Spatial Transformer Network application: A hand alignment case study

Niccolò Bellaccini
niccolo.bellaccini@stud.unifi.it

Tommaso Aldinucci
tommaso.aldinucci@stud.unifi.it

Abstract

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

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

Future Distribution Permission

The authors of this report give permission for this document to be distributed to Unifi-affiliated students taking future courses.

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 [1] 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 [1] 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 skecth of the architecture of a spatial transformer module.

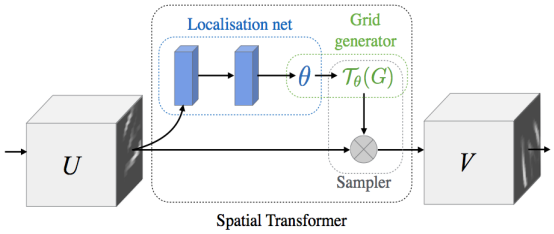

Figure 1: Architecture of a spatial transformer module

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 outputs the parameters of an affine (or projective) transformation which will be applied to our images. Each image of our dataset represent a human hand in some kind of stance.

Creative Senz3d camera was used to acquire every hand-image as a depth map where each pixel represent the distance between the depth sensor and the physical object in front of it. In this manner we were able to separate some visually interesting areas (like a hand) from the rest and, since that each pixel value was encoded as a 16-bit unsigned int, we could increase the ”information rate” per bit.

So, the goal of this project is the development of a STN which aims to transform a hand-input depth map into a hand-target depth map through the parameters learning of some transformation.

In the next section we report our approach used to resolve the task mentioned earlier then, next section explains the implementation details about our method and finally STN experiment results

are evaluated.

2. Related work

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.

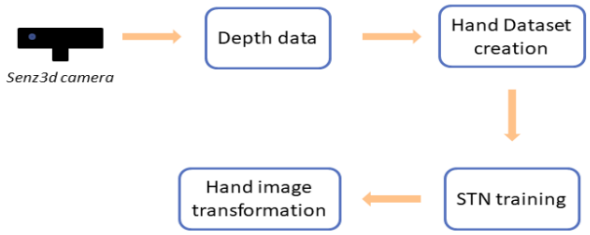

Figure 2: Proposed solution pipeline

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 hand image 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 idenify several open han positions.

One of the big advantage of using the depth map image instead of grayscale (or RGB) image concerns the detection of interest areas. Since that 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 gray intensity value plus 300 (corresponding to 30 cm). Formally, given an input grayscale image $I : N^2 \rightarrow N$, the related output grayscale image $G : N^2 \rightarrow N$ is such that

$$G(x,y) = \begin{cases} I(x,y) & \text{if } I(x,y) \leq \mu + 300 \\ 0 & \text{otherwise} \end{cases}$$

where $\mu := \min_{(x,y)} I(x,y)$

This allow us to select approximately only the hand region of the person in front of the senz3d camera. Figure 3 illustrates an example of this procedure.

(a) *Before preprocessing* (b) *After thresholding*

Figure 3: Depth map samples

For the purpose of STN training every time a stream was acquired we manually selected the target reference frame based on visual informations. Next subsection will clarify the reasons of what we have just said.

2.2. STN development

The architecture of STN, especially the *localisation network* 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 depth maps dataset image 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.

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

TODO: small values and homogeneous values?

3. Implementation details

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

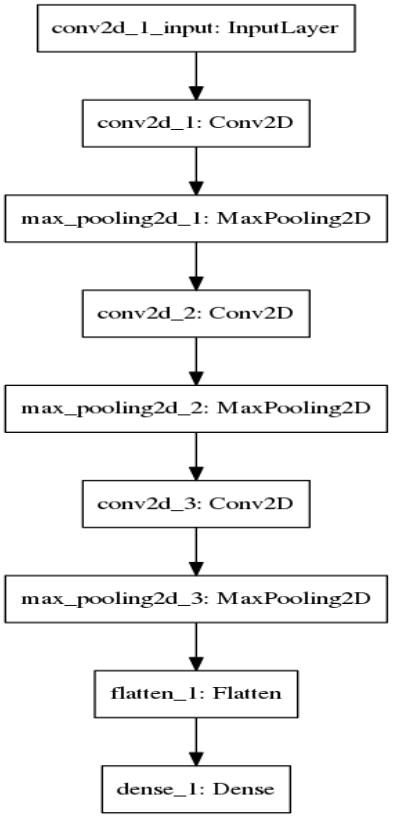

Figure 4: Localisation network model

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 240 x 320 pixels. Other settings, like frame rate and filter applications, were set (using the SDK) to get optimal results.

TODO: binary stream

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 the *SpatialTransformer* layer [ref?] which take as localisation network the previous CNN defined.

As regression problem we reasonably choose *mean absolute error* as loss function and *adam* like optimizer.

In order to read each dataset image as 16 bit per pixel we used OPENCV library and (...) To avoid interpolation problem and for a better visualization we use 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.

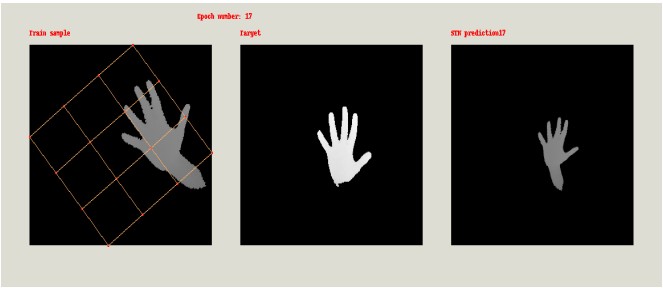

Figure 5: STN training sample

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

References

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