Determining the Orientation of Low Resolution Images of a De-Bruijn Tracking Pattern with a CNN

Andreas Schmid
University of Regensburg
Regensburg, Germany
andreas.schmid@ur.de

Stefan Lippl
University of Regensburg
Regensburg, Germany
slippl@deloitte.de

Raphael Wimmer
University of Regensburg
Regensburg, Germany
raphael.wimmer@ur.de

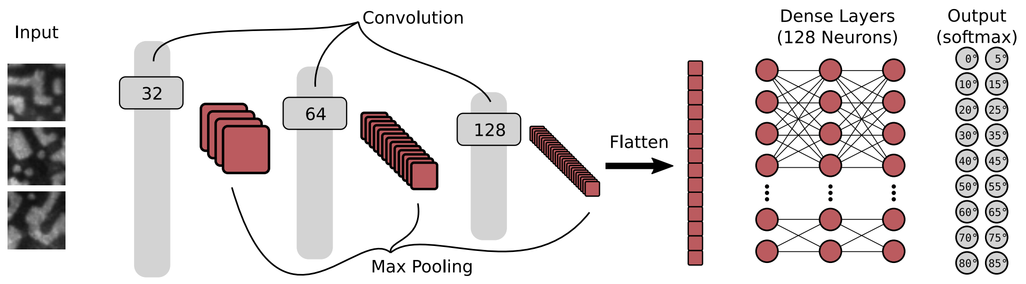

Figure 1: Network architecture. Three 2D convolutional layers (32, 64, 128; 3×3 kernel; linear activation), each followed by a max pooling layer (2×2). The result is flattened and passed to three dense layers (128 neurons eachs, linear activation). A last dense layer with 18 neurons and softmax activation serves as a classifier for the angles.

ABSTRACT

Inside-out optical 2D tracking of tangible objects on a surface oftentimes uses a high-resolution pattern printed on the surface. While De-Bruijn-torus patterns offer maximum information density, their orientation must be known to decode them. Determining the orientation is challenging for patterns with very fine details; traditional algorithms, such as *Hough Lines*, do not work reliably. We show that a convolutional neural network can reliably determine the orientation of quasi-random bitmaps with 6×6 pixels per block within 36×36 pixel images taken by a mouse sensor. Mean error rate is below 2°. Furthermore, our model outperformed *Hough Lines* in a test with arbitrarily rotated low-resolution rectangles. This implies that CNN-based rotation-detection might also be applicable for more general use cases.

KEYWORDS

CNN, Computer Vision, Tracking Pattern

ACM Reference Format:
Andreas Schmid, Stefan Lippl, and Raphael Wimmer. 2022. Determining the Orientation of Low Resolution Images of a De-Bruijn Tracking Pattern with a CNN. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '22 Posters), August 07-11, 2022.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3532719.3543259

1 INTRODUCTION

Reliably tracking the positions of objects (*Tangible Blocks*, Fig. 2) on a tabletop is an important challenge for physical-digital applications in the field of *Tangible Interaction*. Outside-in tracking methods rely on external tracking infrastructure such as cameras or inductive/capacitive sensors. They are therefore prone to occlusion by user's hands and require technical infrastructure. This limitation can be avoided by using inside-out tracking with position sensors integrated into tracked objects. Schüsselbauer et al. [2021] propose a tracking method where a fast, low-resolution (36×36 px) mouse sensor is used to determine the absolute position of tangible blocks on a surface covered with a De-Bruijn torus [Fan et al. 1985] pattern. While the system achieves sub-millimeter resolution, it fails when tracked objects are not aligned with the tracking pattern. In order to determine the object's orientation relative to the pattern and undo the rotation of the captured image, one might apply traditional computer vision algorithms, such as *Hough Line Transformation* [Duda and Hart 1972] or *Iterative Closest Points*. However, we have found that these fail due to the low resolution of the sensor's image.

In this paper, we present a first step for solving this problem with a machine learning model that can determine the rotation of low-resolution dot patterns.

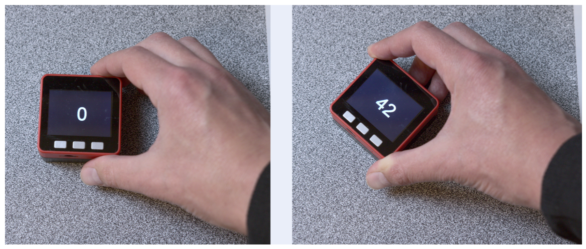

Figure 2: Tangible block on a De-Bruijn-torus tracking pattern. The block's orientation is determined with our model.

SIGGRAPH '22 Posters, August 07-11, 2022, Vancouver, BC, Canada

2 METHOD

We trained a convolutional neural network (CNN) to classify the orientation of a quasi-random dot pattern (Fig. 1). We implemented this model in Python 3.9 using *Tensorflow 2*[1] with the *Keras* API. The network consists of three 2D convolution layers (32, 64, and 128 filters; 3×3 kernel, a stride of one, *same* padding, linear activation), each followed by a 2D max pooling layer (pool size 2×2, *same* padding). After a flatten layer, three dense layers with 128 neurons each and linear activation were added. We found that linear activation performs significantly better than other functions, such as sigmoid and ReLU. Finally, a softmax layer with 18 neurons outputs a probability distribution for each angle (0 – 85° in 5° increments). Hyperparameters of the model were optimized with a manual grid search approach. As the model is only trained with images rotated in 5° increments, in-between angles are interpolated by calculating a weighted average of predictions for each angle. For example, if the model finds a 50% probability for both 35° and 40°, an angle of 37.5° is reported. Training the model with smaller angle increments led to less accurate results.

We collected a training data set of 37800 images (36×36 px each, 1890 for each 5° step between 0° and 85°) by moving a *PixArt PMW3360* mouse sensor[2] over a tracking pattern using an *AxiDraw V3* robotic arm[3]. An additional servo motor rotated the sensor between 0° and 85° in the Z axis. For the tracking pattern, we used a $\{8192, 4096, 5, 5\}$ De-Bruijn torus scaled to 150 pixels per inch and printed at 1200 dpi with a laser printer on glossy paper. Each captured image contains 6.5×6.5 dots of the pattern. Thus, each dot is about 5.5×5.5 pixels in size (Fig 3). This is the same setup as used by Schüsselbauer et al. [2021].

We used 90% of this data set to train our model for three epochs with a batch size of 64, an *Adam* optimizer, and categorial crossentropy as the loss function. Additionally, training data was augmented by rotating each training image in 90° steps. 20% of this training data set were used for validating the model after each epoch. The remaining 10% of the whole data set (233 images per angle) were used for evaluation after the completed training process.

After three training epochs, our model reached 92.98% accuracy on the validation data set (loss: 0.1879) and 88.77% accuracy on the test data set. The final model has 537,746 trainable parameters.

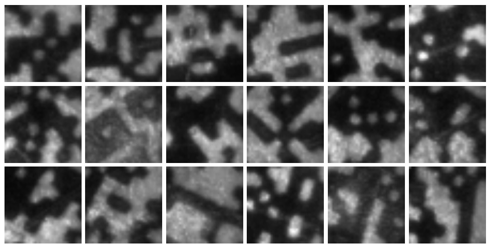

Figure 3: Sample of the training data set. Top row (left to right): 0° – 25°; center row (left to right): 30° – 55°; bottom row (left to right): 60° – 85°.

[1]https://www.tensorflow.org/

[2]Datasheet: http://www.pixart.com/products-detail/10/PMW3360DM-T2QU

[3]https://shop.evilmadscientist.com/productsmenu/846

3 EVALUATION

To evaluate our model's performance, we acquired another data set of 135 images for each 1° angle between 0° and 89° from a different area within the De-Bruijn torus. On average, a prediction using *TensorFlow Lite*[4] took 0.73 ms (sd: 0.23) on our test system[5]. For evaluation images rotated in 1° increments, our model achieved a mean error rate of 1.94° (sd: 1.63, max: 13.41). However, it performed significantly better for only 5° and 10° increments (Table 1).

Table 1: Error rates for different orientation increments.

increment	mean	sd	max	error > 5°	error > 10°
1°	1.94°	1.63	13.4°	4.28%	0.29%
5°	1.27°	1.90	10.8°	3.19%	0.16%
10°	1.25°	1.90	10.0°	3.52%	0.08%

To test whether our model also works for other patterns, we generated 10,000 test images by placing between three and five randomly sized rectangles in a binary 36×36 px image and rotating the result by a random angle between 0° and 89°. We compared the prediction accuracy of our model to the *Hough Lines* algorithm [Duda and Hart 1972] (threshold: 12) which was applied after a *Canny* edge detection (thresholds: 100 and 200). Those parameters were optimized via grid search. Even though our model was not trained on this data set, it attained an error rate of 3.47° (sd: 2.84) - much better than the *Hough Lines* approach (mean error 8.07°, sd: 10.80, and 2.2% of angles unrecognized).

4 CONCLUSION

We have shown that convolutional neural networks can be used to determine the orientation of low-resolution images of dot patterns. As we could achieve high accuracy using only linear activation functions, it might be possible to simplify the model in future iterations. Our CNN could outperform the *Hough Lines* algorithm for a data set of rotated rectangles which it was not trained on. This suggests that the model might also be of use in other domains where low-resolution images need to be analyzed. Source code and data sets are available at hci.ur.de/projects/dottrack.

ACKNOWLEDGMENTS

This project is funded by the Bavarian State Ministry of Science and the Arts and coordinated by the Bavarian Research Institute for Digital Transformation (bidt).

REFERENCES

Richard O. Duda and Peter E. Hart. 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM* 15, 1 (jan 1972), 11–15. https://doi.org/10.1145/361237.361242

C. T. Fan, S. M. Fan, S. L. Ma, and M. K. Siu. 1985. On De Bruijn arrays. *Ars Combinatoria* 19, MAY (1985), 205–213.

Dennis Schüsselbauer, Andreas Schmid, and Raphael Wimmer. 2021. Dothraki: Tracking Tangibles Atop Tabletops Through De-Bruijn Tori. In *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Salzburg, Austria) *(TEI '21)*. Association for Computing Machinery, New York, NY, USA, Article 37, 10 pages. https://doi.org/10.1145/3430524.3440656

[4]https://www.tensorflow.org/lite

[5]*HP EliteBook 850 G4* (*Intel i7* CPU with 2.7 GHz, *Intel HD Graphics 620*, 16 GB RAM