NEURAL CLUSTERING OF CORRESPONDENCES FOR VISUAL POSE ESTIMATION

Tomás H. Maul
School of Computer Science
The University of Nottingham Malaysia Campus
Jalan Broga, 43500 Semenyih,
Malaysia
E-mail: Tomas.Maul@nottingham.edu.my

Sapiyan Baba
Faculty of Computer Science & IT
University Malaya
50603, Kuala Lumpur
Malaysia
E-mail: pian@um.edu.my

KEYWORDS
Unsupervised Learning, Clustering, Higher-Order Neural Networks, Correspondences, Pose Estimation.

ABSTRACT
This paper is concerned with the problem of visual pose estimation, which entails, for example, the estimation of object translations. It adopts a correspondence based approach in general, and in particular, looks into a neural network implementation of the approach. The objective of the paper is to demonstrate how the approach can be learnt via the unsupervised clustering of correspondences into clusters representing different poses. Purely local (i.e. Hebbian) mechanisms were adopted in order to ensure not only the practical value of the learning algorithm but also its biological relevance. The results of the experiments here reported show that the learning strategy adopted allows for the successful unsupervised clustering of correspondences, even when the environment puts forth several difficult challenges, such as scarce or correlated features.

INTRODUCTION
Vision presents us with a vast array of computational problems, e.g.: object recognition, motion analysis, stereopsis, colour constancy and pose estimation. The problem of estimating a pose involves, for example, determining the position, orientation and size of a particular object. The main objective of the paper is to demonstrate the feasibility of an unsupervised clustering approach to the problem, which furthermore incorporates several neurobiological constraints, such that the approach can be relevant to both real-world applications and the simulation of biological processes.

A large proportion of pose estimation algorithms within theoretical and applied Computer Science are based on correspondences, which are nothing other than vectors defined by two points that represent two matching local features: one in the source pattern and the other in the target pattern. Correspondence based approaches can be roughly divided into two categories: 1) search based and 2) vote based. Search based correspondence approaches (Fischler and Bolles 1981; Ullman 1996; Breuel 1992; Baird 1985) score highly in terms of generality and robustness, but falter somewhat in terms of parallelizability, computational simplicity and efficiency. Vote based correspondence approaches, which are based on the Hough Transform (Hough 1962; Ballard 1981; Tian and Shah 1997; Espinosa and Perkowski 1991; Lamdan and Wolfson 1988), are the only approaches reviewed so far that render themselves quite naturally to neural implementations. Because we are interested in a biologically relevant (or neural) unsupervised clustering approach to pose estimation, we have chosen to pursue vote based correspondence approaches.

Seeing that correspondences can be represented by multiplicative nodes and that the transformation space can be represented by an array of summation nodes, the whole approach can be elegantly encapsulated by a special case of a Higher Order Neural Network (HONN), which is closely related to sigma-pi networks. The following specializations distinguish the resulting architectures from general HONNs: 1) two input maps rather than one are used, 2) only second order information is represented and 3) second order information is represented between, rather than within maps (i.e. inter-map rather than intra-map conjunctions). Refer to Figure 1 for a diagram of an artificial neural architecture solving a highly simplified one-dimensional pose estimation problem (i.e. shifts of -1, 0 or +1 pixels).

![Figure 1: Pose estimating artificial neural network.](image-url)
The current paper aims to demonstrate how simple local mechanisms, governed by Hebb-like rules (Hebb 1949), in association with coherent visual stimulation, are sufficient for implementing unsupervised learning of the pose estimating connectivity alluded to in Figure 1 (i.e. the connectivity between correspondence detectors and vote nodes). Note that correspondence detectors, are essentially conjunctions (or multiplicative nodes) between maps, representing whether a particular correspondence between features is being manifested or not, e.g.: if feature $F1$ at location $L1$ and feature $F1$ at location $L2$ are both active at time $t1$, then the correspondence between $L1$ and $L2$ should be detected at that time.

From the perspective of Granular Computing, the transformation from the input layer to the vote node layer, can be seen to be imposing a specific granularity, resulting from the unsupervised clustering of correspondence detectors at different vote nodes (i.e. poses), and realizing two different resolutions of pose estimation differentiated in terms of accuracy (i.e. correspondence detectors represent poses with limited accuracy whereas vote nodes represent poses with significant levels of accuracy). So far, to the best of our knowledge, the problem of pose estimation has never been solved via the unsupervised clustering of correspondences.

**METHODS**

Our simulated environment, responsible for providing visual information to the network, consists of two elements: 1) a static image and 2) a dynamic window. The dynamic window is allowed to move around the image, continuously conveying the underlying information to the developing neural network. More specifically, at each iteration, the window contents of both the current and the previous time steps are conveyed to the network. The transformation (or pose) relating the contents of the window at both time steps recovers the motion undergone by the window. Thus motion analysis in this context can be addressed as a pose estimation problem. By controlling the motion of the window, we control the type of pose estimation problem that the developing neural architecture is confronted with. By controlling the nature of the underlying static image we can control environmental factors that may or may not influence the clustering process.

Our learning algorithm consists of the following four main elements: 1) a stability measure, 2) Hebb-like mechanisms acting on the stabilities of pairs of synapses (each element in a correspondence detector pair is from this point onwards referred to as a synapse), 3) a neural trace and 4) random rewiring. These elements were chosen partly due to their practical significance and partly because of their biological relevance. In the following paragraphs we will describe each one of the model’s main elements. Note that in a biological context the expression learning algorithm can, or should be, substituted by the expression developmental process.

The stability measure determines how likely a synaptic-pair (representing a particular correspondence detector) will be eliminated. The higher the stability value of a synaptic-pair, the less likely it is to be eliminated.

The stability measures of synaptic-pairs indirectly control how the patterns of connections between neurons are gradually molded. The question is then, what controls these stability measures? Our learning algorithm, which simulates a simplified developmental process, employs local Hebb-like mechanisms, which can be summarised by the following two rules: 1) when a synaptic-pair is active (i.e. the correspondence it represents is detected in the environment) and the postsynaptic neuron (i.e. a vote node to which it is connected) is also active, then the stability of that pair increases, 2) when the activity of a synaptic-pair and its postsynaptic neuron is decorrelated, the stability of the pair decreases. The mechanisms employed here exhibit two main differences relative to a conventional Hebbian synapse: 1) two synapses rather than one are involved in the process and 2) the changes caused by the learning rule do not affect the efficacy (see Hebb 1949) of the presynaptic neuron causing the postsynaptic neuron to fire (as in the notion of connection weights) but rather, affect the probability of the synaptic-pair being eliminated.

As already mentioned, in a Hebbian synapse correlated activity between pre and postsynaptic neurons increases the relevant connection strengths (see Haykin 1999 for a good introduction). A neural trace is essentially the prolonging of this effect throughout time. A modified Hebbian rule for example (Földiásk 1991) might state that changes in synaptic strength are proportional to pre-synaptic activity and a temporal average of post-synaptic activity. As demonstrated in (Földiásk 1991), neural traces are very useful for capturing invariant properties of the environment. Interestingly, and maybe unsurprisingly, the same mechanism that throws away invariant information for the implementation of invariant feature-detectors, is useful for the learning (or simulated development) of architectures that can detect/estimate those very same transformations. The idea is based on the notion that transformations are somewhat stable through time. If transformation $X$ occurs between time $t1$ and time $t4$, and node $n1$ is activated at $t1$, then propagating some of $n1$’s activity to the following time-step (similarly to a conventional neural trace), makes it more likely that $n1$ will be activated again for the same transformation $X$, thus accelerating the learning of “$n1$ represents transformation $X$".
When synaptic-pairs are eliminated, new ones must be formed. Ideally, one would like the correct synaptic-pair to be generated, but again, we have to rely on “blind” local mechanisms. Thus random pairs of synapses are chosen for the substitution of eliminated correspondence detectors. The only exception to randomness employed assumes that the constituent synapses of any synaptic pair originate from different maps (i.e. one synapse from the source pattern and the other from the target pattern).

Algorithm 1 shows how the activities and stabilities of synaptic-pairs are updated. The algorithm begins by computing the activations of vote neurons as specified in Algorithm 2. Based on the neural activations computed via line 11 in Algorithm 2, Algorithm 1 then selects the node (i.e. win) with the largest activation, stores the latter’s activity in \( w_A \) and resets the non-winners to zero. Clearly, this is a simplification that warrants further elaboration in future experiments (e.g. iterative isolation of maximum values through lateral inhibition). The model also simplified the neural trace, which essentially consists of an upper bounded propagation of the winner’s activity to the following iteration (see line 4 of Algorithm 2). Following this, Algorithm 1 proceeds to implement four Hebb-like rules (i.e. R1, R2, R3 and R4).

### Algorithm 1: Update Network State

```plaintext
1: procedure UPDATENETSTATE(set, IV_old, IV_new)  
2:    net ← compActivations(set, IV_old, IV_new)  
3:    Votes ← getVotes(net)  
4:    nS ← getNumSynapses(Votes)  
5:    [win, wA] ← getWinner(Votes)  
6:    net ← resetLosers(set win)  
7:    net ← makeTrace(set win, maxTrace)  
8:     for all \( v \) ∈ Votes do  
9:         for all \( s \) ∈ getSynapses(v) do  
10:            if \( v \neq \text{win} \) & \( s.\text{activ} = 0 \) then  
11:               \( s.\text{stab} = \text{win} \cdot 0.5 \)  
12:               end if  
13:            else if \( v \neq \text{win} \) & \( s.\text{activ} \neq 0 \) then  
14:               \( s.\text{stab} = s.\text{stab} - \alpha/s.\text{stab} \)  
15:               end if  
16:            else if \( v = \text{win} \) & \( s.\text{activ} \neq 1 \) then  
17:               \( s.\text{stab} = s.\text{stab} + 1 \)  
18:               end if  
19:            else if \( v = \text{win} \) & \( s.\text{activ} = 1 \) then  
20:               \( s.\text{stab} = s.\text{stab} - 1 \)  
21:               end if  
22:            else if \( v = \text{win} \) & \( s.\text{activ} = 0 \) then  
23:               \( s.\text{stab} = s.\text{stab} + 1 \)  
24:               end if  
25:            else  
26:               update ← (\( s.\text{activ} = 1 \)) \( s.\text{stab} + \beta(v_A/nS) \)  
27:               end if  
28:            end if  
29:         end for  
30:     end for  
31: end procedure
```

Within the rules section of Algorithm 1, \( s.\text{activ} \) refers to the activity level of synaptic-pair \( s \) (one represents a detected correspondence while zero represents an undetected correspondence), \( s.\text{stab} \) refers to the stability of synaptic-pair \( s \), which is limited by upper (i.e. maxStab) and lower (i.e. minStab) bounds. The four rules essentially represent all possible combinations between whether vote neurons are winners or not, and whether synaptic-pairs are active or not.

The environment simulation has one more function, apart from updating the attentional window and the network state, which should be mentioned, i.e.: rewire. This function essentially scans all synaptic-pairs in all vote nodes, and substitutes those synaptic-pairs whose measures of stability are smaller than some lowerbound, with new randomly chosen ones.

### RESULTS

All of the experiments reported here were conducted in the context of constrained estimation, where the transformations of the attention window were restricted to horizontal and vertical shifts (i.e. two unknowns). Recall that correspondences are capable of constraining only two unknowns.

Refer to Figure 2 in order to get a better idea of how overall synaptic-pair stability changes with time. The y-axis (i.e. “percent stable”) refers to the percentage of synaptic-pairs whose stabilities are larger than half of the maximum stability measure. The x-axis measures time in terms of simulation iterations. In order to remove the effects of extraneous factors, the graph averages the results of 5 simulations. In order to get a better understanding of the shape of this graph (e.g. why does learning slow down dramatically at later stages?), one must consider several probabilistic factors governing learning.
Since transformations change randomly, and the formation of synaptic pairs is also random, one can ask the following question: given a certain number of synaptic-pairs (regardless of the number of vote nodes) and given a particular range of transformations, what is the probability that at least one of the random synaptic-pairs matches the random transformation? Assuming constrained problems, in the case where there is no feature scarcity, the probability is defined by:

\[ P(\text{match}) = 1 - \left( \frac{t-1}{t} \right)^s \]  

(1)

where \( t \) represents the range of transformations that can occur and \( s \) represents the number of synaptic-pairs available.

Figure 3 depicts various curves representing different ranges of transformations for numbers of synaptic-pairs varying between 1 and 150.

1. **Feature scarcity.** Environments which exhibit a small proportion of features, lead to low probabilities of correspondence detection, and thus to slow or even impaired learning.

2. **Feature repetition (or correlation).** The repetition of features leads to false correspondences (i.e. correspondences between matching features which nevertheless represent different object parts), which in their very nature are misleading and can therefore lead to erroneous connectivity.

3. **Guidance.** How does each synapse from one map “know” which synapse it needs to pair up with from the other map? There is no global intelligence guiding the whole process, only local mechanisms.

4. **Availability of correspondence detectors.** The total number of possible inter-map conjunctions (or correspondences) is likely to be larger than what each neuron can represent.

Feature scarcity and repetition pertain to environmental issues whereas guidance and detector availability pertain to the neurobiological constraints mentioned earlier.

A synthetic image generator was devised in order to control feature scarcity. Figure 4 illustrates several learning curves under two conditions of feature scarcity, i.e.: low (on the left) and high (on the right). The graphs average the results of 20 simulations. Each graph consists of three curves: stability, T-Coverage (from transformation coverage) and C-Coverage (from correspondence coverage). The stability curve represents the percentage of synaptic-pairs with a stability value larger than half the maximum. The T-Coverage curve represents the percentage of environmental transformations represented by the network. If a particular transformation \( t \) is represented by a certain set of \( x \) correspondences and the synaptic-pairs of a particular vote node representing \( t \) detect \( x/2 \) of that set, then the C-Coverage for that vote node is 50%. The C-Coverage curve represents the average C-Coverages for all of the network’s vote nodes.

As predicted, when the environment manifested abundant features (left-hand side of Figure 4), learning proceeded seamlessly, with all three curves approximating 100%. When, on the other hand, feature scarcity was high (right-hand side of Figure 4), learning was somewhat hindered (i.e. both stability and C-
Coverage curves stagnated at approximately 80%. Having said this, learning was better than expected, seeing that in spite of the scarcity of features, T-Coverage was capable of reaching 95% after about 5000 iterations. This is an indication that the learning algorithm is somewhat robust to feature scarcity.

Repeating (or correlated) features pose another problem to learning seeing that they lead to false correspondences, which by their very nature are misguiding. A synthetic image generator allowed us to control the number of repeating features in the environment. Figure 5 depicts the results from two conditions, defined by the image’s average probability of a true correspondence (i.e. a correspondence between matching features which represent the same object part), which is a direct result of feature repeatability. The curves (stability, T-Coverage and C-Coverage) are defined in the same way as those in Figure 4 and represent averages from 20 simulations.

Both conditions allowed T-Coverage to approximate 100%. However, as the probability of a true correspondence (PTC) decreased from 0.7 to 0.53 the main victims were C-Coverage and learning speed. Regarding the latter, note that when PTC was 0.7, T-Coverage approximated 100% at about iteration 5,000, whereas when PTC was 0.53, T-Coverage approximated 100% at about iteration 18,000.

The PTC of a window can be calculated using the following expression:

\[
P(TC) = \frac{n + \sum_{i=1}^{r} x_i}{n + \sum_{i=1}^{r} (x_i)^2}
\]

where \(n\) represents the number of non-repeating features, \(r\) represents the number of repeating feature types and \(x_i\) represents the number of elements exhibited by repeating feature type \(i\). Thus, the numerator refers to the number of true correspondences provided by a particular window, while the denominator refers to the total number of correspondences (i.e. true and false correspondences).

Figure 7 confirms our optimism and demonstrates that learning proceeds quite effectively in the context of natural images. The simulation was programmed to stop at the overall stability level of 95%, at which point both the T-Coverage and C-Coverage curves both approximated 100%. As before, the curves were averaged from 20 simulation runs.
CONCLUSION

This paper has proposed neural architectures implementing the functionality of unsupervised clustering of correspondence detectors as a viable approach to the problem of pose estimation. Moreover, it has focused on purely local learning mechanisms which makes the approach relevant not only to applied domains but also to the simulation of biological systems. The experiments carried out thus far constitute useful proofs of concept and thus encourage several lines of future work.

The utility of correspondences, and thus the benefits of their unsupervised clustering, are not restricted to visual pose estimation (or even the visual modality). There are many other visual problems that can be solved via correspondences and thus, which warrant the same kind of study presented here, e.g.:

1. Motion analysis. The problem of motion analysis (or object tracking), as this paper has shown, can be formulated in terms of pose estimation between time frames.
2. Stereopsis. The computation of disparity for stereopsis relies on finding corresponding features from both retinas, which in turn can be implemented by an architecture employing inter-map conjunctions.
3. Shape representation. In (Osada et al. 2002), it was shown how a shape can be represented by the statistics of the distributions of correspondences occurring between the shape and itself.
4. Edge orientation estimation. Correspondences within an edge pattern can vote for particular orientations, thus calling for an architecture involving inter-map conjunctions and vote summation.

In another line of research it should be useful to relax the neurobiological constraints applied in this study and thus for example allow each vote-node to represent all possible correspondences. In this scenario, correspondences would no longer require a stability measure, and instead would use a firing efficacy measure (much like a traditional weight), which can be interpreted as a fuzzy membership function. It would be interesting to see how different network architectures might be capable of learning different kinds of efficient representations (e.g. can six vote nodes efficiently represent a space consisting of 100 different transformations, through the unsupervised learning of a broad tuning coding strategy?).

The results reported here demonstrate the application of correspondences with regards to a constrained problem (i.e. horizontal and vertical shifts). Although work has already begun in this direction, it would be interesting to further pursue the question of how the approach can be optimized with reference to unconstrained problems, i.e.: problems that involve more than two unknowns, such as similarity transformations (translation, rotation and scaling).

To conclude, we would also like to look into the question of how to prove (or disprove) the implementation of correspondence-based approaches in biological neural systems. Relatively recent work has made it clear that the computational machinery required to implement the approach efficiently does indeed exist in biological systems. Correspondence detection, for instance, can be computed by logical conjunctions implemented by non-linear computations taking place at local regions of dendritic trees (Polsky et al. 2004), whereas vote nodes can be implemented by traditional signal integration processes. Recent neurohistological methods (Gan et al. 2000; Grutzendler et al. 2003) might provide the means to prove (or disprove) our hypothesis. In these methods, multiply colored dyes are applied to neural tissue via particle-mediated ballistic delivery, fully staining neurons in a Golgi-like manner. If it is possible to identify and target two neuronal maps and apply differently colored labels to them, then it should be possible to deduce some of the computations taking place between them by observing the colors and placements of synapses at their target locations (see Figure 8 for a simplified illustration).

REFERENCES


AUTHOR BIOGRAPHIES

TOMAS H. MAUL was born in Madeira, Portugal and did a BSc. in Biological Psychology at the University of St. Andrews, an MSc. in Computer Science at Imperial College and a PhD. in Computational Neuroscience at the University of Malaya. He worked for two years at MIMOS Bhd. as a Senior Researcher in the fields of Pattern Recognition and Computer Vision. He is currently an Assistant Professor at the University of Nottingham Malaysia Campus, where he conducts research in the areas of Neural Computation, Optimization and Computer Vision. His e-mail address is Tomas.Maul@nottingham.edu.my and his Web-page is http://baggins.nottingham.edu.my/~kcztm/

SAPIYAN BABA was born in Malaysia and did a BSc. at Essex University, an MSc. at Dundee University and a PhD. at Keele University. Professor Sapiyan Baba is currently serving as dean of the Faculty of Computer Science and Information Technology at the University of Malaya. His areas of expertise include Artificial Intelligence, Neural Networks, Cognitive Science, Cognitive Robotics, Intelligent Tutoring Systems and Bioinformatics. His e-mail address is pian@um.edu.my.