A Study on Deep Learning Based Search Method

CMPUT 428 Final Project Report

Chen Jiang, Steven Weikai Lu, Martin Jagersand*
Department of Computing Science
University of Alberta
Edmonton, Canada, T6G 2E4
{cjiang2, weikai, mj7*}@ualberta.ca

Abstract — In the project, we investigate in planar object tracking task and present with an idea of using a deep convolutional neural network for estimating the relative updating warp parameters between a pair of images. A method concerning a corner displacement parameterization is used to represent and recover the warping transformation between two corners. Initial theory workout and the training of the network have shown that it is feasible to develop such a neural network based search method for warping parameters updates.
Keywords —Planar Object Tracking, Convolutional Neural Networks(CNNs), Deep Learning, Warping Transformation

1. Introduction

While many open researches focus on object tracking problems, we still notice a short range of researches among planar object tracking in recent years. Fast and high precision planar visual tracking [1] is critical to the success of several robotics and virtual reality applications like autonomous navigation and visual servoing [2] and has been a popular aspect in object tracking research [3]. Apart from the challenging aspects of object tracking tasks like occlusion, illumination changes and fast motion, the nature of object tracking demands robustness in arbitrary object tracking [4], which makes the task even more challenging.

Modular Tracking Framework (MTF) [5] decomposes the object tracking problem into an iterative process which consists of three modules, that is, Appearance Model (AM), State Space Model (SSM), Search Method (SM). The AM predicts an object's appearance, including color, lighting, and shadows in an image as a function of some underlying states. The SSM defines the degrees of freedom (DOF) needed to update the tracking window. The SM computes the optimal warping parameters to update the tracking window under the restriction of SSM. In the project, we focus on studying and improving the SM for the overall tracking task.

In recent years, combined with successful architectures like VGG-Net [6], convolutional neural networks have achieved remarkable performance over a variety of computer vision task like object detection, image recognitions, etc. Several recent works have expanded the studies of deep neural networks to solve a variety of traditional computer vision tasks like homography estimation [7], visual servoing [2], medical RegNet [8], etc. It is possible to develop a deep learning powered SM to search for optimal warping parameters. In this project, we present our scheme and design of DL-based SM and relevant analysis, experimentation and future planning.

The rest of the report is organized as follows: In Section 2, the proposed methodology is presented. Project experimentation and any other relevant information are presented and discussed in Section 3. Finally, concluding remarks are drawn with a brief future plan discussion in Section 4.

2. Methodology

Our methodology emphasizes on the design of SM and is designed based on the study of deep homography estimation – given a pair of images and their sparse 2D 4-points features which are defined as two corners, we estimate the relative warping transformation between those two corners. The overall pipeline is depicted in Figure 1.

We propose an overall pipeline starts from setting similar experimentations as DeTone et. al [7] and generate a seemingly infinite synthetic data pairs of $(I_A, I_B, \Delta p)$. Unlike DeTone et. al [7], we introduce some variations of random corners generation based on the different DOF requirement to synthesis the real-world planar tracking scenario. We use our generated synthetic data to power up our CNN architecture before applying transfer learning and fine-tuning the model on actual real-world tracking data.

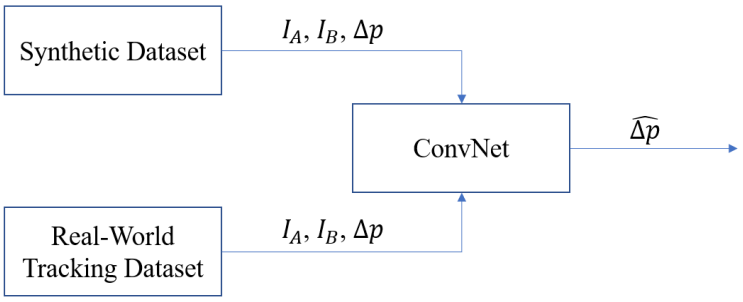


Figure 1: DL-powered SM pipeline.

2.1 N-Points Parameterization of Warping Transformation

We started our research in the parameterization of the warping transformations based on our experiment observation that it is hard to directly predict the parameters of the warping transformation matrix. Such a fact was also reported in DeTone et. al [7] that, balancing the rotational and translational terms as part of an optimization problem tends to be difficult. One possible explanation for this is that traditional warping transformation infos represented by those matrices are less linearly correlated to images than pixel-wise location infos with which we traditionally utilize to power up regression network and perform a variety of tasks like bounding box regression.

We define our parameterization of transformation (DOF defined by SSM) using a N-points parameterization method. Letting $\Delta u_i = u'_i - u_i$ be the x-coordinate offset for the corners and $\Delta v_i = v'_i - v_i$ be the offset on y-coordinate, where i is the number of corners determining DOF. Here are some formulation for common 2-D transformation classes.

Class I: Isometries: Since Euclidean transformation has 3 DOF, we take 1 points each with 2 coordinate value and 1 point with 1 coordinate value to retain 3 DOF.

Class II: Similarity transformation: Since similarity transformation has 4 DOF, we take 2 points each with 2 coordinate value to retain 4 DOF:

$$H_{2\_points} = \begin{pmatrix} \Delta u1 & \Delta v1 \\ \Delta u3 & \Delta v3 \end{pmatrix} \quad (1)$$

Class III: Affine transformation: Since affine transformation has 6 DOF, we take 3 points each with 2 coordinate value to retain 6 DOF:

$$H_{3\_points} = \begin{pmatrix} \Delta u1 & \Delta v1 \\ \Delta u2 & \Delta v2 \\ \Delta u3 & \Delta v3 \end{pmatrix} \quad (2)$$

Class IV: Projective transformation(Homography): Since projective transformation has 8 DOF, we take 4 points each with 2 coordinate value to retain 8 DOF:

$$H_{4\_points} = \begin{pmatrix} \Delta u1 & \Delta v1 \\ \Delta u2 & \Delta v2 \\ \Delta u3 & \Delta v3 \\ \Delta u4 & \Delta \mathrm{v}4 \end{pmatrix} \quad (3)$$

The 4-points parameterization is equivalent to projective transformation. The 3-points parameterization is equivalent to affine transformation. The 2-points parameterization is equivalent to similarity transformation. The 1-points parameterization is equivalent to pure translation transformation. The corner displacement representation of the warping transformation shares the same DOF as the traditional matrix representation. Once the displacement of the four corners is known, it is easy to convert the corner displacement parameterization back into traditional matrix representation by solving linear system.

2.2 Synthetic Dataset Scheme

The dataset used for model training is the most critical part of the process. During our experimentation, we notice the fact that, direct training using real-world tracking dataset leads to struggling result once the object used for tracking changes dramatically. The nature of object tracking requires robustness to arbitrary objects. Therefore it is critical to teach the neural networks to perform proper feature encodings given different objects. Since the number of objects presented in real-world planar tracking dataset is significantly less compared to other real-world image dataset, therefore we look for an alternative approach to achieve a synthetic tracking effect.

The core part of the training sample is random relative motion of bounding box between two frames. Our synthetic dataset generation pipeline is mostly inspired by the process from DeTone and VS et. al. The overall procedure of generating a training sample is shown as follows:

1) A rectangle corner C_A of size specified by the ConvNet model is randomly placed on a randomly selected large image I at position p.
2) To synthesize better on the planar tracking scenario, we then randomly perturb a selected number of corners defining the position p by a random variable which

follows a uniform distribution within range [- ρ_1 , ρ_1]. The number of corners that need to be perturbed corresponds to the DOF of the warping transformation we desire.

3) A unique warping, namely Δp , can be derived from a random variable following a uniform distribution within range [- ρ_2 , ρ_2]. It defines the relative difference of the original corner C_A and new perturbed corner C_B . We update to C_B by adding C_A with Δp . A patch I_A is then cropped according to the C_A .
4) We compute the warping transformation matrix M_{AB} from C_A to C_B . The inverse of the $(M_{AB})^{-1} = M_{BA}$ is then applied to the large image I to produce image I'. A second patch I_B is then cropped from I' according to C_A .

Δp will be transformed to N-points parameterization format as stated in 2.1. This process is relatively simple, where the warp will be simulated then the corner difference can be calculated algebraically as the N-points parameterized Δp . Thus we have acquired a triple of training sample $(I_A, I_B, \Delta p)$. By doing this we can have theoretically unlimited amount of image pairs of consecutive frames.

2.2.1 Corners Rectification

One of the implementation issues of adapting deep neural network to search method module is that, the deep neural network only accepts fixed-size square inputs, while during the planar tracking scenario that a fixed-size square tracking window is not guaranteed, especially if any transformation with DOF higher than 3 is present. To solve this problem, we propose two rectification strategies to prove square inputs for our deep neural nets while maintaining lossless information from the planar tracking window.

Average Cropping: We take the average of the four corner coordinates from the planar corner. Provided with a desired patch size, we construct an approximate rectangle corner using half of the patch size.

Min-Max Cropping: We take the minimum and the maximum of the x, y coordinates from the four corner coordinates. We then define the approximate rectangle corner in the range of minimum and maximum coordinates values.

Figure 2 presents a visualization of two different cropping strategies for a randomly generated planar corner.

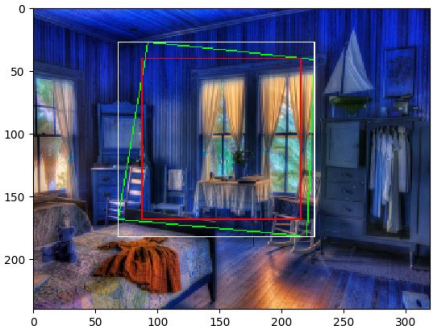


Figure 2: A visualization of two different corner cropping strategies for a randomly generated planar corner. The green corner stands for the ground truth corner, while the red and the white boxes stand for average and min-max strategy respectively.

In order for the entire pipeline to make sense, such rectification process has to preserve lossless information within the original bounding box, or in other words, a mapping satisfying bijective property between the original bounding box and rectified square bounding box must exist. Procedure to accomplish this is illustrated as follows:

1. Input a frame $I_{original}$
2. Let P be the warping matrix derived from the original bounding box $b_{original}$ and rectified square bounding box b_{square}, where $b_{square} = P \cdot b_{original}$ and $I_{square} = P \cdot I_{original}$. Note that P is the bijective mapping between the two fields.
3. During run time, b_{square} is used to crop a patch as input of ConvNet producing a predicted $\Delta\hat{p}_{square}$.
4. Update b_{square} by $\hat{b_{square}}' = warp(b_{square}, \Delta\hat{p}_{square})$.
5. Update original bounding box by $\hat{b_{original}}' = P^{-1} \cdot \hat{b_{square}}'$.

Figure 3 presents the general pipeline for corner rectification.

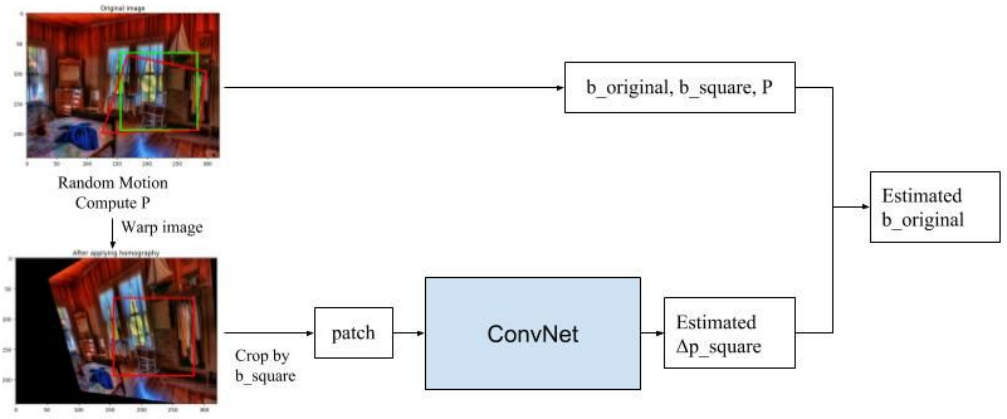


Figure 3. Corners Rectification Pipeline

2.2.2 Illumination and Occlusion perturbation

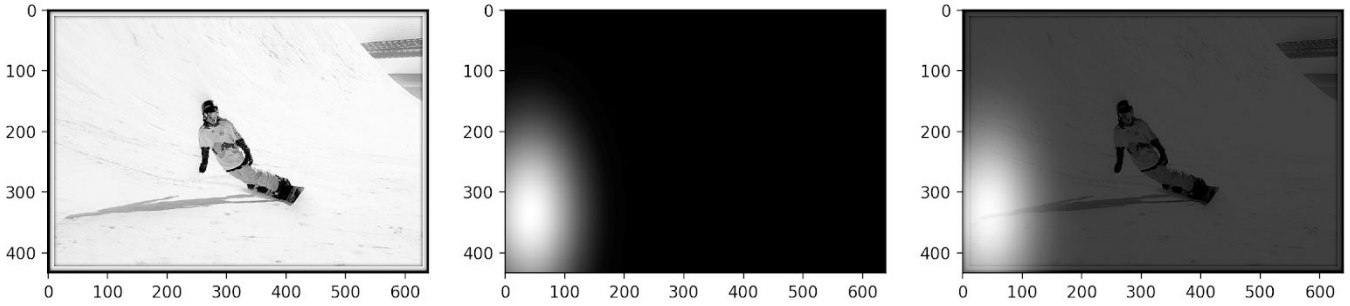
In practice, appearance of the tracked object can be perturbed by various unknown factors. Warping function can only account for the pose change of planar objects, while not able to handle other non-geometric changes, commonly light and occlusion. Hence, two other perturbations are introduced to the training sample for more robust performance. We model global light reflection by a random 2-dimensional Gaussian distribution. Local light changes are more subtle and challenging, involving computational-intensive rendering procedure. Hence we slightly abused the assumption that our task is limited to 3-dimensional planner objects only, which allows direct 2-dimensional light source. For an image I to be used to create a sample, we apply the following 2-D Gaussian to each pixel (x, y):

$$I_l(x,y) = I(x,y) + f(x,y) \quad (4)$$

where the 2-D Gaussian $f(x,y)$ is defined as:

$$f(x,y) = A \times exp(-(\frac{x-x_0}{2\sigma_x{}^2} + \frac{y-y_0}{2\sigma_y{}^2})) \quad (5)$$

where (x_0, y_0) is the light projection center, A is the light intensity factor, and (σ_x^2, σ_y^2) is the independent variance on the two axes respectively. Figure 4 presents the visualization of the Gaussian lighting on an example image.



(a) (b) (c)

Figure 4. Overall procedure of adding illumination perturbation: (a) Example original grayscale image; (b) Example random grayscale light from 2D Gaussian distribution; (c) Example light perturbed grayscale image.

It is easy to see that the synthetic light is a change only in 2-D domain, regardless of the material and textures. However, such more non-linear perturbation can be introduced by occlusion. Occlusion is more challenging to handle since it could varies in size, shape and even semantic content. To accomplish this, we use a segmentation-based occlusion method. This enable us to use realistic object from a variety of scene and context. The MS-COCO dataset is also very compatible with segmentation algorithms. The procedure of imposing such perturbation can be done automatically. Figure 5 presents the visualization of the occlusion perturbation on an example image.

1. Two different images are randomly drawn from MS-COCO dataset, one of them I_A is the target image on which occlusion perturbation is imposed, while the other I_{seg} is where the real-world object for occlusion taken from.
2. Apply segmentation algorithm on I_{seg}, producing n disjoint regions set R_{seg}.
3. Randomly select one of the regions from R_{seg}, and overlay that on I_A.
4. Output modified I_A.

(a) (b)

Figure 5. Overall effect of occlusion perturbation: (a) Example original image; (b) Perturbed image.

2.3 ConvNet Model

We experiment with the HomographyNet proposed in DeTone et. al [7]. Figure 6 presents the architecture of the HomographyNet. The network shares a similar architecture to the Oxford's VGG-Net. Inputs of the channel-wise stacked images using patches I_A *and* I_B are fed into the network. 8 convolutional layers of 3x3 filters and stride 1, with numbers of filters settled as 64, 64, 64, 64, 128, 128, 128 and 128 are presented, each followed by a batch normalization layer and a ReLU activation. A max pooling layer of 2x2 filters and stride 2 is used after every two convolutions. The convolutional layers are then followed by a dropout with a probability of 0.5 and two fully connected layers. The first fully connected layer has 1024 units, followed by a ReLU activation. Dropout with a probability of 0.5 is then applied after the output from the last fully connected layer and the output is then forwarded into the last fully connected layer with units equal to the number of parameters required to parameterize the desired warping transformation.

We use the Euclidean L2 distance function for direct task:

$$loss(D) = \| p(x) - q(x) \|^2 \qquad (6)$$

where $p(x)$ to be the ground truth corner displacement, $q(x)$ to be the predicted corner displacement.

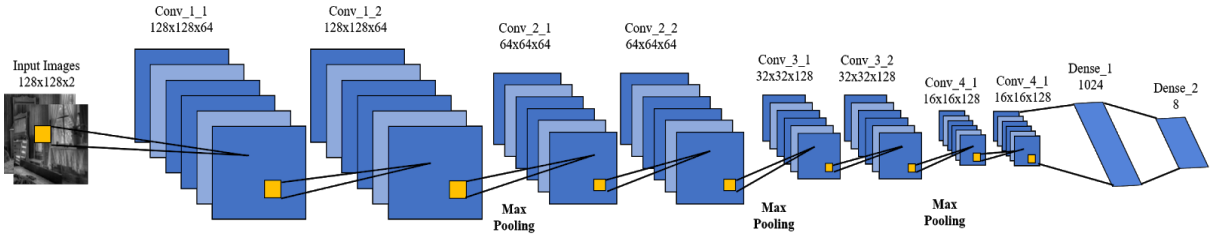


Figure 6. Architecture of HomographyNet.

2.5 Fine-Tuning on Real-World Tracking Data

The modular decomposition of tracking process proposed by MTF [5] can be formulated into a Markov Decision Process (MDP), where AM gives an evaluation of state, SSM define the search space, and SM searches for optimal parameter. Under this assumption, only the two patches, usually the template patch and current patch cropped by corners, are used by MTF to determine the optimal parameters of corner updates for the SM. Combined with the parameterization of warping transformation we previously defined, it is not difficult to see that the state of the search state can be formulated by $(I_A, I_B, \Delta p)$. This assumption enables the end-to-end supervised training of convolutional neural networks and thus enabling SM to be formulated into a regression problem. Therefore it is possible to apply transfer learning using our trained model on the real-world tracking dataset. Similar ideas are also utilized in another planar object tracker based on nearest neighbor search. [9]

Given two conservative frames F_A, F_B and their corresponding tracking window C_A, C_B, we can calculate the corner displacement Δp as the substitution of two tracking window, that is,

$$\Delta p = C_B - C_A \qquad (7)$$

We then crop the patches I_A and I_B using C_A at frame F_A and F_B, in a similar fashion in our synthetic dataset generation. The trained model will be fine-tuned using the acquired real-world tracking data.

3. Experiment Details

Since no pre-trained weights are provided by DeTone et. al [7], therefore we start our experiments by re-creating the training procedure of the paper and first acquire a trained weight of the HomographyNet. The CNN network is implemented and trained using Keras backend with Tensorflow 1.8 in Python 3.5. The network is initialized using Xavier initialization [10] and is trained for 12 epoch using SGD with a momentum of 0.9. Initial learning rate is 0.005 and is decreased by a factor of 4 every four epoch.

To create the training data with a rich set of image features, we use the MS-COCO [11] 2014 Training Set to generate our synthetic dataset under the protocol described in Section 2. All images are resized to 320x240 and converted to grayscale. We randomly choose 41, 600 image from the MS-COCO Training Set and generate 12 synthetic image pairs for each image, resulting in a number of 499, 200 training samples. We choose our patch size = 128 and ρ_2 = 32. We then generate our testing set using grayscale images resized to 640x480 with patch size = 256 and ρ_2 = 64 from MS-COCO 2017 Test Set.

Unfortunately, due to the time limit of the project and the limit of our GPU resource for handling this large amounts of data, we are only able to successfully finish the re-creation of Deep Homography Estimation for now. Table 1 shows the testing average mean corner error of our re-creation and of DeTone et. al [7]. Figure 7 presents an example homography prediction using HomographyNet.

Model	Our Re-creation	DeTone et. al.
Mean Corner Error	9.3951	9.2

Table 1: Testing Mean Corner Error on the generated MSCOCO-Warp Testing Set.

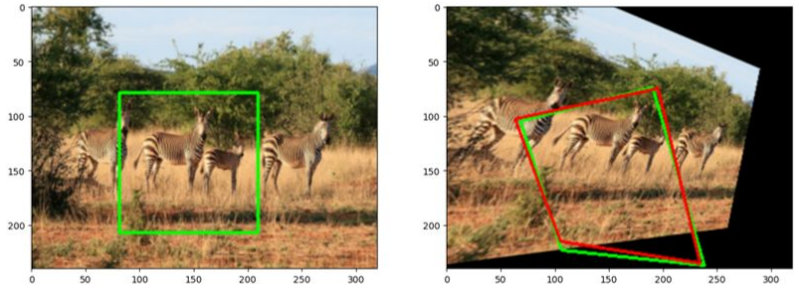


Figure 7.. A prediction example. Error between ground truth(green) and prediction(red) corners is 5.9161

More experiments described in section 2 will be conducted in time.

4. Conclusion and Future Plans

In this report, we present our project methodology and pipeline for developing a deep learning based search method. While more experiments are needed for further proofs of the possibilities of the deep search method, the experiments already performed have helped us gain insights in overall methodologies design and have demonstrated a degree of effectiveness of our methods. In the future, we plan to continue our experiment settings and keep an up-to-date project updates.

Acknowledgement

This is the final project report for in CMPUT 428, instructed by Prof. Martin Jagersand and Teaching Assistant Abhineet Singh, University of Alberta. While we work on our project in a deeply-cooperating fashion, a work distribution of each group member can be roughly listed as follows:

- Chen Jiang: Overall project design, network architecture design, coding
- Steven Weikai Lu: Synthetic dataset generation, illumination and occlusion perturbation.

Project Checklist

Although the methodology of the DL-based search method is nearly complete, several experiments still need to be conducted before reaching concrete conclusions. The following lists suggest the experiments that have been done currently for the submission, and the future experiments that will be done:

- Develop the codings for HomographyNet training, validation, testing and prediction. (Done)
- Develop a better data I/O buffer to handle huge amounts of image inputs. (Done)
- Acquire a pre-trained weight for the HomographyNet, to be used as a possible model for transfer learning. (Done)
- Train the HomographyNet using synthetic dataset under the protocol of DeTone et. al. (Done)
- Train the HomographyNet using synthetic dataset under the methods described in section 2. (TO-DO)
- Generate Δp ground truths for tracking dataset like VOT, CMT, etc. (Done)
- Finetune the HomographyNet using data from tracking dataset. (TO-DO)
- Evaluate HomographyNet model on synthetic testing dataset under the methods described in section 2 with different DOFs. (TO-DO)
- Evaluate HomographyNet model on real-time tracking dataset. (TO-DO)

Reference

[1] Liang, P., Wu, Y. and Ling, H., 2017. Planar object tracking in the wild: A benchmark. *arXiv preprint arXiv:1703.07938*.

[2] Bateux, Quentin & Marchand, Eric & Leitner, Juxi & Chaumette, François & Corke, Peter. (2017). Visual Servoing from Deep Neural Networks.

[3] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A. and Torr, P.H., 2016, October. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision* (pp. 850-865). Springer, Cham.

[4] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A. and Torr, P.H., 2016, October. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision* (pp. 850-865). Springer, Cham.

[5] A. Singh and M. Jagersand, 2016. Modular tracking framework: A unified approach to registration based tracking. arXiv preprint arXiv:1602.09130.

[6] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[7] DeTone, D., Malisiewicz, T. and Rabinovich, A., 2016. Deep image homography estimation. arXiv preprint arXiv:1606.03798.

[8] Sokooti, H., de Vos, B., Berendsen, F., Lelieveldt, B.P., Išgum, I. and Staring, M., 2017, September. Nonrigid image registration using multi-scale 3D convolutional neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 232-239). Springer, Cham.

[9] Dick, T., Quintero, C.P., Jägersand, M. and Shademan, A., 2013, June. Realtime Registration-Based Tracking via Approximate Nearest Neighbour Search. In *Robotics: Science and Systems*.

[10] Glorot, X. and Bengio, Y., 2010, March. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256).

[11] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.