Cloth Smoothing Simulation with Vision-to-Motion Skill Model

Peter Nimac[1,2], **Matija Mavsar**[1,3], **Andrej Gams**[1]

[1] *Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[2] *Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[3] *University of Ljubljana, Faculty of Electrical Engineering, Tržaška cesta 25, 1000 Ljubljana, Slovenia*
E-mail: peter.nimac@ijs.si

Abstract

The handling of textiles by robots is currently a largely unexplored and underdeveloped area of robotics. The reason for this is the complexity of the actions resulting from the properties of textile and the difficulty in accurately determining the state of textile. Due to the considerable variability in the shape and size of planar, non-rigid objects, we have found that this challenge can best be addressed using advanced deep learning methods. In this paper, we demonstrate a vision-to-motion DNN (Deep Neural Network) trained to straighten a single crumpled corner on a rectangular piece of fabric that was deformed and captured inside a simulated environment. The neural network was trained to identify a correct grab point at which to grab the simulated fabric, and also a correct drop point to which to move the grabbed piece of fabric. For this simplified example, our trained model was able to achieve satisfactory results with an average error of 5.43 px in predicting the grab point position and an average error of 7.08 px in predicting the drop point position.

1 Introduction

Robotic manipulation of textiles remains a largely unexplored field. Therefore, the handling of clothing, fabrics and other textile elements is still done manually, except for the most basic tasks. The complexity of actions resulting from the properties of textiles, deformability, self-collision and self-occlusion; lack of general approaches on the detection and robotic handling of such materials; and the wide variety of actions involved in handling textiles, i.e. for different types of clothing, are prohibitive factors that keep operations involving textile manual. Only recently garment state estimation has been addressed [1]. The sheer number of possible states of textile objects and the actions associated with them makes it not only difficult to manipulate a textile object, but even just to describe it. However, advances in machine learning methods have made it possible to generate end-to-end perception-action pairs for robots.

In this paper, we present a cloth smoothing method where the skill is trained with a DNN describing a vision-to-motion model, as shown in Fig. 1. Similar methods have been shown in [2]–[5]. In [2], Tsurumine et al. demonstrated the use of DRL (Deep Reinforcement Learning) algorithms, combining value function-based reinforcement learning with automatic feature extraction from high-dimensional observations in DNN to improve the sample efficiency and learning stability with fewer samples.

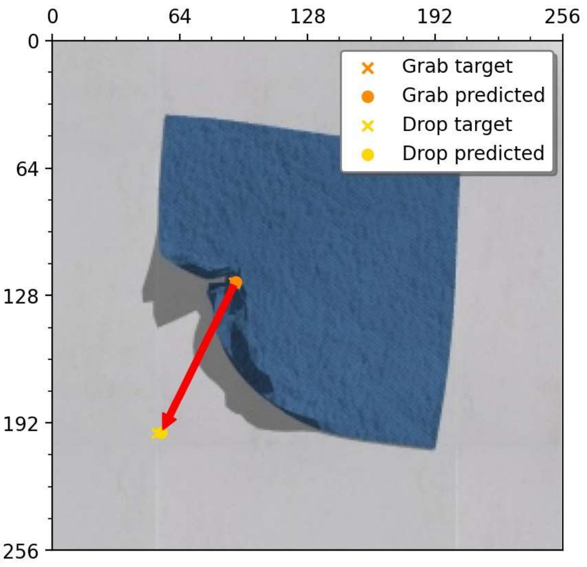


Figure 1: Prediction of grab and drop points for rectangular cloth manipulation based on a single input image. Crosses mark target grab (orange) and drop (yellow) points, while dots mark the predicted grab and drop points.

In [3] Wu et al. proposed another image-based DRL method to speed up training by separately training the pick policy and place policy instead of training both as a single set of actions. The latter approach would work for rigid objects because an entire object is moved uniformly. This is not the case for non-rigid objects due to their deformation during movement, which can lead to inefficient learning.

In [4] Seita et al. have proposed a DIL (Deep Imitation Learning) model-free approach for fabric smoothing that does not use reinforcement learning. Unlike reinforcement learning, imitation learning is a type of supervised learning in which the agent learns a control policy by analysing demonstrations of the policy performed by an algorithmic or human supervisor [6]. For smoothing fabrics, Seita et al. adopted an approach in which the fabric is smoothed by being pulled at its corners. With this

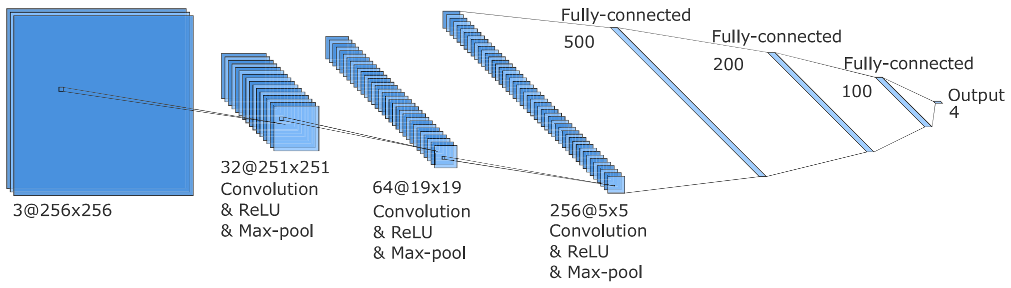


Figure 2: The architecture of image-to-motion DNN that was used in our experiments.

approach, the policy can be easily defined in a simulator and coded as an algorithmic supervisor.

In [5] Lee et al. have recently shown that fabric handling can be learned directly in the real world by the random interaction of the robot with a piece of fabric. To do this, they used DQL (Deep Q-Learning) approach with a self-supervised learning pipeline to learn fabric manipulation through autonomous data acquisition.

We propose to use a similar approach where the RGB image of the deformed fabric is used as input to an image-to-motion DNN and the starting and ending points of the robot motion are the output.

The rest of this paper is organized as follows. In Section 2 we describe the architecture of the image-to-motion DNN, that we used. Section 3 describes used training dataset and the method we took to construct it. In Section 4 we evaluate our model's performance, which is then followed by a discussion and a conclusion in Section 5 and Section 6 respectively.

2 DNN architecture for vision-to-motion skill encoding

Based on the previous work by Pahič et al. [7], we constructed an image-to-motion network (pictured in Fig. 2) that was used in our experiments and is structured as follows. The architecture consists of three convolutional layers and three fully-connected layers. The convolutional layers takes a $3 \times 256 \times 256$ pixel RGB image as input, where the first layer has a 6×6 kernel size and is followed by a ReLU layer and a max-pool layer with kernel size of 4×4 and stride 4. The second layer takes an input with the size of $32 \times 251 \times 251$, has a 5×5 kernel size and is followed by a ReLU layer and a max-pool layer with kernel size of 3×3 and stride 3. The third and final convolutional layer takes an input with the size of $64 \times 19 \times 19$ and has a kernel with size 4×4. Likewise, it's followed by a ReLU layer and a max-pool layer with the same construction as the one that is followed by the second layer.

The third layer then produces an output with the size of $256 \times 5 \times 5$, which is reshaped into a one-dimensional vector and passed to a fully-connected layer of 500 neurons. The second and third fully-connected layers count 200 and 100 neurons, respectively, and finally produce the output with the size 4. The output represents the predictions of four coordinates for grab and drop points defined in a 2D Cartesian space.

3 Dataset for training

The dataset was built with the use of PyBullet physics simulation engine [8]. The cloth was described in the simulation as a infinitely thin square plane with the side $a_p = 10$ cm. On both lengths, the plane was divided by 25 cuts of 4 mm in length to form a mesh made of isosceles right-angled triangles with catheti $a_c = 4$ mm and the hypotenuse $a_h = a_c\sqrt{2}$. The end effector was simply described as a small sphere with radius $R_{ef} = 3$ mm.

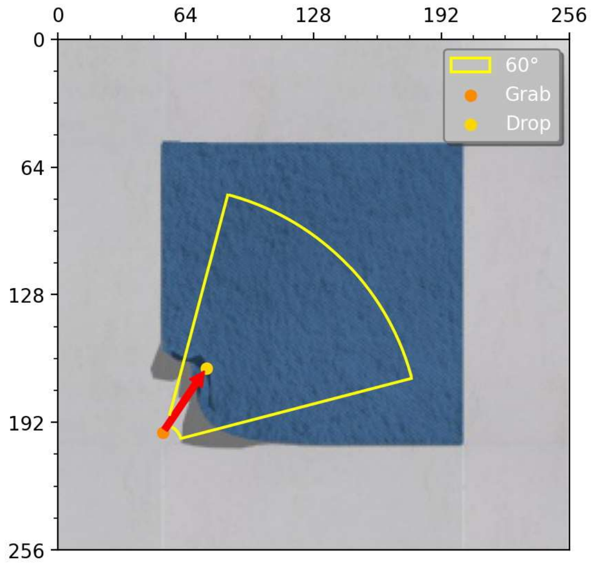
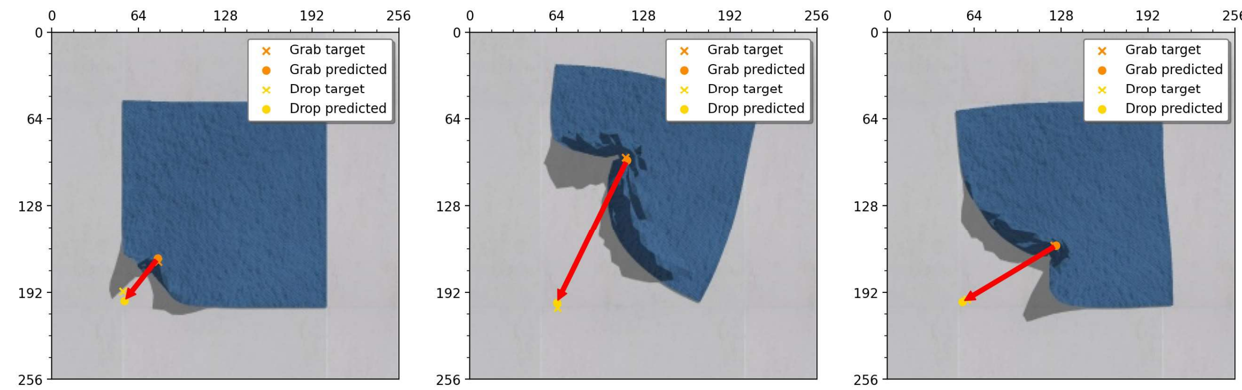


Figure 3: During simulation the bottom left corner is pushed to a random location within the area marked with a yellow border.

For each image within the dataset, the simulator performed the operation which was the opposite to smoothing. That is, in each simulation the end effector pressed down on the cloth at a random location within 4 mm range of the bottom left corner. Then it pushed the corner towards the center of the cloth for a random distance, up to 7 cm away from the initial position, at a random angle between 15° and 75°, as shown in Fig. 3. At the beginning of each simulation, the cloth was set in the same position and in a smooth state. Our idea is to train the DNN to correctly identify the starting and stopping points of the linear motion that deformed a flat cloth. With this information, a similar motion can be performed in reverse for cloth smoothing. Three thousand (3000) different simulations were performed in this manner and an



(a) $d_{err} = 7.1$ px, $g_{err} = 2.39$ px (b) $d_{err} = 3.89$ px, $g_{err} = 2.45$ px (c) $d_{err} = 3.74$ px, $g_{err} = 1.45$ px

Figure 4: Three examples of trained model predicting grab and drop points from a test input image.

image with resolution of 1000×1000 pixels of the deformed cloth was taken after each iteration, which was saved along with the initial and final position of the end effector. Before training, we also reduced the resolution of all images to 256×256 pixels to reduce the amount of excessive data and features that the DNN would have to process during training. This reduces the required computational resources and training time. We also observed that our DNN achieved a lower validation loss when trained on the input images with a chosen lower resolution than when trained on the images with original higher resolution. Although a DNN trained on higher resolution images could potentially perform better, this is not necessarily the case. As the size of the input image has the effect on predictive performance of the DNN [9]. Alternatively, the images could have been initially acquired at a desired lower resolution. However, it is easier to downsample higher resolution images to a lower resolution than the opposite when high resolution images are required. This approach also better reflects a real-life workflow where the workspace is monitored with a camera and the photos it takes are usually of a higher resolution.

Generated this way, the data pairs D in the dataset have the following structure:

$$D = \{C_j,\ M_j\}_{j=1}^{P}, \tag{1}$$

where P is the number of generated data pairs, constructed of images $C_j \in \mathbb{R}^{3\times H\times W}$ with height H and width W and corresponding grab and drop points of linear smoothing motions

$$M_j = \{\mathbf{g}_j,\ \mathbf{d}_j\}. \tag{2}$$

In above equation, vectors $\mathbf{g}_j$ and $\mathbf{d}_j$ are defined as

$$\mathbf{g}_j = \{g_{x,j},\ g_{y,j}\}, \tag{3}$$

$$\mathbf{d}_j = \{d_{x,j},\ d_{y,j}\}, \tag{4}$$

and represent x and y pixel coordinates for grab $\mathbf{g}_j$ and drop $\mathbf{d}_j$ locations.

4 Experimental evaluation

The network was trained during the course of 1500 epochs, where the learning rate was set to 1×10^{-7}, weight decay rate to 1×10^{-4} and the batch size to 10. Best validation loss was used as training metric and the training was automatically halted if validation loss remained unchanged for 100 consecutive epochs. The Adam optimiser [10] was used to update network parameters during training.

The model's achieved accuracy was then evaluated on a test dataset of 200 images, that were generated in the same way as a training set. Model's performance is shown on three samples from the test set on Fig. 4 and it's accuracy in the form of drop point error and grab point error is shown in Fig. 5. To estimate the model's output error, we calculated the Euclidean distance between the target and predicted points. According to calculations, the model on average misses the target drop point by $\bar{d}_{err} = 7.08$ px and it misses the the target grab point by $\bar{g}_{err} = 5.43$ px. The median values of both errors are slightly lower, with $\tilde{d}_{err} = 6.97$ px and $\tilde{g}_{err} = 4.77$ px for drop point error and grab point error, respectively. The largest tested drop point error was $d_{err,\ max} = 21.24$ px and largest grab point error was $g_{err,\ max} = 15.22$ px. On the contrary, the smallest drop point and grab point errors were measured at $d_{err,\ min} = 0.66$ px and $g_{err,\ min} = 0.14$ px.

5 Discussion

This paper describes our initial research towards robot manipulation of textile, where an experiment using a single cloth was carried out to predict smoothing motions.

For higher applicability of the described approach, the diversity of the training database would have to be increased. In these initial experiments we focused on a single instance of flattening a square, mono-coloured cloth. Furthermore, the initial deformation was always performed according to the same procedure. Due to this, our model trained for a very specific case, that is to smooth a mostly flat blue cloth of a square shape with only having a bottom left corner crumpled in a random position.

To better generalise our model, in the future we will first construct a larger dataset with more sample variability. The dataset needs to contain images of differently shaped cloths with different colours and patterns. Then

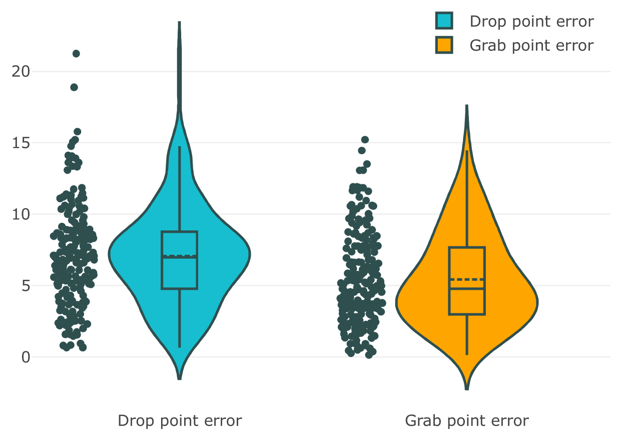


Figure 5: Drop point error and grab point error distribution represented with a violin plot and a box-and-whisker plot. On the left side of each plot, the dark dots represent actual calculated errors for all tested samples. Samples that fall within the area of the boxes represent 50 % of all tested samples.

the images need to be captured at various camera angles and at different cloth deformations. That is deformations from different points, deformations from a full set of angles, deformations from multiple locations (e.g. from the edge of the cloth or from a random point on the cloth) and deformations from more complex end-effector trajectories.

Additionally, the learning accuracy can be improved with additional data, e.g. angle at which the deformation occurred or what path the end-effector took to deform the cloth. Introduction of this variability will make the model more robust for deployment on a real robot system with a real cloth through domain randomisation method of sim-to-real transfer [11]. By utilizing highly randomised simulation environments we will be able to approach the distribution of the real world data, thus reducing the discrepancy between the model, trained in the simulation, and a model, that would be trained in the real world [12].

Aside from PyBullet, we also tried the MuJoCo robotics simulator, but in its present state and in the case of cloth simulation, it often became unstable as cloth deformation level and deformation complexity increased. Another alternative approach for cloth simulation, that we have yet to try, is described in [4], where Seita et al. designed a simulator, that is specifically tailored for cloth simulation with the support of Blender.

6 Conclusion

In this work, we have shown how to generate and apply image-to-motion skill model for cloth smoothing on a simplified example. We constructed a DNN architecture for generation of linear robot trajectories based on images of deformed cloth. Initial results are promising with a mean error of 5.43 px in grab point position prediction and mean error of 7.08 px in drop point position prediction for images with size of 256×256. This provides a foundation for our future work, where we will extend the training dataset with more samples and include clothes in random crumpled states with more complex cloth manipulation trajectories. In the future we intend to apply our approach in a real world application. Prior to that, the approach will be tested in various simulated environments through a sim-to-sim-to-real method.

References

[1] D. Triantafyllou, I. Mariolis, A. Kargakos, S. Malassiotis, and N. A. Aspragathos, “A geometric approach to robotic unfolding of garments”, *Robotics Auton. Syst.*, vol. 75, pp. 233–243, 2016.

[2] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, “Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation”, *Robotics and Autonomous Systems*, vol. 112, pp. 72–83, 2019.

[3] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, “Learning to Manipulate Deformable Objects without Demonstrations”, *arXiv:1910.13439 [cs]*, Mar. 2020, arXiv: 1910.13439.

[4] D. Seita, A. Ganapathi, R. Hoque, *et al.*, “Deep Imitation Learning of Sequential Fabric Smoothing Policies”, *CoRR*, vol. abs/1910.04854, 2019.

[5] R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner, “Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience”, *CoRR*, vol. abs/2010.03209, 2020.

[6] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, “DART: Noise Injection for Robust Imitation Learning”, p. 14, 2017.

[7] R. Pahič, B. Ridge, A. Gams, J. Morimoto, and A. Ude, “Training of deep neural networks for the generation of dynamic movement primitives”, *Neural networks : the official journal of the International Neural Network Society*, vol. 127, pp. 121–131, 2020.

[8] E. Coumans and Y. Bai, *PyBullet, a Python module for physics simulation for games, robotics and machine learning*, http://pybullet.org, 2016–2021.

[9] M. L. Richter, W. Byttner, U. Krumnack, A. Wiedenroth, L. Schallner, and J. Shenk, “(Input) Size Matters for CNN Classifiers”, *Artificial Neural Networks and Machine Learning – ICANN 2021*, 2021, pp. 133–144.

[10] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *The International Conference on Learning Representations (ICLR)*, 2015.

[11] W. Zhao, J. P. Queralta, and T. Westerlund, “Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey”, *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744, 2020.

[12] J. Tobin, “Real-World Robotic Perception and Control Using Synthetic Data”, University of California, Berkeley, 2019.