6D Pose Autoencoder

Project report Reconstructing and Understanding the 3D World
at the IWR Institute in Heidelberg

submitted by

Till Bungert and Carsten Lüth

under the supervision of

Prof. Dr. Carsten Rother
and
Siva Karthik Mustikovela

2018

Contents

Abstract

Lorem Ipsum add abstract or not

1 Introduction

The determination of 6D poses for known objects is a very important subject for robotics and augmented reality. For example in robotics it is crucial to have an accurate understanding of the position of an object to interact with it in a meaningful way. Almost the same applies to augmented reality because to enhance any object by rendering information on its visible surface it is important to know the position of the said object.

There are many different ways to determine the 6D-Pose of a known object but in this work we will concentrate on a method which is based on the Augmented Autoencoder as designed by Martin Sundermeier et al XXX. This approach only uses RGB Images and does not require a big amount of labelled data to be trained because the autoencoder is tasked to reconstruct its input and not to predict the rotation and translation directly. This implicit learning makes it very robust against symmetries. Also in their paper they show that it can be fully trained on artificial generated images and also able to generalize to real images.

The general goal of this work is to use the augmented autoencoder to disentangle the rotation and translation information in the latent space with a novel training scheme as well as to give an theoretical motivation for the rotation learning of the network which gets backed up by our experiments.

2 Methods

In the following section we focus on the novel 6D-Autoencoder which is a new interpretation of the Augmented Autoencoder (AAE), introduced by Martin Sundermeier et Al. (XXXXXXXXX). First we introduce the concept of invariant latent space representations of an Autencoder via objective functions and then present a method to get the 6D-Pose based on the latent space representation.

2.1 Autoencoder

The original Autoencoder is a technique to map a high dimensional input (x), such as image or audio data, into a lower dimensional representation (z) ($D << N$) while maintaining the important information of its input. To achieve this the model which consists of an Encoder φ and Decoder ϕ is tasked to reconstruct its input after passing it through a low dimensional bottleneck (z), where each of them is a general function approximator (neural network), resulting in the following objective function:

$$\min_{\varphi,\phi} L_{\text{rec}}(x, \phi(\varphi(x))) \tag{1}$$

L_{rec} can be any arbitrary distance metric on $\mathbb{R}^N$. A specialized form of the Autoencoder is the Denoising Autoencoder. For this type of Autoencoder the input x gets pertubed with random noise $\tilde{x} = f_{\text{noise}}(x) = x + \epsilon \ \epsilon \sim \mathbb{P}_\epsilon$ while beeing tasked to reconstruct the unpertubed x. This results in the following objective function:

$$\min_{\varphi,\phi} L_{\text{rec}}(x, \phi(\varphi(f_{\text{noise}}(x)))) \tag{2}$$

The trained model can be used to reconstruct unpertubed test images. But how is the latent space effected by this training schedule? This question leads to the core assumption to create the AAE which gets motivated by a later experiment.

Hypothesis 1 To optimally reconstruct the unpertubed x for the previously unseen test data the encoder should get in theory invariant to the pertubation. Meaning: Encoded Latent space representation of the input is invariant against the perturbation.

Regularized Autoencoder The regularized Autoencoder as described in XXXDeepLearningBookXXX is defined by an additional loss/ part of the objective function which is solely effected by the latent space representation. By adding this additional loss the latent space gains some desired properties such as a specific distribution with the Variational Autoencoder by using the KL-Divergence as latent loss. Resulting objective function of a regularized Autoencoder:

$$\min_{\varphi,\phi} L_{\text{rec}}(x, \phi(\varphi(f(x)))) + L_{\text{lat}}(\varphi(f(x))) \tag{3}$$

f can be any arbitrary augmentation or just the identity.

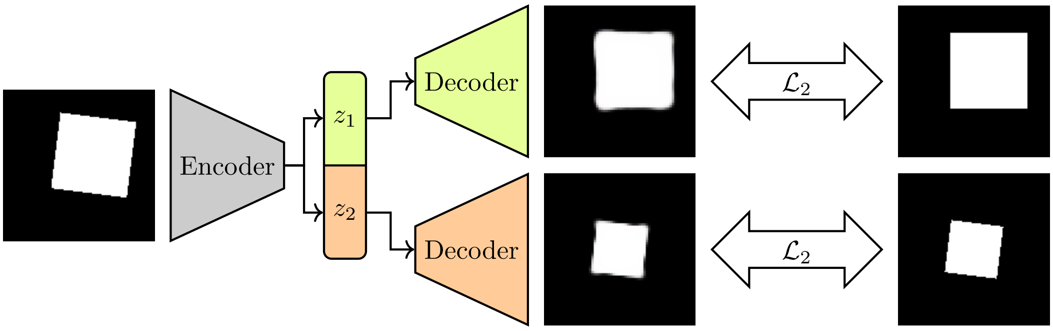

Figure 1: The Model

2.2 6D Pose Autoencoder

The motivation of the 6D-Pose-Autoencoder is to get two latent space z_{rot} and z_{trans} from which the 6D pose comprised of rotation and translation can be read out separately. Therefore z_{rot} encodes the orientation and should be invariant against translation while z_{trans} encodes the translation while being invariant for rotations. This is done by taking two separate Augmented Autoencoder, one for the translation and one for the rotation. They receive the same input but are trained to reconstruct differently augmented versions of the input x_{rot} and x_{trans} . The training schedule is inspired by **Hypothesis 1**. This is achieved by applying random augmentations f_{aug} comprised of a f_{trans} and f_{rot} on the input x.

$$\begin{aligned} \tilde{x} =& f_{\text{aug}}(x) = (f_{\text{trans}} \circ f_{\text{rot}})(x) \\ x_{\text{rot}} =& f_{\text{rot}}(x) \\ x_{\text{trans}} =& f_{\text{trans}}(x) \\ \hat{x}_{\text{rot}} =& (\phi_{\text{rot}} \circ \varphi_{\text{rot}} \circ f_{\text{aug}})(x) \\ \hat{x}_{\text{trans}} =& (\phi_{\text{trans}} \circ \varphi_{\text{trans}} \circ f_{\text{aug}})(x) \end{aligned} \tag{4}$$

The objective function of the system thereby is:

$$\begin{aligned} \min_{\varphi_{\text{rot}}, \phi_{\text{rot}}} & L_{\text{rec}}(x_{\text{rot}}, \phi_{\text{rot}}(\varphi_{\text{rot}}(f_{\text{aug}}(x)))) \\ \min_{\varphi_{\text{trans}}, \phi_{\text{trans}}} & L_{\text{rec}}(x_{\text{trans}}, \phi_{\text{trans}}(\varphi_{\text{trans}}(f_{\text{aug}}(x)))) \end{aligned} \tag{5}$$

For the final model $\varphi_{\text{rot}} = \varphi_{\text{rot}}` \circ \varphi$ and $\varphi_{\text{trans}} = \varphi_{\text{trans}}` \circ \varphi$ so they are encoded by the same encoder and z_{rot} and z_{trans} are just created by a split int its latent space. The complete architecture can be seen in figure XXXXX.

Rotation A rotation in one axis can be represented by a circle in $\mathbb{C}$ parametrized by $\theta \in [0, 2\pi]$ with $\exp(i\theta)$ the choice is near to define $z_{\text{rot}} \in \mathbb{R}^2$ and using as $L_{\text{lat}}(z) = \text{abs}(||z_{\text{rot}}||_2^2 - 1)$ which forces the encoded z_{rot} to be on the unity circle.
Rotations with three axes can be represented in a similar manner as hyper sphere in $\mathbb{H}$. So for three rotational axes we defined z_{rot} to be in $\mathbb{R}^4$ which is a generalization of $\mathbb{H}$ with the same latent loss as stated above.

Translation The translation of the object is a simple regression problem with uncorrelated factors. So we chose the simplest method to obtain a dense feature space in $z_{\text{trans}} \in \mathbb{R}^3$ by taking as $L_{\text{lat}}(z) = z_{\text{trans}}^2$. This is done because traditional autoencoders on mnist for example often have translations as single latent dimensions. An interesting fact is that the z or depth axis could also be parametrized by scale so for objects with similar shape but different scales there would also be some ambiguities.

2.3 Codebook

The 6D-AE is after training able to extract the 3D orientation in z_{rot} and the translation in z_{trans}. The clarity, orientation and translations of the reconstructions are a first indicator of the quality of the encodings. To extract the orientation and translation from the latent space representations we create a codebook for both rotation and translation.
This is done in the following way:

1. Render synthetic objects with known translation and rotation

2. Create a codebook by encoding the rendered images to $z_{\text{rot}} \in \mathbb{R}^{O-1)}$ and $z_{\text{trans}} \in \mathbb{R}^3$ where each of those belongs to a specific rotation and translation with respect to the camera

The next step is to utilize the K-Nearest-Neighbour algorithm separately on the two latent space representations of the test images with unknown labels compared to the codebook with known labels.
The Distance metric used for the rotations z_{rot} is the cosine similarity. This is done because we initially made the assumption that our representations lie in a hyper sphere with unit distance from the origin.

$$\cos_i = \frac{z_i \cdot z_{\text{test}}}{||z_i|| \cdot ||z_{\text{test}}||} \tag{6}$$

For the translation the euclidean distance was chosen as distance metric to determine the next nearest neighbours.

3 Evalutation

In the following section we present our results to prove our concepts presented in the chapter methods 2. XXXXXXXXXXXXXXXXXXXXXXXX

3.1 2D Square in 3D Space

The goal of this experiment was to reproduce the results of Martin Sundermayer et al. in XXXXXXX with the AAE and show that the rotation and translation can be disentangled in principle with this very easy object. z_{rot} was defined to be in in $\mathbb{R}^2$ representing $\mathbb{C}$ as described in chapter 2 due to the single rotation axis of the object. Due to the two perpendicular axes of symmetry the square posses we should see a shifted and cosine function with the frequency $f = \frac{4}{2\pi}$, because the square appears exactly the same after a rotation of $\frac{\pi}{2}i$.

The second part was to show that the translation could be extracted from the latent space representation z_{trans} and that different rotations have neglegible effect on it. The resulting plot for the correlation between z_{rot} and the rotation φ as well as z_{trans} and the translation can be seen in figure XXXXXXXX.

3.2 3D Cube

In this experiment we wanted to expand our predictions to a more complex case by taking the next complex object from the 2D square in 3D, namely the 3D cube. the 3D cube can be rotated in 2 axes due to trivial reasons of symmetry which leads to the conclusion that the rotations should be able to be represented by a hyper-sphere in 3 dimensions. So z_{rot} was chosen to be element of $\mathbb{R}^3$. The correlations between z_{rot} and the rotations as well as z_{trans} and the translation can be seen in figure XXXXXXXX. Due to the more complex nature of the correlations in this case we also used the codebook created with our fully trained model to show the capabilty of this architecture. The resulting errors can be seen in table XXXXXXXXXXXX.

3.3 Line Mod

Our final experiments were conducted on the LINE MOD dataset from XXXXXXXXXX where we decided to evaluate the capability of this model on two very different objects. Namely the cat and the egg-box where the model should show that it can be used on cases with no symmetry like the cat as well as many symmetries such as the egg-box in general cases with all 3 rotations and thereby the resulting latent space representations $z_{\text{rot}} \in \mathbb{R}^4$. Ideally the resulting model can be used to determine the 3 rotation values as well as to be able to detect rotation symmetries like in the egg box by itself making it very robust against ambiguities.

3.3.1 Cat

As mentioned before the cat object of the line mod dataset has no symmetries which makes it a good proof of concept to show that the 3 rotations can be displayed by the network on the unity sphere in $\mathbb{R}^N$ which makes it a generalization of the hamiltonians $\mathbb{H}$. Also this allows us to test whether this assumption really fits by calculating the resulting i, j and k of two rotations and applying (multiplicating) them onto each other. The resulting quaternion is then compared with the quaternion which we get by computing the quaternion we get from the network predicted.

3.3.2 Egg Box

The Egg Box has many symmetries and the ideal case is that the network learns these symmetries by changing the frequencies of the latent space repetitions in z_{rot}

4 Conclusion

5 Sources

*