Spatial Transformer Networks: An application to hand alignment task

Niccolò Bellaccini
niccolo.bellaccini@stud.unifi.it

Tommaso Aldinucci
tommaso.aldinucci@stud.unifi.it

Abstract

In this paper we present an application of Spatial Transformer Networks (STN) module to hand alignment task. We used a depth camera to represent hand images in order to obtain a better result. We developed a STN that learns the parameters of a transformation which aims to align each input hand image with a certain target hand.

Experiments show good results in terms of model accuracy and visual perception.

Future Distribution Permission

1. Introduction

Traditionally *Convolutional Neural Networks* (CNNs) are able to make your model invariant to image transformations (*e.g.* affine transformation) only when the training set contains a lot of examples made properly.

Spatial Transformer module can be included into a standard neural network architecture to provide spatial transformation capabilities. Unlike pooling layers, the STN module is a dynamic mechanism that can actively spatially transform an image (or a feature map) by producing an appropriate transformation for each input sample.

As described in [2] STN is a differentiable module which applies a spatial transformation to a feature map during simple forward pass; it is made by a combination of three main components: a localization network, a grid generator and a sampler. Figure 1 shows a sketch of the architecture of a spatial transformer module.

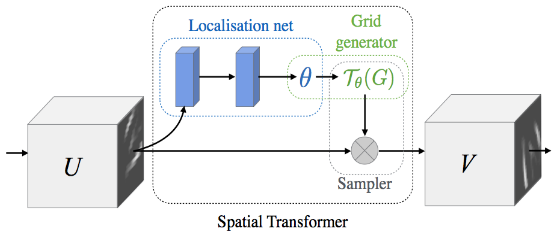


Figure 1: Architecture of a spatial transformer module

For our purposes, we are not interested to STN as an additionaly module to be included in a CNN, but rather as a stand-alone module outputing the parameters of an affine (or projective) transformation which will be applied to our images. Each image of our dataset represents a human hand in some gesture positions.

A *Creative Senz3d camera* was used to acquire each hand-image as a depth map where pixels represent the distance between the depth sensor and the physical object in the scene. In this way we were able to separate some visually interesting areas (like a hand) from the rest and, since each pixel value was encoded as a 16-bit unsigned int, we have more information per pixel than a standard 8-bit grayscale image.

So, the goal of this project is the development of a STN which aims to transform a depth map of a hand into a depth map representing the same input aligned with a target image, through an affine or projective transformation.

In the next section we report the approach used to resolve the task mentioned before.

2. Method overview

The workflow was arranged mainly in this way: depth stream capture, hand dataset creation, STN development and finally STN perfomance evaluation. Figure 2 summarize our proposed procedure.

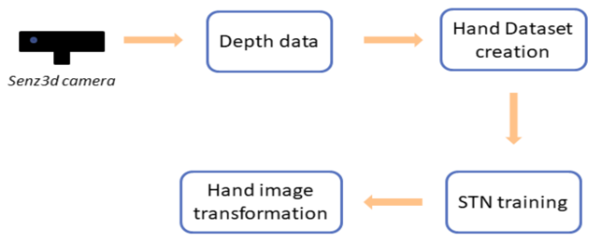


Figure 2: Proposed solution pipeline

2.1. Dataset creation

The dataset is composed by a large set of depth map images which represent, as much as possible, a hand in some kind of configuration. Each depth map was acquired capturing frame by frame the depth stream of the Senz3d camera.

In order to create a consistent and diversified dataset we recorded many depth streams of both hands (left and right) placing each hand in front of the depth sensor and attempting to identify several open hand positions. A correct fitting of our ST-CNN was achieved normalizing the dataset making each pixel value in [0,1]. We would like also to give more prominence to values near 1 (white) which will correspond to near object in the image.

To do this we implemented a thresholding method. One of the big advantage of using a depth map image instead of grayscale (or RGB) image concerns the detection of interest areas. Because each depth stream was acquired moving the hand in front of the depth sensor, we decided to zero (*i.e.* set to black color) all the pixels whose values were strictly greater than the minimum of the regarding image depth value plus 300.

Formally, given an input depth map $I : N^2 \to N$, the related output grayscale image $G : N^2 \to N$ is such that

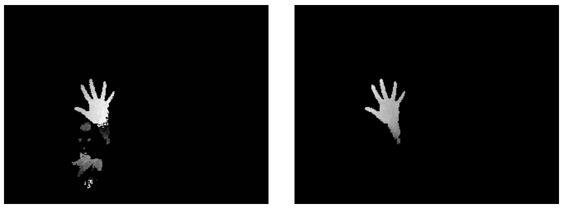
$$G(x,y) = \begin{cases} I(x,y) & \text{if } I(x,y) \le \mu + 300 \\ \mu + 300 & \text{otherwise} \end{cases}$$

where $\mu := \min_{(x,y)} I(x,y)$ represents the nearest point to the camera.

This allowed us to select approximately only the hand region of the person in front of the senz3d camera.

Next we normalized to [0,1] subtracting the minimum of the image and simply dividing by 300 (approximately the value range of the interesting area) and finally we inverted each image as we wanted 1 to be the value representing the least distance.

Figure 3 illustrates an example of this procedure.



(a) *Pre processing* (b) *After thresholding*

Figure 3: Depth map samples

We also implemented a data augmentation method to increase and diversify our dataset applying random translation, rotation and scaling. The target images were manually selected from a target reference frame, based on visual informations for each hand in the dataset.

Next subsection will clarify the reasons of what we sad before.

2.2. STN development

The architecture of STN was achieved dealing our problem as a regression problem. For this reason the localisation network input was composed by a number of neurons equal to the dimension of each dataset image (320 x 240) and likewise for output layer.

Input neurons assume values of some dataset depth maps while output neurons take values of the target depth map relates to that input. Figure 4 shows a localisation network summary plot.

The localisation network builded is a 223K parameters CNN formed by 3 convolutional 3x3 layers with 20 filters each, stride 1 and same padding for no dimension reduction.

We interponed a max pooling 2x2 layer between each convolution to iteratively downsample the feature map.

Furthermore each convolutional layer has an identity activation function that is in general recommended for Spatial Transformer Module. Finally a fully connected layer outputs the parameters of a transformation.

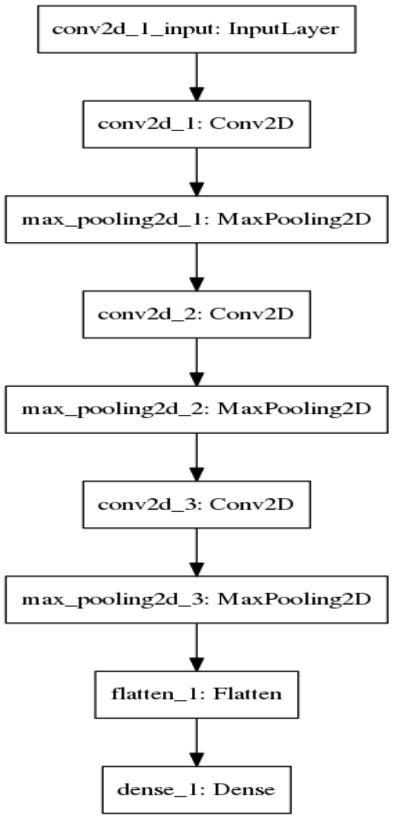

conv2d_1_input: InputLayer
conv2d_1: Conv2D
max_pooling2d_1: MaxPooling2D
conv2d_2: Conv2D
max_pooling2d_2: MaxPooling2D
conv2d_3: Conv2D
max_pooling2d_3: MaxPooling2D
flatten_1: Flatten
dense_1: Dense

Figure 4: Localisation network model

This network predicts 9 transformation parameters (representing affine or projective matrix) which transform a regular grid into another sampling grid used to get output image sampled from the input at the grid points.

3. Implementation details

In this section we report some programming details about our procedure.

3.1. Dataset creation

Each depth map image of the dataset was obtained through the *DS325 depthsense* SDK using Windows 7. We wrote a c++ program for real time depth stream acquisition through the use and setup of the previous SDK and OPENCV library.

This lead us to save each depth frame image as a 16 bit per pixel grayscale PNG image of dimension 320 x 240 pixels. Other settings, like frame rate and filter applications, were set (using the SDK) to get optimal results.

3.2. STN implementation

The localisation network was implemented as a convolutional neural network through the use of *Keras* library. The entire STN module was assembled with a *SpatialTransformer* layer implemented in Keras and using Tensorflow as backend. This layer [1] uses as localisation network the previous CNN implemented.

As regression problem we reasonably chose *mean squared error* as loss function and *SGD* (with learning rate $\alpha = 10^{-2}$ and momentum $\rho = 0.9$) like optimizer.

In order to read each dataset image as 16 bit per pixel we used OPENCV library.

To avoid *out of memory* problem and for a better visualization we used the keras *fit_generator* method that allows the training of very large dataset. During the training process we saved for an input image (on epoch begin), the sampling grid and the image transformed. Figure 5 displays a sample of this kind of image.

Finally a gif image can be created attaching each image one by one (in order of growing epochs) showing the entire learning process.

4. Results

Experimental results were verified during the training process on a validation set and later on

[1]https://github.com/oarriaga/spatial_transformer_networks

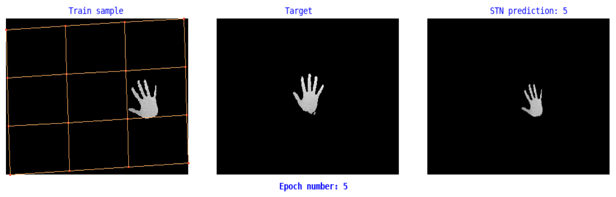


Figure 5: STN training sample

a test set with the same format type as train set. Figure 6 shows the trend of the loss function over 50 epochs highlighting a progressive decrease of both functions.

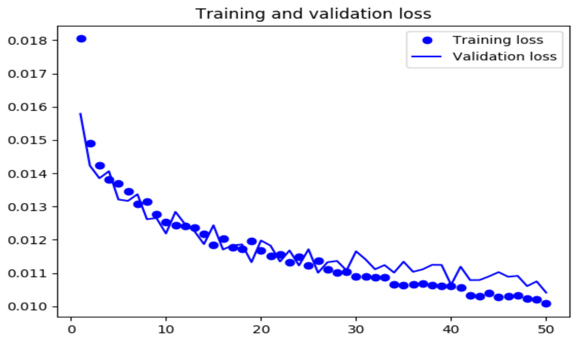
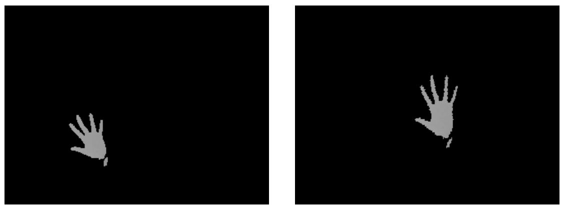


Figure 6: STN loss functions

The proper operation of training process was visually evaluated over some test set images.

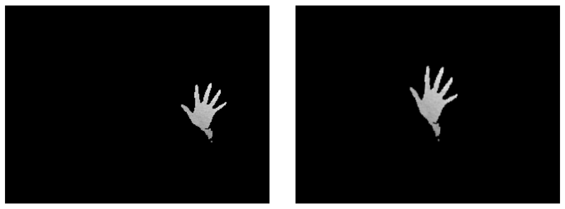
Below some of that examples are reported.



(a) *STN input* (b) *STN output*

Figure 7: STN working example

All above reported results were obtained on a machine with an Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz, a 62 GiB of RAM and a NVIDIA GeForce GTX 970 as graphic card.



(a) *STN input* (b) *STN output*

Figure 8: STN working example

5. Conclusions

This paper revealed that STN module could produce good results on a not trivial task like hand alignment. Localisation network was builded as a CNN based on STN reference articles. The depth information given by the Senz3d camera have made possible individuation of the hand in the image context thanks to the simple thresholding method that we have applied. This approach allowed the achievementof meaningful results in term of visual perception.

References

[1] F. Chollet. *Deep learning with python*. Manning, 2017.

[2] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.