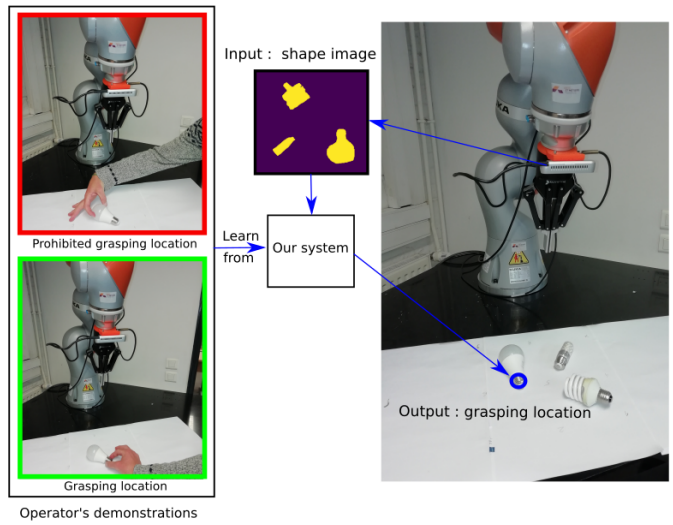


Fig. 1: First, for a new object, the system learns from operator's demonstration. After few minutes of training, the system will be able to retrieve the demonstrated location on a depthmap.

dimensions. The use of 2D shape reduces the size of the input space and contributes to our objective of generalization.

For the **output**, a grasping affordance pixel wise representation $\mathbf{G} \in [-1, 1]^{n \times m}$ is calculated, where$-1$ stands for a prohibited grasping location and $+1$ stands for an authorized grasping location. All pixels of the table (0 in $\mathbf{I}$) are set to 0 in $\mathbf{G}$. First, we determine the highest grasping affordance pixel $(u_g, v_g) = argmax_{u,v}(G(u, v))$. Then, the grasping angle $\alpha$ in image frame is calculated by performing a PCA on a sub-region of the input centered around the grasping point. Finally, geometrical transformations based on hand eye calibration are made to convert $(u_g,\ v_g,\ \alpha)$ to $\mathbf{g}$. Therefore, the robot can directly perform a grasping action in its workspace.

### B. Training Dataset

**Data Capture** Training is done directly from demonstrations without using an external database. The operator's thumb and index fingers are covered with coloured pads so that they can be easily identified by the RGB-D camera mounted on the robot. Grasping gesture on authorized and/or prohibited location are stored by recording fingers' coordinates (Fig. 2-a)). Labels are generated as an image $L \in \{-1, 0, 1\}^{n \times m}$ with value 1 for authorized zones, 0 for zones without information and $-1$ for prohibited zones. The corresponding input $\mathbf{I}$ is captured from the scene (Fig. 2-b)). Demonstrations are collected from different angles and

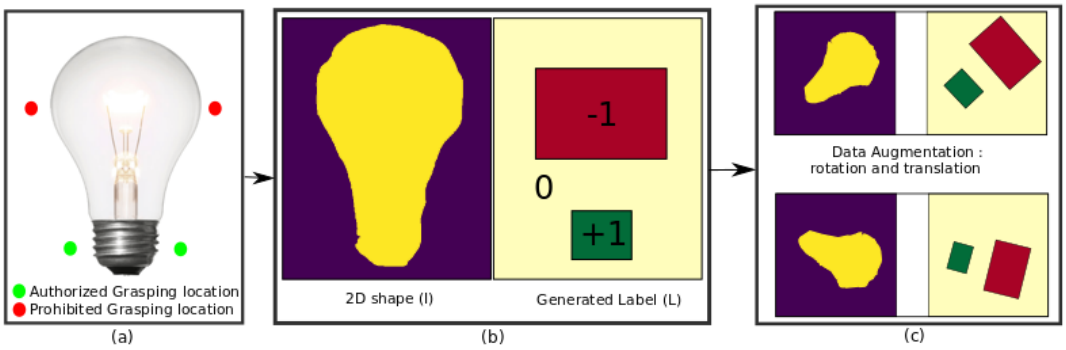


Fig. 2: Data capture and data augmentation pipeline

stored as tuple (**I**nput, **L**abel).

**Data Augmentation** The number of these tuples are then increased by random translations and rotations (Fig. 2-c)). We also erase part of the input randomly in order to give variability in the training data. Moreover it reflects that the camera output is noisy, and geometric shapes evolve with the relative position of objects according to the camera.

### C. Network pipeline

**Architecture**: to generate the output **G** from the input **I**, the mapping function is a CNN created from a part of Densenet 121 [25] and a light non-pretrained CNN. The overall architecture is presented in Fig. 3, activation functions are "RELU" except at the output where the "tanh" function has been chosen to distribute the values between $-1$ and 1. The dropout was set to $40\%$ to prevent the network from overfitting and to generalize well to unseen data. This small convolution network has only 6914 parameters and can be trained with our training dataset in a few batches using an appropriate loss function.

The **Loss function** should allow our pipeline to generalize from a few demonstrations, to classify a pixel wise representation with unbalanced area sizes (as shown in Fig.2-b), prohibited, neutral and authorized zone have different size) and to focus on important parts to learn quickly. To do this, we introduce a modified version of the pixel-wise L2-loss function [30] by multiplying each pixel error by a specific weight:

$$\mathcal{L}_{weighted-L2} = \frac{1}{n \times m} \sum_{i=1}^{n \times m} \omega_{i,label_i} (pred_i - label_i)^2 \quad (1)$$

The weighted factor $\omega_{i,label_i}$ is chosen as follows:

$$w_{i,label_i} = |pred_i| + \lambda_1 \times \frac{1}{N_{label_i}} \quad (2)$$

where $N_{label_i}$ represents the number of pixels in **L** containing label value $label_i$. The first component $|pred_i|$ of (2) is used to focus the network's attention on interesting parts by focusing the gradient descent over areas of interest. The second component is used to accentuate learning over under-represented areas of the label map by reducing the importance of large areas. The hyperparameter $\lambda_1$ balances the two components. The benefits of this weighted loss function is studied in section IV-C. To prevent overfitting, we commonly use a L2 regularization loss $\mathcal{L}_{L2\ reg}$ applied on the weights and bias of the network. The finale composite loss-function is:

$$\mathcal{L} = \mathcal{L}_{weighted-L2} + \lambda_2\ \mathcal{L}_{L2\ reg}$$

We trained the network by stochastic gradient descent, with a learning rate of $10^{-4}$, a momentum of 0.9, $\lambda_1 = 20$ and $\lambda_2 = 5.10^{-5}$.

## IV. Experiments

To evaluate our proposed algorithm, we are holding a series of experiments. We will study 5 points : 1) grasping object at the right location in different positions, 2) the benefits of our modify $L_2 - loss$ function, 3) the ability of the algorithm to generalize to unseen similar objects, 4) storing grasping locations of 2 different objects into the same network 5) performing grasp in a cluttered environment.

### A. Real World Experiment

We use a modified Python version of the Matlab Kuka sunrise toolbox [31] to control a Kuka iiwa LBR 7 DOF robot equipped with a Robotiq 2F-140 gripper. The workspace of the robot is a $30 \times 30$ cm flat square. An Intel®Realsense™ Depth Camera D415 is mounted on the wrist of the robot. Training and computation were made on a PC with a Nvidia RTX 2080 8Gb graphic cards and Intel®8 Core™ i7 9700K 3.6 GHz CPU. The implementation was done in Python 3.6 using Tensorflow 1.13. We train a network for 4 epochs with 800 tuples randomly sampled from the dataset. Between each epoch, a new dataset is generated by performing a data augmentation. On average the training last 250 seconds for one object.

### B. Grasping at the right location

We have to measure the ability of our network to find a specific grasping location learned during the demonstration phase. In Fig. 4, we present our panel of objects with the name of their grasping locations.

**Protocol:** For each object, we train a specialized network from 1 to 3 demonstrations of the same authorized grasping

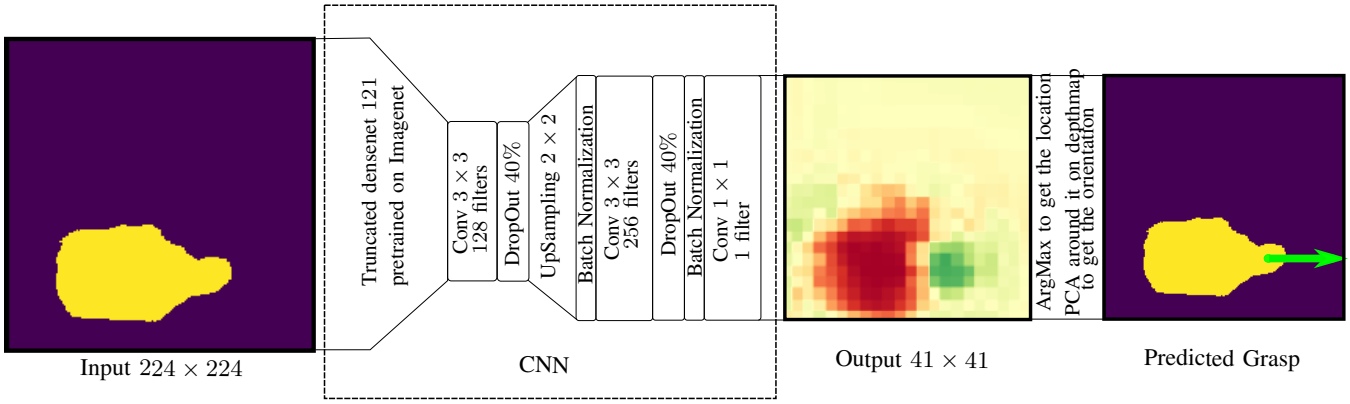


Fig. 3: Overview of our CNN pipeline

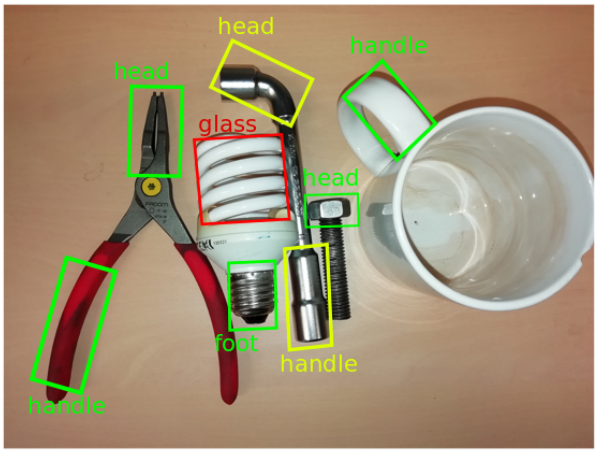


Fig. 4: Objects used for our experiment, with name of grasping locations. Green color (resp red color) denotes authorized (resp prohibited) grasping location. Yellow color is used to illustrate authorized/prohibited location and vice versa, depending on the task (for example the accessibility of the screwing operation). For the pliers, two different authorized locations are tested separately.

TABLE II: Results of our grasping test, percentage (number) of good grasps over the 36 unseen positions.

| Object | Location | Number of demonstration(s) | | |
|---|---|---|---|---|
| | | 1 | 2 | 3 |
| Socket wrench | handle | 81% (29/36) | 92% (33/36) | 94% (34/36) |
| | head | 86% (31/36) | 94% (34/36) | 94% (34/36) |
| Pliers | handle | 97% (35/36) | 100% (36/36) | 100% (36/36) |
| | head | 92% (33/36) | 97% (35/36) | 97% (35/36) |
| Bulb | foot | 100% (36/36) | 92% (33/36) | 97% (35/36) |
| Cup | handle | 70% (25/36) | 92% (33/26) | 97% (35/36) |
| Screw | head | 97% (35/36) | 100% (36/36) | 100% (36/36) |

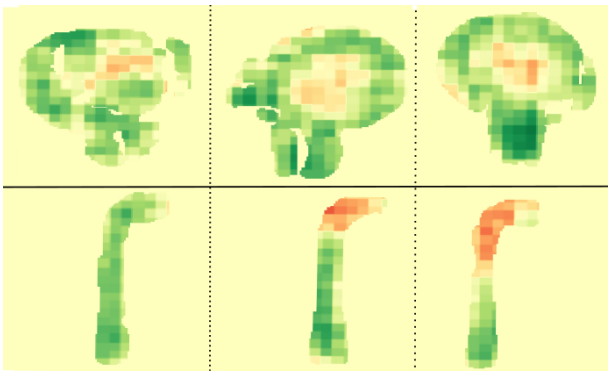

Fig. 5: Different segmentation quality results. The top images (cups) were trained showing one authorized grasping location (handle), The bottom images (socket wrench) were trained showing one authorized (handle) and one prohibited (head) grasping location. The left column shows bad segmentation resulting in bad grasping decision, the middle one shows average segmentation, and the right one shows good segmentation of the object. For the purpose of illustration, outputs where reoriented and resized. Those outputs were obtained using networks trained on 3 demonstrations.

location (with eventually a prohibited grasping location) under different object positions. Then we evaluate the network's ability to find those locations on 36 unseen positions of the object. The evaluation is done by placing the object at 9 points of the workspace and by rotating it in 4 orientations (0°, 90°, 180° and 270°). A grasp is counted as good each time the object is caught at the authorized location.

**Results & Discussion** Our pipeline achieved good results (Table II) with only one demonstration, especially when the object has geometrical differences and a simple geometry. Indeed for bulb, screw and pliers, the grasp success in the authorized location is over 90%. For socket wrench (81% and 86%) and cup (70%), the decrease in performance respectively comes from the geometric similarity of the authorized/prohibited grasping location, and a more complex geometry. Adding 1 or 2 demonstrations from other positions seems to solve that issue. For socket wrench and cup grasping, results raised over 90% of success with 2 demonstrations. In these worst cases, we suppose that data augmentation does not reproduce efficiently the different possible views of the shape of an object. Prediction quality evolves depending on the input image. In Fig. 5, we can see different cases where the system outputs good, average or bad segmentation of the object. Bad segmentation occurs when current shape is very different from the demonstrated one. It shows limitation in the generalization abilities of our system.

TABLE III: Influence of the weighted $L_2$ loss. Training was made with a set of 3 demonstrations.

| Object | Location | $L_2$ loss | weighted $L_2$ loss |
|---|---|---|---|
| Socket wrench | handle | 61% (22/26) | 94% (34/36) |
| | head | 86% (31/36) | 94% (34/36) |
| Pliers | handle | 92% (33/36) | 100% (36/36) |
| | head | 72% (26/36) | 97% (35/36) |
| Bulb | foot | 94% (34/36) | 97% (35/36) |
| Cup | handle | 75% (27/36) | 97% (35/36) |

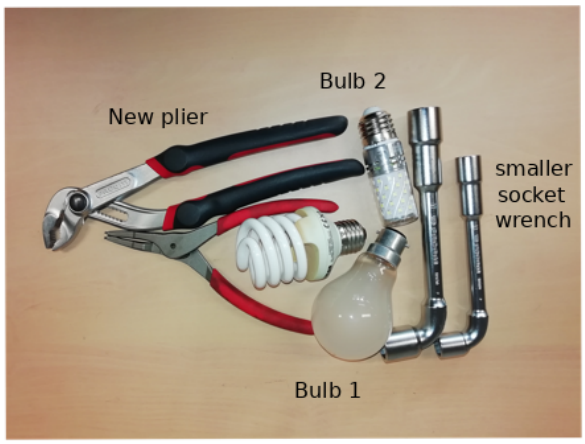


Fig. 6: Similar objects used to test our algorithm generalization abilities.

### C. Benefits of the weighted $L_2$ loss function

We measure the impact of our weighted loss function by comparing results with and without it.

**Protocol** We follow the same protocol as presented in section IV-B with the exception that we change the training method by setting all weights $\omega_{i,label_i}$ to 1 (regularization parameter $\lambda_2$ remains equal to $5.10^{-5}$). Training is performed using 3 demonstrations and results are compared with the corresponding ones of the previous experiment.

**Result & Discussion** Table III highlights that adding our weights in the $L_2$ cost function has improved the system performance in every case. Moreover, without weights, we noticed that a bad behavior could occurred: the training fails to focus on the important location ($+1$ and $-1$) and does not converge to a solution. This experiment validates the relevance of our proposed loss function.

### D. Grasping similar objects

We measure the ability of our algorithm to generalize the grasping location learned for an object to another similar one.

**Protocol** We used networks trained in section IV-B to perform the test with similar objects (Fig. 6). Similar objects were closed from the original ones but with different size or geometry.

**Result & Discussion** In Table IV, except for bulb 2, we observe a good ability of the network to generalize grasping location to other similar objects with no degradation of the performances compare to those obtained on the trained object. Taking a closer look at the different bulbs grasping affordance pixel wise representations (Fig. 7), the demonstrated knowledge is quite well transferred to bulb 1 with a precise segmentation of the unseen object. On the other hand, the shape of bulb 2 is quite different from the first bulb, and the segmentation gives an unprecise result, especially for the authorized location where grasping affordance values are low.

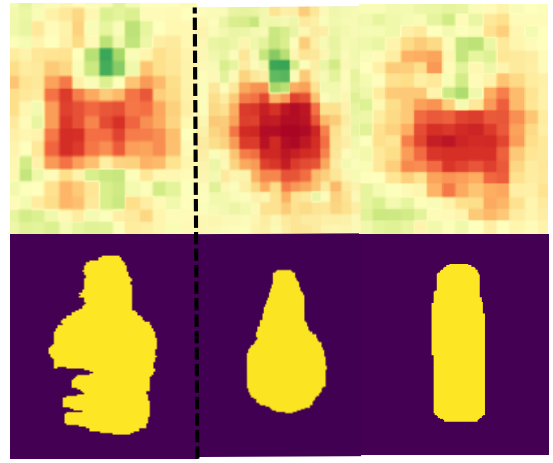

Fig. 7: Bulbs grasping affordance pixel wise representations. From left to right : bulb model on which the training was done, bulb 1, and bulb 2 .

TABLE IV: Ability of the Network to generalize to an unseen similar object. The network was trained after demonstrations on an object and the test was performed on unseen similar objects. In red, the results obtained on the object used for training are recalled

| Object | Location | Number of demonstration(s) | | |
|---|---|---|---|---|
| | | 1 | 2 | 3 |
| Smaller socket wrench | handle | 78% (28/36) 81% | 92% (33/36) 92% | 94% (34/36) 94% |
| New plier | handle | 97% (35/36) 97% | 100% (36/36) 100% | 100% (36/36) 100% |
| Bulb 1 | foot | 100% (36/36) 100% | 92% (33/36) 92% | 97% (35/36) 97% |
| Bulb 2 | foot | 89% (32/36) 100% | 78% (28/36) 92% | 81% (29/36) 97% |

### E. Store grasping locations of 2 different objects into the same network

Storing several object segmentation into the same network allows to not use a specific one for each task. We done a preliminary experiment, limited to two couples of objects, to test the ability of our system.

**Protocol** We train a network with demonstrations from 2 objects (1 to 3 demonstration(s), for each object). Then, for each object, we separately evaluate the grasping accuracy, under the 36 positions.

**Results & Discussion** We limited our experiment to two couples of objects because the performances (Table V) shows a quite limited ability of our pipeline to grasp 2 objects with 2 different grasping strategies. By working directly with the depthmap from the RGB-D camera, we probably

TABLE V: Grasping accuracy results for coupled of objects sharing the same network.

| Number of demonstrations | Plier<br>Bulb | | Bulb<br>Socket Wrench | |
|---|---|---|---|---|
| $1+1$ | 29/36<br>23/36 | 72% | 26/36<br>29/36 | 76% |
| $2+2$ | 32/36<br>21/36 | 74% | 30/36<br>35/36 | 90% |
| $3+3$ | 32/36<br>32/36 | 90% | 31/36<br>34/36 | 90% |

Fig. 8: The three groups of similar objects used in our test.

could improve the performances (however, it will increase the training time). It is a direction for future experiments.

### *F. Grasping in a clutter environment*

Grasping several identical or similar objects laying on a workspace surface is a important task in industry. It allows thereafter to place these objects in boxes for example.

**Protocol** Experiments are done on three different groups of similar objects (Fig. 8) with their corresponding specialized network trained from 3 demonstrations. For each groups, several objects are placed in the robot's workspace leaving the grasping location free of access. We let the robot grasps the objects one by one until the workspace is cleaned, when a failure occurs we remove the object manually. We repeat this process with other object's configurations until we reach 20 attempts for each group

**Results** & **Discussion**: Table VI shows performances slightly under those of individual objects. It is a promising result as our proposed algorithm, despite being trained with one object in it workspace, achieve relevant actions when several spaced object are presented to it. Few attempts to grasp objects touching each other has led to bad segmentation of objects.

TABLE VI: Grasping accuracy results for cluttered similar objects

| Bulbs | Socket wrench | Screws |
|---|---|---|
| 19/20 (95%) | 17/20 (85%) | 19/20 (95%) |

## V. Conclusion

In this work, we show that fast reconfiguration of a robot to grasp objects is possible from 1 (or more) demonstration. Furthermore, our pipeline achieves promising results to generalize to unseen similar object or to clean a workspace composed of several spaced and similar objects. Moreover, architecture does not require any dataset, CAD model or simulation. Our method combines a reduce state space, a light CNN plugged on a pretrained network and weighted loss function and then is able to quickly learn from few data. Our CNN network pipeline fulfills our initial motivation of creating a task-oriented grasping system that can be quickly reconfigure by an operator from the field.

Our work presents limitations that suggest future works. The selected input space limits our algorithm to simple 2D shapes. Working directly with the depthmap from the RGB-D camera will allow to consider more complex 3D objects. Semantic segmentation of objects may fail in some cases. Detecting these failing cases would allow the operator to be asked for more demonstrations. This work is a step in setting up collaborative robots in factories and demonstrates that operators can simply and rapidly configure them by demonstration.

## Acknowledgment

Authors would like to thank Gabriel Audry from Renault Technocentre who gave us the idea that, in industrial use cases, learning prohibited locations is as important as learning authorized ones. Authors also thank Arts et Métiers fundation, AMValor fundation and Région Hauts-de-France for supporting this research through PhD grants.

## References

[1] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," IEEE International Conference on Intelligent Robots and Systems, vol. 2017-Septe, pp. 769–776, 2017.

[2] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," IEEE International Conference on Intelligent Robots and Systems, vol. 2016-Novem, pp. 4461–4468, 2016.

[3] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," Proceedings - IEEE International Conference on Robotics and Automation, vol. 2016-June, pp. 3406–3413, 2016.

[4] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," International Journal of Robotics Research, vol. 34, no. 4-5, pp. 705–724, 2015.

[5] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis-A survey," IEEE Transactions on Robotics, vol. 30, no. 2, pp. 289–309, 2014.

[6] S. Caldera, A. Rassau, and D. Chai, "Review of Deep Learning Methods in Robotic Grasp Detection," Multimodal Technologies and Interaction, vol. 2, no. 3, p. 57, 2018.

[7] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," International Journal of Robotics Research, vol. 37, no. 4-5, pp. 421–436, 2018.

[8] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards," pp. 1–10, 2017. [Online]. Available: http://arxiv.org/abs/1707.08817

[9] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," Advances in Neural Information Processing Systems, vol. 2017-Decem, no. Nips, pp. 1088–1099, 2017.

[10] R. Detry, J. Papon, and L. Matthies, "Task-oriented grasping with semantic and geometric scene understanding," in IEEE International Conference on Intelligent Robots and Systems, vol. 2017-Septe. IEEE, 9 2017, pp. 3266–3273. [Online]. Available: http://ieeexplore.ieee.org/document/8206162/

[11] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision," 6 2018. [Online]. Available: http://arxiv.org/abs/1806.09266

[12] R. Antonova, M. Kokic, J. A. Stork, and D. Kragic, "Global Search with Bernoulli Alternation Kernel for Task-oriented Grasping Informed by Simulation," no. CoRL, 2018. [Online]. Available: http://arxiv.org/abs/1810.04438

[13] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," Proceedings - IEEE International Conference on Robotics and Automation, vol. 2015-June, no. June, pp. 1374–1381, 2015.

[14] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation," 6 2018. [Online]. Available: http://arxiv.org/abs/1806.08756

[15] J. Gibson, The theory of affordances, 1979.

[16] R. Jain and T. Inamura, "Learning of Tool Affordances for autonomous tool manipulation," 2011 IEEE/SICE International Symposium on System Integration, SII 2011, no. December, pp. 814–819, 2011.

[17] T. T. Do, A. Nguyen, and I. Reid, "AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection," Proceedings - IEEE International Conference on Robotics and Automation, pp. 5882–5889, 2018.

[18] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," IEEE International Conference on Intelligent Robots and Systems, vol. 2016-Novem, no. October, pp. 2765–2770, 2016.

[19] ——, "Object-based affordances detection with Convolutional Neural Networks and dense Conditional Random Fields," IEEE International Conference on Intelligent Robots and Systems, vol. 2017-Septe, no. September, pp. 5908–5915, 2017.

[20] P. Van Molle, T. Verbelen, E. De Coninck, C. De Boom, P. Simoens, and B. Dhoedt, "Learning to Grasp from a Single Demonstration," 2018. [Online]. Available: http://arxiv.org/abs/1806.03486

[21] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan Robust grasps with synthetic point clouds and analytic grasp metrics," Robotics: Science and Systems, vol. 13, 2017.

[22] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," no. CoRL, pp. 1–10, 2017. [Online]. Available: http://arxiv.org/abs/1706.04652

[23] Karen Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," International Conference on Learning Representations (ICLR), pp. 1–14, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 2261–2269, 2017.

[26] R. Olga, J. Deng, H. Su, and J. Krause, "ImageNet Large Scale Visual Recognition Challenge," IJCV, 2015.

[27] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

[28] A. Aakerberg, K. Nasrollahi, C. B. Rasmussen, and T. B. Moeslund, "Depth Value Pre-Processing for Accurate Transfer Learning based RGB-D Object Recognition," pp. 121–128, 2017.

[29] M. Schwarz, H. Schulz, and S. Behnke, "RGB-D Object Recognition and Pose Estimation based on Pre-trained Convolutional Neural Network Features," ICRA, pp. 1329–1335, 2015.

[30] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss Functions for Neural Networks for Image Processing," pp. 1–11, 2015. [Online]. Available: http://arxiv.org/abs/1511.08861

[31] M. Safeea and P. Neto, "KUKA Sunrise Toolbox: Interfacing Collaborative Robots With MATLAB," IEEE Robotics & Automation Magazine, vol. 26, no. 1, pp. 91–96, 3 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8542757/