A RECURRENT NEURAL NETWORK SENSORIMOTOR SEMANTICS FOR EMBODIED AGENTS

by

Tatyana Beall

A Thesis Submitted to the Faculty of

The University of Utah

In Partial Fulfillment of the Requirements of the

Bachelor of Science degree

in

The School of Computing

Spring 2017

Approved:

_____	_____
Thesis Supervisor	Director, School of Computing

_____

Date

TABLE OF CONTENTS

ABSTRACT	1
INTRODUCTION	2
BACKGROUND	3
METHODS	4
RESULTS	8
CONCLUSIONS AND FUTURE WORK	12
REFERENCES	12

ABSTRACT

People use languages to store and transfer their knowledge. Embodied agents need their own efficient representation of things they "see" with their sensors to be able to operate effectively in the world. This work gives a framework (called *DISCERNN*) for neural network exploitation which combines computationally exact recurrent neural networks (CERNNs) to form representations (concepts) of shape. A CERNN is not a learned network, but a direct translation of an exact computation into a neural network. The concept of shape, as an output of the network, includes not only information about what the shape is, but also how it was created. To achieve this representation, we used Leyton's group theoretic wreath product [7]. It describes a sequence of actions on a set of points such that the resulting points form boundaries of the given shape. The network combines an agent's perception and actuation systems to determine the types of symmetry present in a shape. The resulting wreath product can then be passed to an agent's motor systems for further use. Classification results from running the network on different shapes are described.

INTRODUCTION

The project's goal is to use sensor (image) and actuation (pan and tilt of an eye, or movement of an arm) data to form both an abstract and operational representation of a shape. By finding point, translational and rotational symmetries, we can get such a representation in the form of the wreath product. A set of recurrent neural networks (CERNNs) was constructed for intermediate steps in finding such symmetries. In turn, the networks can be connected to each other by edges with appropriate weights to form one overall neural net for finding symmetries, given a 2D image of a shape. A mechanism for the composition of CERNN sub-networks into a single executable network is provided. Figure 1 shows a high level view of the resulting network for finding the three types of symmetries.

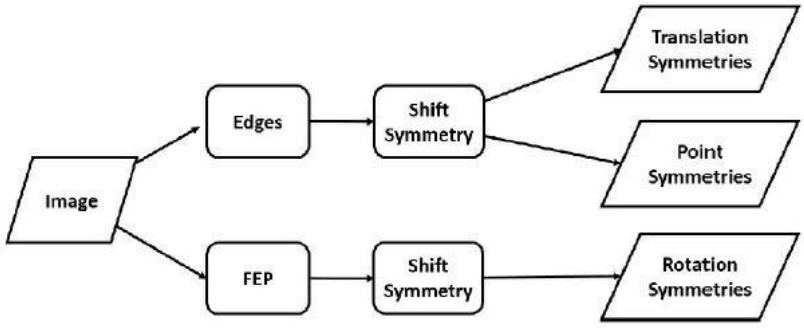


Figure 1: Process of symmetry recognition in an image. *Translational symmetries* are found by first detecting edges of a shape, then shifting the image in the major edge directions and finding unchanged edge pixels. The edges are found by first calculating gradient in each pixel, then magnitude, and finally finding logical *and* of the image and its magnitude in each pixel. *Point symmetries* here are end points of line segments. *Rotational symmetries* are found by converting an image into Frieze Expansion Pattern (FEP), shifting it, and comparing with the original FEP. If they are the same, then rotational symmetry exists.

As we can see, checking for all symmetries requires shifting points in edges or in the FEP. To do that, an actuation model (specific to an agent's actuation system) translates movement of the actuation system into movement of a "camera" (perception system). The translational symmetries are found by using the actuation system to move the camera in the direction of edges orientations. The edges in the new shifted image are recomputed and compared to the edges in the original image. If they match, they have translational symmetry (linear segments of the image). The linear edges are represented by the following wreath product: $\{e\} \wr \mathfrak{R}$, which means a point {e} acted on ($\wr$) by a translation ($\mathfrak{R}$). Corners, or end points of line segments, are found by shifting the image, comparing the edges of the new and original images and noting the closest changed points to the line segments.

Rotational symmetries are found by first transforming an image to a Frieze Expansion Pattern (FEP) [6], where the center of expansion is in the shape's center. If the FEP is created for all angles from 1 to 360, then shifting every point column-wise in the resulting FEP 360 times is equivalent to rotating the image around its center 360 degrees. Every shift is equivalent to a 1 degree rotation. If the shifted FEP matches the original for some angle, the image has the rotational symmetry for that angle.

BACKGROUND

Leyton [7] proposed the wreath product as the basis of cognition. A wreath product is a group formed by the semi-direct product of two subgroups, the fiber group F and the control group C whose action is to map copies of F to each other (also see [3]). For example, the wreath product representation of an outline of a square can be the following: $e \wr Z_2 \wr \Re \wr Z_4$. The first element e is a fiber group consisting of a specific point in a given coordinate frame and the identity element acting on the point. Z_2 is a control group for a characteristic function, which selects points in the shape, and rejects points not in the shape. R is a control group which translates a point along a specific line in space. The last group Z_4 rotates a specific line segment 0, 90, 180, 270 degrees about the center of the square.

Neural networks have been shown capable of detecting image features (e.g., corners, gradients, edges, blobs, etc.), and more recently, some work has been done on implementing graphics-like geometric transforms on image data. In particular, Hinton et al. [5], have proposed transforming auto-encoders to produce a vector of instantiation parameters to deal with variations in position, orientation and scale in images; also see their work on factored higher-order Boltzmann machines [8]. An alternative approach is given by Berkes [2] for transformation learning, but this method can only learn parameters that are linear functions of the input; however, they argue that their "model corresponded closely to functional and anatomical properties of simple and complex cells in the primary visual cortex." None of this previous work formulates a unified group theoretic sensorimotor analysis as proposed here. Beall and Henderson presented preliminary results on this topic (see [1]).

METHODS

Matlab was used for coding neural networks. Also, two virtual sensorimotor systems were used to simulate the movement of a camera and for reproducing shapes by an agent. The first system is an "eye," which is capable of pan and tilt. The second system is a two-link "arm," which is fixed at one end.

A recurrent neural network (RNN) is a set of N nodes (neurons) connected to each other by edges with weights w_{ij} (weight of an edge from the node j to the node i). Each neuron has an activation function f_i, which is applied to each node as follows:

$$u_i(t+1) = f_i(\sum_{j=1}^{N} w_{ij} u_j(t)), i = 1 \dots N$$

f_i is one of a predetermined small set of functions, such as identity, absolute value, greater than zero, etc. One step in running the RNN calculates a new value for each neuron. The RNN can be run for a fixed number of steps (starting from some initial configuration), or until the neuron values stabilize.

CERNNs were constructed for each of the sub-tasks in Figure 1. The discovery of translation symmetry is achieved by moving the virtual *eye* and finding invariant pixels (i.e., pixels with the same orientation at the same place relative to the position of the eye). There are different ways of finding rotational symmetry. By rotating an image (or a camera) a small amount, the similarity of the edges to the original image can be checked. The Frieze Expansion Pattern (FEP) [6] is used to find rotational symmetry which can be detected as a translational symmetry along the horizontal axis.

A translational symmetry in a certain direction means that, if we move a set of points in that direction, it will be isomorphic to the original set. If we have a finite set of points, such as a line segment, then we can add a characteristic function to each point. The function's value will be 1 if the given point is in the set of points and 0 otherwise. Then, if we move a camera, for example, in a positive direction along the line, the right most point of the line segment will move out of the line segment and no longer be selected. Using information about edge directions, an actuation model which can simulate a small displacement, and a function for calculation of similarity between two images, we can find selected points and combine them into line segments. The following algorithm *$\mathcal{R}$-Symm* detects translation symmetry at each pixel.

Algorithm R-Symm
Data: image, orientation, histogram, actuation model
Result: line segments

for each orientation above threshold
    move image small amount in orientation direction and record invariant pixels;
    move edge image small amount in negative orientation direction and record invariant pixels;
every invariant pixel with similar neighbors is added to linear segments

All of the functions used in the computation, such as the calculation of dx and dy at each point, absolute value, logical *and* and shifts in four directions are implemented as hard-wired CERNNs. As an example, the RNN for dx shifts an image in the x direction and subtracts the original image from the shifted image. Figure 2 shows a CERNN composite network for computing edge pixels.

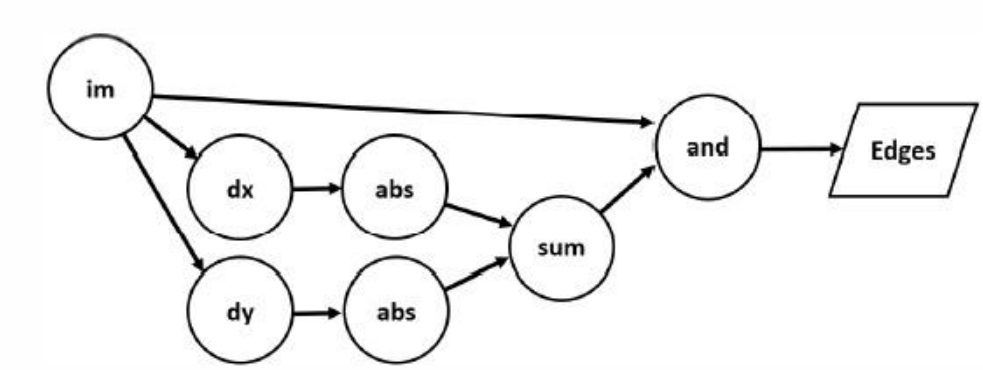


Figure 2: CERNN composite network for finding edges of an image. Since the images we consider here are created using 1s and 0s for simplicity, the magnitude of a pixel can be calculated as a sum of absolute values of dx and dy. By and'ing the magnitudes with the original image, the edges are recovered from the whole image.

Translational symmetries are computed using shifts in major directions. In particular, we considered shifts in four directions as shown in Figure 3.

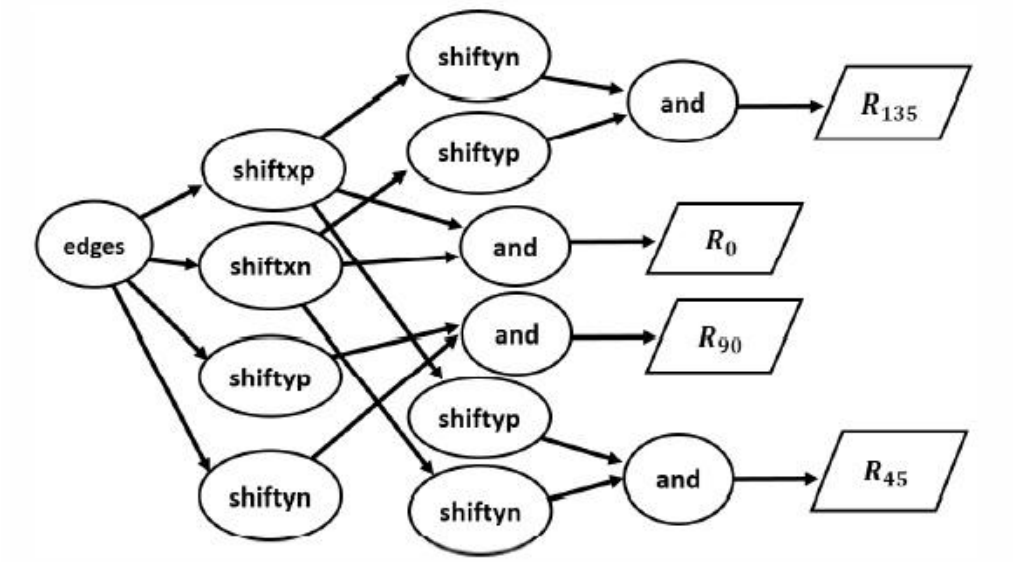


Figure 3: CERNN composite network for finding translational symmetry in four major directions: 0, 45, 90 and 135 degrees.

Figure 4 shows detection of corners (end points of line segments) for horizontal edges. End points of vertical edges are found similarly.

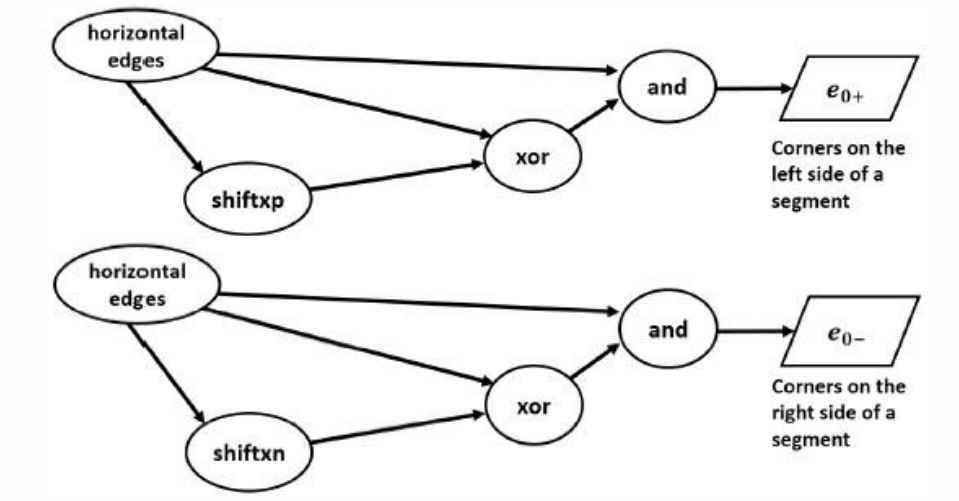


Figure 4: Finding point symmetries for horizontal edges. First, an edge is shifted. Then, using "xor" and "and" combination, the pixels which were "on" in the original image, but became "off" in the shifted image are found.

Rotational (O(2)) symmetry was detected assuming that the center of the shape is the location of the rotation axis. The CERNN composite network structure for finding this symmetry is shown in Figure 5.

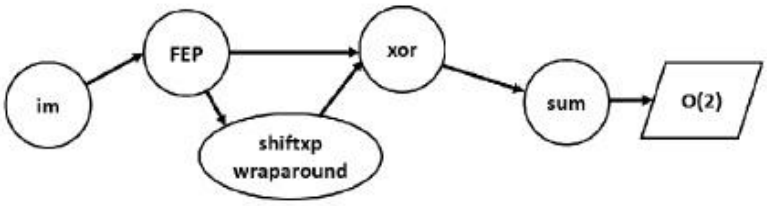


Figure 5: CERNN network for finding rotational symmetry. Image is transformed into FEP, which is shifted with wraparound. Then the difference between original and shifted FEPs is found by xor function. If they are the same, then the output of the xor function will be all zeros and the sum of all zeros will be zero. So, the output of the network is 0 if there is a rotational symmetry. If this network is run more than once, we can find rotational symmetry for different rotation angles.

The FEP is achieved by a simple mapping of Cartesian layout pixels to a polar form. The example of mapping the middle point of an image into the middle row of its FEP is shown in Figure 6 on the left; the full transform is shown on the right.

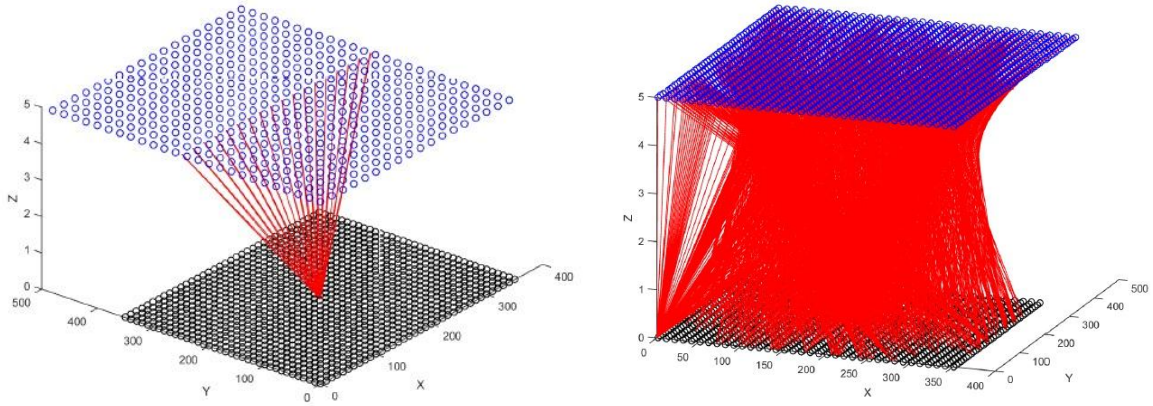


Figure 6: Middle row of FEP comes from the middle point of an image (left).
Wiring Diagram of Neural Network Mapping from Regular Image to FEP (right).

Figures 7 shows the transformation of three shapes into FEPs. For example, a circle is transformed into a rectangle.

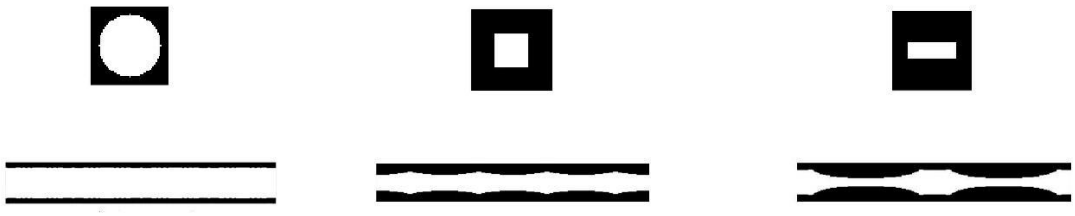


Figure 7: Image and FEP of Circle (left); Image and FEP of Square (center); Image and FEP of Rectangle (Right).

After the FEP of a shape is found, the rotational symmetry can be found by translating the FEP image horizontally (with wraparound). The following algorithm Zk-Symm finds rotational symmetries of a shape.

Algorithm Z k-symm
Data: FEP (one for each CoM)
Result: Θ_ vector (symmetry angles)
for all shifts t
    if similar to untranslated FEP
        add t to Θ_ vector

A recurrent neural network is used to shift the FEP along the x axis. The result is compared to the original FEP and if they are similar, the angle corresponding to one FEP shift is output as a symmetry angle.

RESULTS

The analysis for translational and rotational symmetry was run on the following shapes: square, rectangle, and circle. In addition, the shapes were distorted by deleting and adding boundary points. The same analysis was run for noisy images. The edges and symmetries were successfully identified. Figure 8 shows lines with x axis direction (0 degrees) translational symmetry in a square (on the left), and y axis direction (90 degrees) symmetry in the square (on the right). Figures 9-14 show FEPs and rotational symmetry measures for a square, a rectangle, a circle and their noisy versions. We can see that the method is robust to the noise.

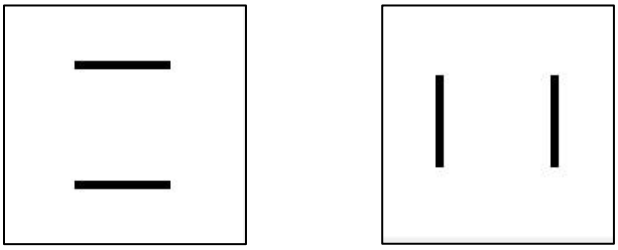


Figure 8: Identified points of translational symmetry in a square. 0 degree symmetry is on the left, 90 degree symmetry is on the right. Two horizontal and two vertical sides of the square were found.

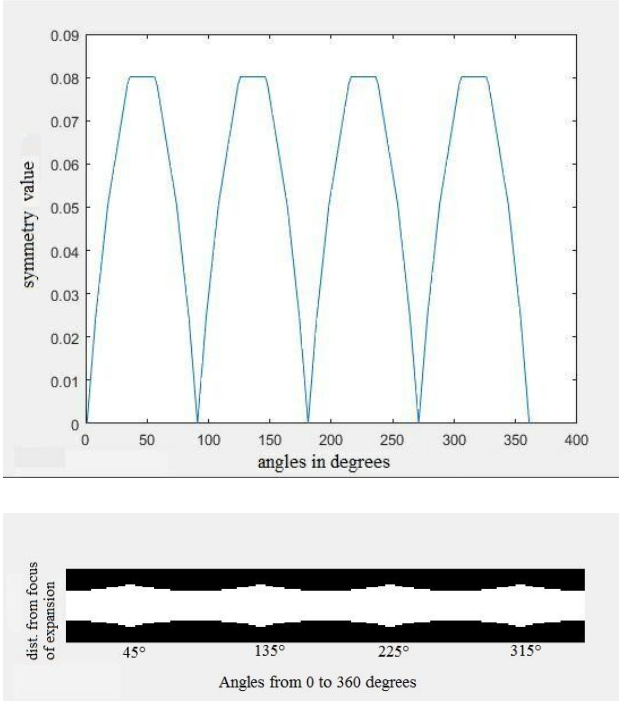


Figure 9: Square Rotational Symmetry Analysis
The network successfully identified the four rotational symmetries in a square. Symmetry value (top picture) equals zero at angles where the rotational symmetry is present. For the square, the angles are 0 (or 360), 90, 180 and 270 degrees. The bottom picture shows the FEP of a square with the biggest distance from focus of expansion at 45, 135, 225 and 315 degrees.

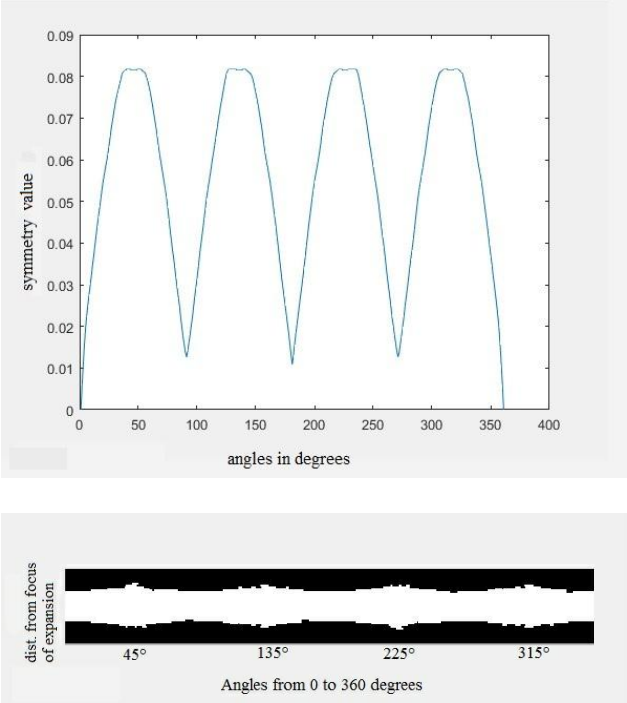
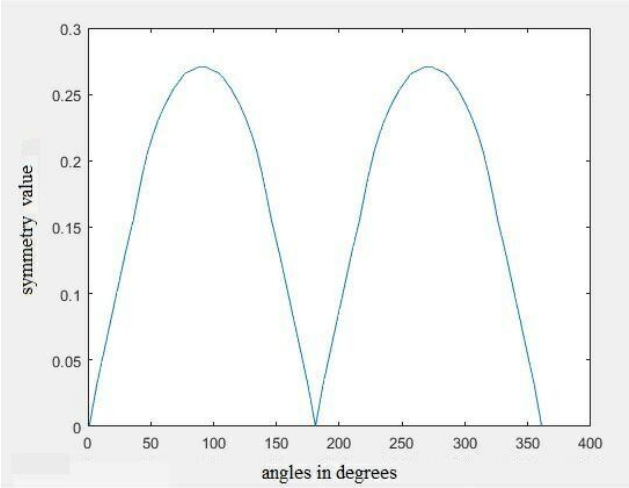


Figure 10: Noisy Square Analysis
Even with the noise the network identified rotational symmetry at the four angles. Note that the symmetry value at symmetry angles is not as perfect as with no noise.



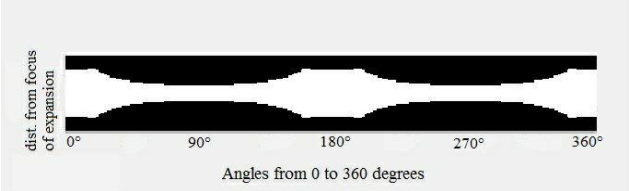
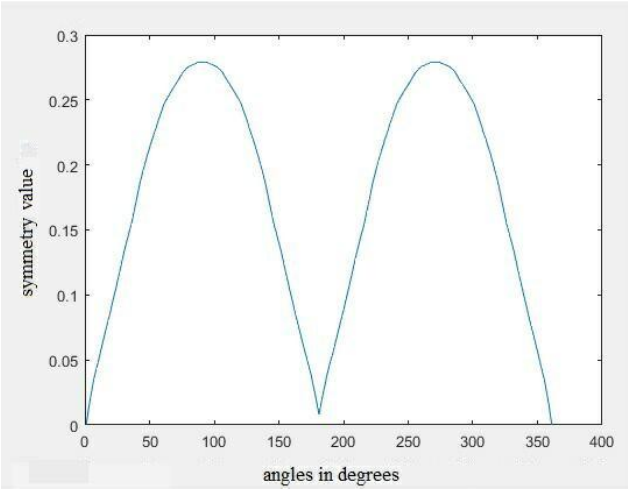


Figure 11: Rectangle Analysis
Rectangle has 2 angles of rotational symmetry: 0 (or 360) and 180 degrees.



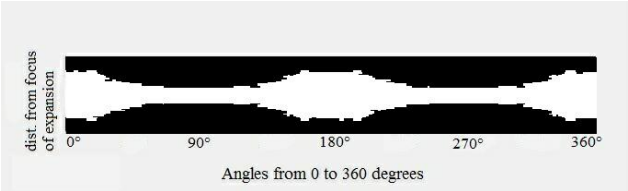


Figure 12: Noisy Rectangle Analysis
The network is robust and can identify the same two symmetry angles in a rectangle. Bottom picture shows the level of noise in the rectangle.

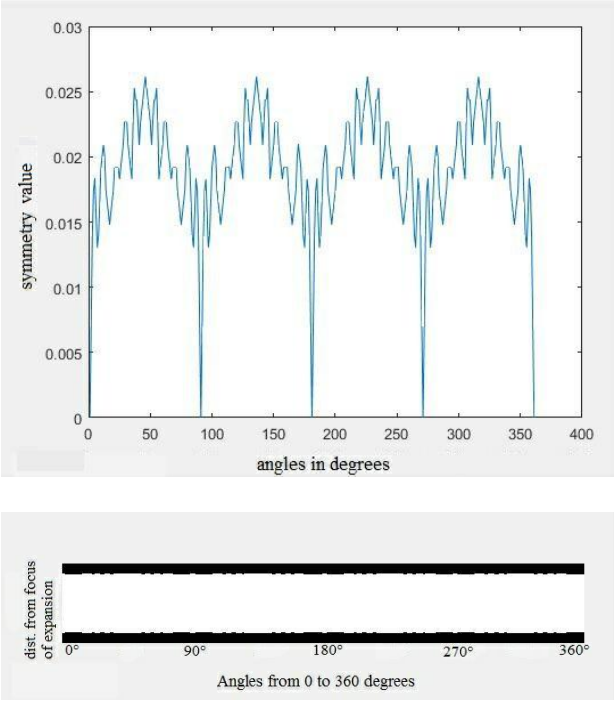


Figure 13: Circle Analysis
Theoretically, symmetry measure for a circle should be zero for all angles. Because of an approximation in drawing a circle using pixels, the symmetry measure is close to zero.

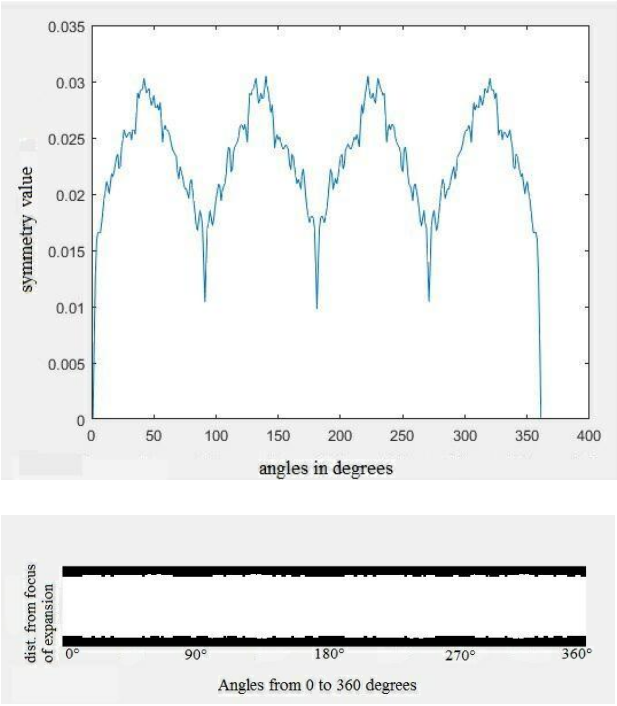


Figure 14: Noisy Circle Analysis
The network still recognizes a circle symmetry with slightly higher error than for a true circle.

To get the wreath product, we catenate the results for the translation and rotational symmetries. When a shape is represented as a wreath product, the representation can be passed to another motor system, like a two-link robot arm to generate the shape. As an example, a neural network was learned for the transformation from the pan-tilt actuation system to the arm (see Figure 15). For more on knowledge transfer between motor systems for shape generation, see Henderson et al. [4].

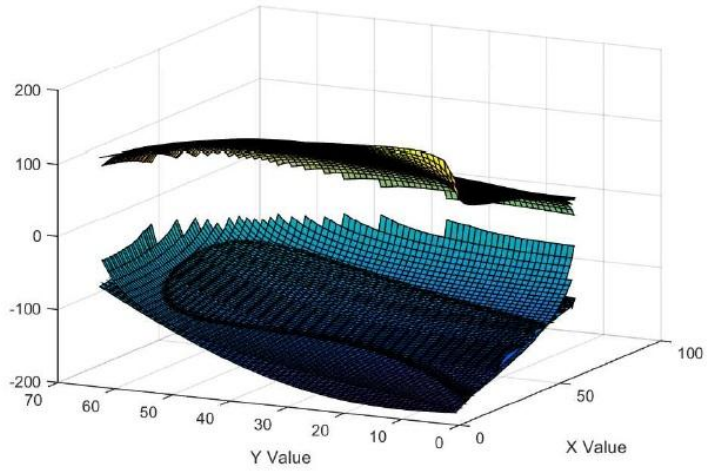


Figure 15: Neural Network Output for Map from (x, y) to (Θ1, Θ2).

CONCLUSIONS AND FUTURE WORK

The framework (DISCERNN) was created for combining recurrent neural networks (CERNNs) into composite networks from subnets created by hand, by learning the weights, or by conversion from procedural function descriptions. Our set of CERNNs takes an image as an input and produce a wreath product representation of the 2D shape. Moreover, this model proved to be robust to noise in images. In the future, in-depth experiments can be conducted to further test the robustness of the method. For instance, more shapes can be included into tests, as well as more noise can be added to the shapes. The method can be adapted to larger scale applications such as text analysis or 3D shape analysis. In addition, we can explore the use of CERNNs in combination with Deep Belief Networks.

References

[1] Tatyana Beall and Thomas C. Henderson. A Sensorimotor Approach to Concept Formation using Neural Networks. *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, Baden-Baden, Germany, September 2016.

[2] P. Berkes, R.E. Turner, and M. Sahani. A Structured Model of Video Reproduces Primary Visual Cortical Organisation. *PLoS Computational Biology*, 5(9):1–16, 2009.

[3] D.S. Dummit and R.M. Foote. *Abstract Algebra*. Wiley & Sons, Hoboken, NJ, 2004.

[4] Thomas C. Henderson, Narong Boonsirisumpun and Anshul Joshi . Actuation-based Shape Representation applied to Engineering Document Analysis. *Proceedings International Conference on Agents and Artificial Intelligence*, Rome, Italy, Feb 24-26, 2016.

[5] G.E. Hinton, A. Krizhevsky, and S.D. Wang. Transforming Auto-Encoders. In *International Conference on Artificial Neural Networks and Machine Learning*, Espoo, Finland, June 2011.

[6] S. Lee, R. Collins, and Y. Liu. Rotation Symmetry Group Detection via Frequency Analysis of Frieze-Expansions. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[7] M. Leyton. *A Generative Theory of Shape*. Springer, Berlin, 2001.

[8] R. Memisevic and G. Hinton. Learning to Represent spatial Transformations with Factored Higher-Order Boltzmann Machines. *Neural Computation*, 22:1473–1492, 2010.