Semester Project: Real to Synthetic Image Translation for Pose and Image Understanding in Robocup

Serena Bono
sebono@student.ethz.ch

15/06/2021

Abstract

Image translation has been very successful in generating images with various domain shifts using the popular CycleGAN framework. Few image translation models have also been able to generate different deformations or properties of objects. In this semester project, we are interested in exploring image understanding through image translation models. In particular we want to understand the pose of the camera in order to automatically detect key image features of interest. We use the framework in the context of Robocup football matches for understanding the football field. RoboCup is a scientific initiative with the goal to advance the state of the art of intelligent robots able to play soccer, originally aimed to reach human soccer World Cup champions level by 2050. The work presented below suggests two possible unsupervised strategies for image translation from real to synthetic: one based on GANs and the other based on pose regression and homography warping. The GAN framework shows promising qualitative results where real data are converted to synthetic-like images. The images can be easily thresholded to compute the regions of interest. We also show results of direct pose regression on the translated images.

1 Introduction

Current computer vision and particularly GAN models have been very successful with translating images (e.g., between different domains such as seasons, weather conditions etc...)./todoremove ...'s everywhere Image translation has also been used for understanding, particularly for semantic segmentation [10]. In this semester project, methods are investigated to perform image translation for image and pose understanding. The specific scenario chosen is that of Robocup. Robocup is an international robotics competition where fully autonomous humanoid robots play the game of soccer competing in adversarial teams. One of the main challenges of the competition is localizing the robot based on the images recorded by the camera positioned on its forehead. 6-dof pose understanding through image translation is a novel concept and has not been well-explored for images.

We propose two different methods of understanding poses and thus the scene through image translation:

1. **Cycle-GAN + Posenet**: The Cycle-GAN takes as inputs real images and synthetic images and aims to translate the real images to the synthetic domain, in order to create a set of translated images as similar as possible to the synthetic images. The synthetic images and their pose are fed as training inputs to the Posenet. After that, the Posenet attempts to guess the position of the translated images.

2. **Posenet + Unsupervised Learning**: The synthetic images and their pose are fed to the Posenet as training inputs. After few epochs the Posenet attempts to to guess the position of the real images. For each pose a synthetic image is generated and a loss is computed between the real and the newly generated synthetic image.

To evaluate image pose regression methods I generated synthetic images from the resulting pose and placed them alongside to the real images.

2 Related Work

The work flow of the project relies on some well known domain translation and pose regression techniques.

2.1 Cycle-GAN for Domain Translation

References to Cycle-GANs used in domain adaptation tasks could be found in [5]. As explained in detail in the paper, Cycle-GANs are exploited for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. A GAN attempts to learn the underlying distribution G_{AB} : X $\rightarrow$ Y, such that the results of $G_{AB}(X)$ are indistinguishable from Y. The problem is that the GAN is trying to solve is under-determined, meaning the underlying distribution might not be bijective, but more commonly is surjective. As explained in the following sections this is one of the biggest challenges when working with GANs and it can also cause the so-called mode collapse. To try to ensure bijectivity the GAN introduces another mapping to close the cycle: the mapping $G_{BA}(G_{AB}(X)) : G_{AB}(X) \rightarrow$ X. The task of distinguishing between the generated data and the input data is performed by a discriminator. The discriminator is adversarial to the generator in the sense that it points out generator's defects in such a way that both improve one another to reach the optimal state in which the generator completely defeats the discriminator (i.e. the discriminator outputs a 50/50 probability of having a generated/input image). Monitoring the convergence of the Cycle-GAN is a very complex matter, because the generator and discriminator's losses do not give information about it. The literature contains some works related to this issue, such as Wasserstein GAN [7]. The generate Cycle-GAN Architecture is depicted in Figure 1.

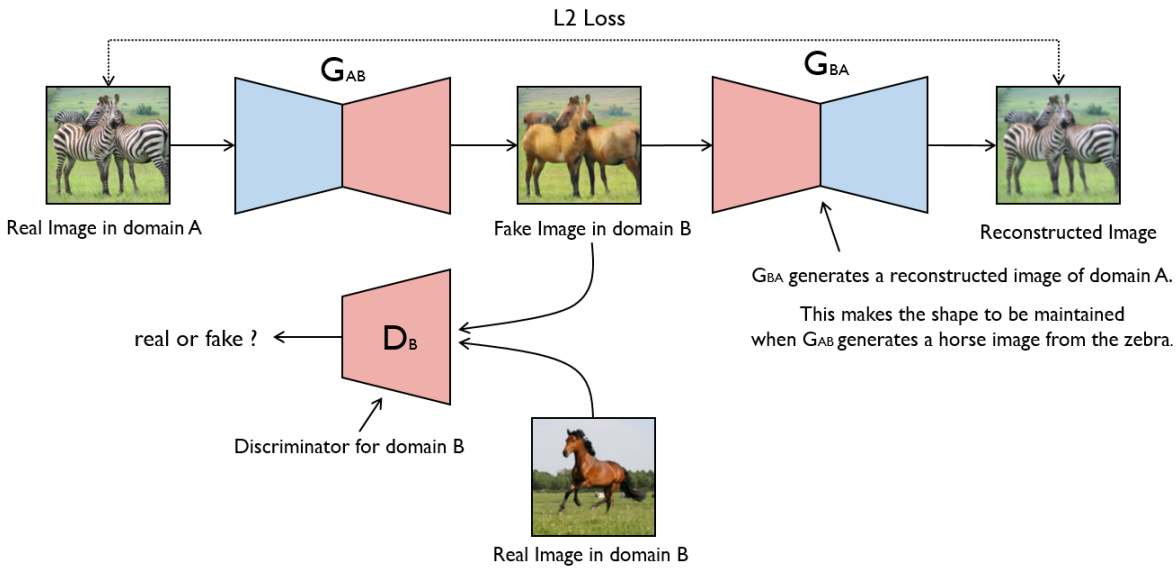


Figure 1: Cycle-GAN "cyclic" architecture
`https://modelzoo.co/model/mnist-svhn-transfer`

2.1.1 Metric of Evaluation in Cycle-GANs: FID

The Frechet Inception Distance (FID) [2] is well-known metric for GAN evaluation: it calculates the distance between feature vectors of real and generated images. The score represents the dissimilarity of any two distributions, therefore lower the score higher the accuracy. The perfect score is 0.0 and it indicates that the two groups of images are identical. Rather than comparing the two images pixel to pixel, the FID metric computes the mean and standard deviation of one of the deeper layers of the neural network, namely a layer closer to output nodes that correspond to real-world domain. The metric is given as follows:

$$\text{FID} = |\mu - \mu_w|^2 + \text{tr}(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{1/2}). \tag{1}$$

However, Eq. (1) requires the features to be computed meaningfully for both the distributions. While for natural images, ImageNet [4] pre-training provides meaningful feature vectors, this is not as reliable for the images of the Robocup field.

2.2 Resnet for Pose Regression

Resnet is also known as *Deep Residual Network*. Deep Residual Network are well known for being arguably the most groundbreaking work in the computer vision/deep learning community in the last few years. Thanks to their peculiar architecture, they are able to train up to hundreds or even thousands of layers and still achieves compelling performance. The core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in Figure 2. The input x is fed to the function F, to be re-added at the end to the output of $F(x)$; additionally it has been argued that stacking layers are not going to reduce the performance, because one could naively stack identity mappings in between, without changing absolutely nothing in the final output. The advantages of using ResNets are mainly two:

1. The skip connections in ResNet solve the problem of vanishing gradient in deep neural networks by allowing an alternate shortcut path for the gradient to flow through.
2. The model learns the identity functions: this ensures that the higher layer will perform at least as good as the lower layer.

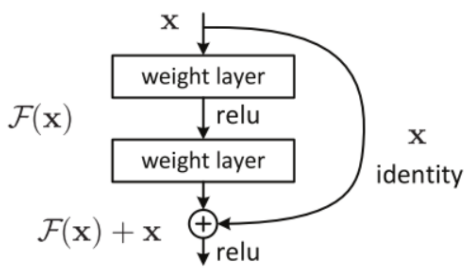


Figure 2: Rensnet's residual blocks

There have been many attempts to use ResNet for pose estimation like in [12] or in [8]. The paper tackles the problem as a regression task and regresses an image to a normalized pose vector. The main idea behind it was to define some model with its parameters, feed them to the network hoping that the loss would decrease through optimization.

3 Domain Translation

The Cycle-GAN was already implemented on GitHub [3]. In order to start experimenting with the Cycle-GAN, I had to generate synthetic images of the soccer field. To do so, it was enough to write a code snippet (GitHub repository [11] under the path */generate_synthetic_images/TrainBmain_no_robots.py*) that would generate homograpies of the top viewpoint of a soccer field. The task was eased by the fact that soccer fields can be accurately defined by a 2D image and then projected using an image warp. The missing details (i.e. the goal post depth/height) were not too relevant. The position of the camera, irrelevant for this specific task, will be very important in the following, therefore it is saved in the form of $(\delta(x), \delta(y), \delta(z), q_1, q_2, q_3, q_4)$. The first three values represent the displacement in the respective directions, while the last four are a particular measure of orientation named *quaternions*: 4-D values easily mappable to legitimate rotations by normalization to unit length.

Training a Cycle-GAN is always a difficult task: tuning the hyperparameters such as images input/output size and the learning rate are crucial to a correct implementation. The chosen values are listed in Table 2.

Epochs	Decay Epochs	Batch	LR	Pull Size	Load Heigth	Load Width
200	100	1	2e-4	5	128	128

Table 1: Hyperparameters for Cycle-GAN Implementation

(a) Dark Green Becomes Black (b) Low-Res Synthetic Images (c) *MODE COLLAPSE*

Figure 3: Some of the problems encountered while training the Cycle-GAN

3.1 Some Results

Some of the problems so far encountered were:

1. The synthetic images were low resolution as shown in 3b. This would negatively impact the ability of the Cycle-GAN to detect features in the synthetic images and find the correct mapping to the real images. This problem was very easily solved by changing the resolution in the saving process.

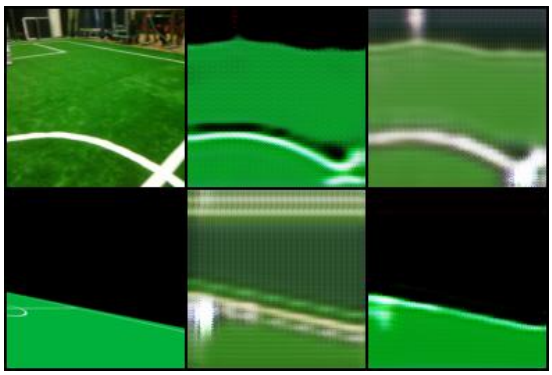
2. The results did show some mode collapse, the entire picture would turn black as shown in 3a. Even though it is extremely easy to encounter, mode collapse is a complicated problem to solve. It is due to the fact that the generator starts producing only one, especially plausible output. If the generator starts producing the same output (or a small set of outputs) over and over again, the discriminator's best strategy is to learn to always reject that output. But if the next generation of discriminator gets stuck in a local minimum and doesn't find the best strategy, then it's too easy for the next generator iteration to find the most plausible output for the current discriminator.

3. The robots did not have representation into the synthetic images. The reason for that, is that I originally thought that including robots would increase too much the complexity of the image, therefore inevitably leading the generator to mode collapse. Nevertheless the robots did have some kind of representation (usually white as the soccer field lines), therefore trying to give them a unique correspondence in the synthetic images might have helped the representation.

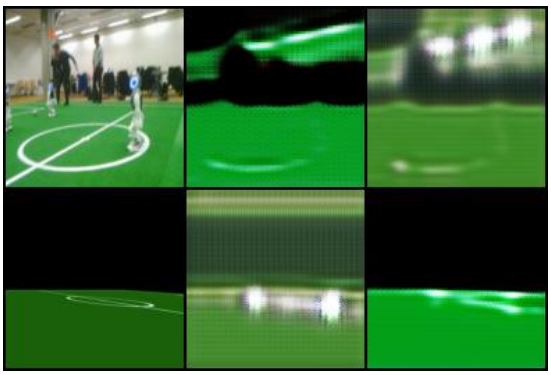
4. Some of the more shaded areas of the pictures were learned by the cycle-GAN as background and turned into black as shown in 3c.

Point 2) and 4) suggested that dark green and black where too similar, they often managed to fool the generator into considering them the same color. This could suggest that changing the background to another color would attenuate the mode collapse. Even though this might seem a good intuition, I found that changing the background to gray did not increase the quality of the translation, but it rather worsened the mode collapse effect. Some other unsuccessful attempts were to vary different green shades within the synthetic image set and introducing squared grey "robots" inside the synthetic representation.

Another remarkable observation was that increasing the resolution of the image would worsen of some amount the outputs of the Cycle-GAN (see Figure 4). One reasoning for that could be that the higher resolution images would decrease the variability of the overall image, reducing the gradient nearly to zero while computing the loss. To maintain the benefits of having sharper synthetic images, without the differentiability problem, one idea was to insert some noise. The disturbance could be injected by randomly increase or decrease the pixel intensities by a certain amount, but also gradually increase the intensity bottom to top as to maintain the bottom (more meaningful) nearly unaltered, while significantly varying the top. (Ref. Figure 5)



(a) to be compared to 3a



(b) to be compared to 3b

Figure 4: Results of higher resolution synthetic images

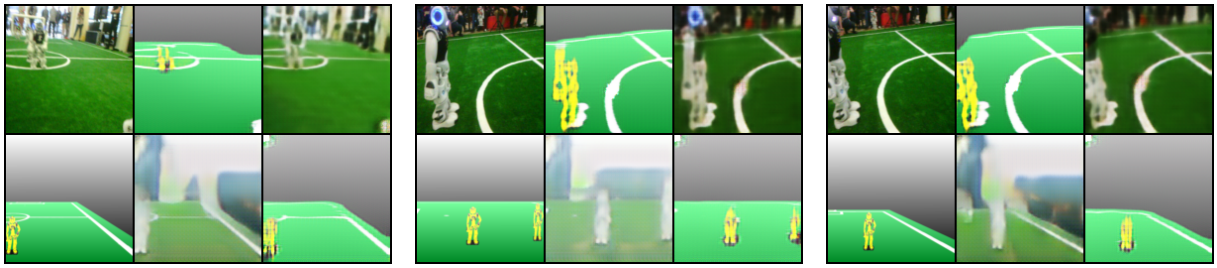


Figure 5: Results obtained by inserting yellow robots representation into synthetic images

3.2 Generating Robots

As presented before, one of the problems was that, even though the robots did not appear in the synthetic representation, they were still reconstructed by the Cycle-GAN and mapped to other feature colors. Mapping the robots naively to gray squares was completely unsuccessful since the gray squares were not recognized by the Cycle-GAN as robots. The reason was clearly that the robots where too different from their representation. Next, I tried to recreate a more likely representation by projecting few real robot images at random in the field. As shown in Figure 5 the Network was now able to identify and reconstruct the robots to some extent. The code snippet to generate robots can be found in the GitHub repository [11] under the path */generate_synthetic_images/trainBmain_w_robots.py*.

4 6-dof Pose Estimation

PosNet, the network to regress pose from images, was already implemented on GitHub [6]. The architecture was composed of an underlying core network, originally *GoogLeNet*, and a final fully connected layer regressing the 6-dof pose. I changed the base network to *Resnet34*, already implemented on GitHub [9] (code: [11] at path "*/visloc-apr*").

4.1 Model for Deep Regression of Camera Pose

The chosen model for deep regression of camera pose was already presented in section 1: the form was $(\delta(x), \delta(y), \delta(z), q_1, q_2, q_3, q_4)$ where the first three values represented the displacement in the respective directions, while the last four were measure of rotation, namely quaternions. As explained in the paper [1], in order to regress pose, the convnet was trained on Euclidean loss using stochastic gradient descent with the following objective loss function:

$$loss(I) = ||\hat{x} - x||_2 + \beta||\hat{q} - \frac{q}{||q||}||_2, \tag{2}$$

where β is a scale factor chosen to keep the expected value of position and orientation errors to be approximately equal.

4.2 The Training-Testing Flow

As mentioned above, the whole structure was built upon a core network: ResNet 34. The input images were the same synthetic images initially fed to the Cycle-GAN, while the testing images were real to synthetic images, translated by the Cycle-GAN. Together with the input images, the real pose had to be provided to the network for the supervised training session. The "training images" where fed to the core block, the 6-dof pose was regressed, the values were compared to the real ones and the validation loss was computed. After a certain amount of epochs, the distances in homo-meters and in angle-degrees, for displacement and rotation respectively, were computed and the final results saved. This whole process was repeated over and over until the end of preset number of epochs.

Epochs	Weight Decay	Batch	LR	β	optimizer	weight_init [1]	val_epochs [2]	Load Size
1000	1e-4	75	5e-3	100	*Adam*	True	10	224x224

Table 2: Hyperparameters for PoseNet Implementation

4.3 The metric

The metric of result evaluation was hard to find. The network was trained on synthetic images, while tested on synthetic to real translated images. Those images didn't have any pose annotation, thereby the only way to compare the position regressed and the translated image was to generate synthetic images having the regressed pose parameters and compare them with the translated images.

4.4 Some Results

The results did not meet the expectations: the translation and rotation values were often off, even though the validation accuracy in the synthetic dataset often reached very high values (over 70% of the regressed poses where below a threshold of 2 meter displacement and 5 degrees angle from the real ones). The obvious reason was that the network was not training on the translated images, but on the synthetic ones, therefore the accuracy in predicting their poses was not as high as expected. (Some results are shown in Figure 6)

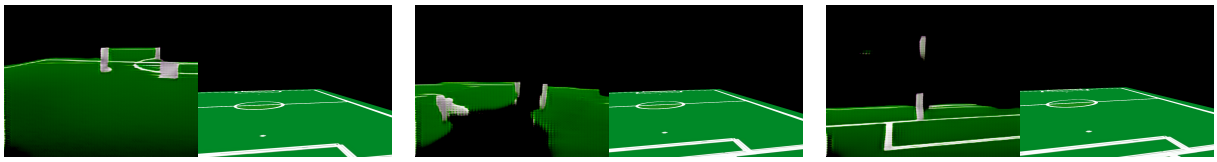


Figure 6: Results obtained by Posenet

5 Network Architecture

As suggested in the previous section the poor performance of my implementation of Posenet was due to the fact that the network was trained on synthetic images, but tested on Cycle-GAN translated images. Furthermore the synthetic images provided to the network were generated with similar focal length and similar displacement along y (height), but the dataset of real images contained a lot of more variability. Most of the images could not be correctly processed because there was no correspondence to their real pose in the network training data. To wave this issue, I introduced a self-supervised module. (GitHub repository [11] under the path */new-architecture/*)

[1] Weight initialization by GoogLenet places googlenet_places.extract.pth
[2] Number of Epochs til validation

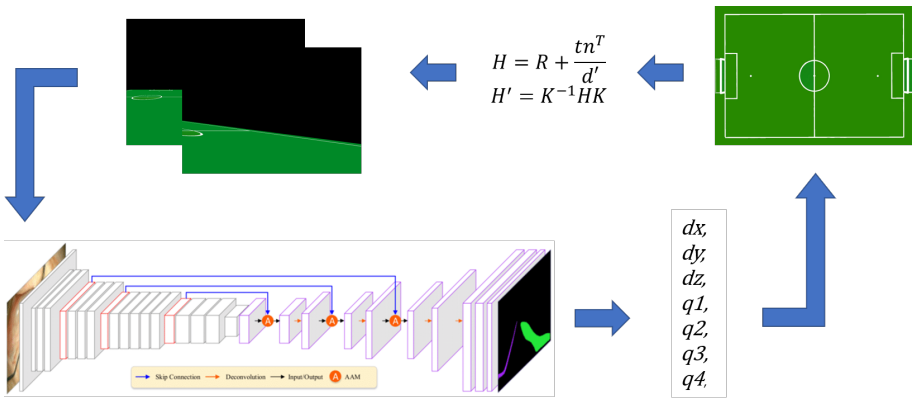


Figure 7: New Architecture: Posenet + Self-Supervised Module

5.1 The Training-Testing Flow

The training and testing are depicted in the diagram of Figure 7. The synthetic images and their poses were fed to the main block of the architecture. For the validation step I had two different loss functions: the first, repeated every epoch, was the one between the synthetic images predictions and their real poses, while the second, repeated every few epochs, was where the self supervised module came into play. The real images were fed to the network, their poses computed, and the corresponding synthetic images generated. Finally, the second loss was computed between each real image and the corresponding newly generated synthetic image.

Epochs	Weight Decay	Batch	LR	Every#epoch	optimizer	Loss-Func	Load Size
1000	1e-4	75	2e-4	5	Adam	MSE/L1	224x224

Table 3: Hyperparameters for Self Supervised Module

5.2 The Loss Function of the Self-Supervised Model

The loss function was a very delicate matter. As matter of facts the images to be compared did come from two very different domains. The main objective was to take into consideration only the features truly important for the localization, otherwise the variability of the real images (presence or absence of ball/robots/public, lighting etc ...) would completely overtake the loss function. The images had to be preprocessed using differentiable tools such as filters. Few of the different combinations were:

1. **MSE Loss**
2. **L1 Loss**
3. **L1 Loss + 2D Gaussian Blur + Sobel Filter**: The gaussian blur smoothed out the real images and made them more uniform, while the Sobel filter underlined the contour of the figures and therefore the lines of the field.

The results of the L1 Loss and the mixed approach were the most satisfying, some results are reported in Figure 8

6 Conclusions

The self-supervised module was successful in predicting from real images regardless of their intrinsic variability. The results show how the focal length and the height varies within the predictions. The model is still not perfect, lots of images are off, probably because the loss functions found are still not the best possible choice in detecting the main features through domains. If I were to keep working on this project I would probably spend some time in searching more effective loss functions and in testing their accuracy both on the real images and Cycle-GAN translated images.

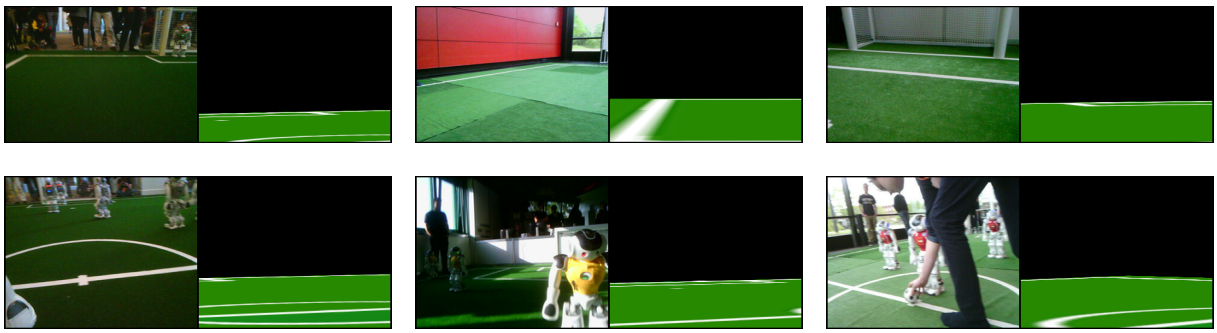


Figure 8: Results obtained by Posenet + Self-Supervised Module with L1 Loss (top)
Results obtained by Posenet + Self-Supervised Module with L1 Loss + Gaussian Blur + Sobel Filter (bottom)

References

[1] Roberto Cipolla Alex Kendall Matthew Grimes. "PoseNet: A Convolutional Network for Real-Time 6-dof Camera Relocalization". In: *arXiv* (2015).

[2] Frederik Hvilshøj Alexander Mathiasen. "Backpropagating through Fréchet Inception Distance". In: *arXiv* (2009).

[3] arnab39. *cycleGAN-PyTorch*. `https://github.com/arnab39/cycleGAN-PyTorch`. 2018.

[4] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[5] Jun-Yan Zhu Taesung Park Phillip Isola Alexei A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *ArXiv* (2017).

[6] GrumpyZhou. *visloc-apr*. `https://github.com/GrumpyZhou/visloc-apr`. 2019.

[7] Sebastian Lunz Jonas Adler. "Banach Wasserstein GAN". In: *arXiv* (2019).

[8] Tinghui Zhou Matthew Brown Noah Snavely David G. Lowe. "Unsupervised Learning of Depth and Ego-Motion from Video". In: *arXiv* (2017).

[9] pytorch. *vision*. `https://github.com/pytorch/vision`. 2019.

[10] Radu Timofte Samarth Shukla Luc Van Gool. "Extremely Weak Supervised Image-to-Image Translation for Semantic Segmentation". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* (2020), pp. 3368–3377.

[11] serenabono. *Semester Project*. `https://github.com/serenabono/Semester-Project`. 2021.

[12] Wanggen Wan Xiao Xiao. "Human pose estimation via improved ResNet50". In: *4th International Conference on Smart and Sustainable City (ICSSC 2017)* (2018).