7-23-2019

Approximate Pattern Matching using Hierarchical Graph Construction and Sparse Distributed Representation

Aakanksha Mathuria
Portland State University, aakanksha.mathuria@gmail.com

Dan Hammerstrom
Portland State University

Citation Details

"Approximate Pattern Matching using Hierarchical Graph Construction and Sparse Distributed Representation," A. Mathuria, D.W. Hammerstrom, International Conference on Neuromorphic Systems, Knoxville, TN, July 23-5, 2019

Approximate Pattern Matching using Hierarchical Graph Construction and Sparse Distributed Representation

Aakanksha Mathuria
Electrical and Computer Engineering
Portland State University
USA
aakan@pdx.edu

Dan W. Hammerstrom
Electrical and Computer Engineering
Portland State University
USA
dwh@pdx.edu

ABSTRACT

With recent developments in deep networks, there have been significant advances in visual object detection and recognition. However, some of these networks are still easily fooled/hacked and have shown "bag of features" failures. Some of this is due to the fact that even deep networks make only marginal use of the complex structure that exists in real-world images, even after training on huge numbers of images. Biology appears to take advantage of such a structure, but how?

In our research, we are studying approaches for robust pattern matching using still, 2D Blocks World images based on graphical representations of the various components of an image. Such higher order information represents the "structure" of the visual object. Here we discuss how the structural information of an image can be captured in a Sparse Distributed Representation (SDR) loosely based on cortical circuits.

We apply probabilistic graph isomorphism and subgraph isomorphism to our 2D Blocks World images and achieve *O (1)* and $O(n^k)$ complexity for an approximate match. The optimal match is an NP-Hard problem. The image labeled graph is created using OpenCV to find the object contours and objects' labels and a fixed radius nearest neighbor algorithm to build the edges between the objects. Pattern matching is done using the properties of SDRs. Our research shows the promise of applying graph-based neuromorphic techniques for pattern matching of images based on such structure.

KEYWORDS

Sparse Distributed Representation, Hierarchical Graph, Approximate Matching, Graph Isomorphism

1. Introduction

With the recent advances in deep networks, there has been significant progress in visual object detection and recognition. However, some of these networks have shown "bag of features" failures [48] similar to the other traditional object recognition techniques such as GIST [42], HOG (histogram of oriented gradients) [43], SIFT (Scale-invariant feature transform) [39, 40] and special envelope [41]. Deep networks make only marginal use of the complex structure that exists in real-world images, even after training on large numbers of images. None of these techniques actually captures the spatial relationships of the low level or high-level features, which biological networks appear to do.

Efficient graph representations capture the higher order information content of the objects and provide algorithmic benefits when recognizing complex images [33, 35]. Such higher order information represents the "structure" of the visual objects. Also, an important difference of the work described here is that we are using a non-standard representation of the graphical data based on sparse distributed representations.

Neuromorphic techniques such as Sparse distributed representations (SDR) of data, shapes, and graphs can play an important role in complex image processing. SDRs leverage the unique properties of the objects to provide algorithmic benefits. The use of sparse representations of data is motivated by a) the abundance of visual data b) the abundance of features in real life images and c) the ability of sparse representations to provide speed up via unique properties (e.g. union) of the representations. An SDR encodes any type of data into a binary vector which consists mostly 0's with a few 1's. SDR is very memory efficient, as only a few bits would have to be stored in the memory as the indices of the active bits. SDRs are the result of various research efforts into understanding the operation of cortical circuits [1, 2].

In the research described here, we are exploring new ways to represent images as hierarchical graphs to preserve the connectivity information among the objects and perform pattern matching using graph isomorphism. The graph of an image uses objects as the nodes. It contains the spatial information (connectedness, adjacency) of the objects in the image. The connections can be described as the Euclidean distance between the nodes. We formulate SDRs for all the nodes in the graph using their attribute information such as the number of edges, their sizes, connectivity and attributes of their neighbors. Then we use Euclidean distance criteria to represent the hierarchy in the graph, which can be used for efficient pattern matching.

An example of a hierarchical graph construction for an image containing multiple individuals can be used with three levels. For the first level, we can consider small body parts such as nose, mouth, eyes, etc. as nodes for a graph representing the face of a person. Each of these small body parts can be represented by SDRs

with their own attributes. Similarly, graphs of other large body parts such as hands, legs, etc. can be defined. With the properties of SDRs such as union, one can define SDRs for the entire graph, in this case of large body parts such as hands, legs, and face, etc. As a second level hierarchy, the graph can be constructed of these large body parts as nodes and connectivity between them and the graph representing an entire individual. Again, SDR of this entire graph can be obtained by performing union over the SDRs of the individual nodes. To construct the graph of the entire image with different individuals, the SDRs of each person can be considered as a node of the graph. This type of representation promises an efficient pattern-matching algorithm when implemented using graph isomorphism.

To demonstrate these ideas assume simple objects, e.g., rectangles and triangles, from a 2D blocks world. These are recognized using traditional algorithms (OpenCV). We then create graphs of these objects to allow the efficient recognition of more complex objects, built from the simple objects. Figure 1 and Figure 2 show how real-world objects can be broken into simple objects that can be easily and effectively represented using SDRs.

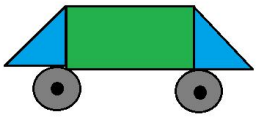

Figure 1: Simple Blocks-World image

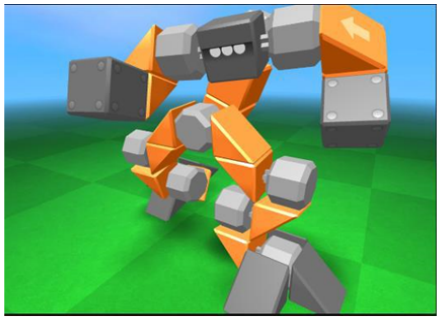

Figure 2: Complex Blocks-World image (source: www.pocketgamer.com)

In this work, we use probabilistic graph isomorphism and subgraph isomorphism to perform efficient approximate pattern matching in images. The optimal match is an NP-hard problem. However, with the help of SDR properties, we can perform graph matching in *O(1)* time and further choose *k* nodes subgraph out of big graph of *n* nodes in $O(n^k)$ and do the matching in *O(1)*. By combining the SDRs and graphs, we can perform pattern matching, which leverages structural information in an efficient manner.

1.1 Our Contributions

- We create a hierarchical graph representation to capture the structural information of an image.
- We implement the Sparse Distributed Representations for the hierarchies of a graph, which leverages algorithmic parallelism and makes computation faster and more power efficient.
- We demonstrate the approximate graph matching in *O(1)* and by choosing *k* nodes' subgraph out of *n* nodes' big graph in $O(n^k)$, subgraph matching in *O(1)* instead of solving in non-polynomial times with the help of SDR properties.

Our method allows us to capture structural information in images for doing pattern matching and uses very little data.

2. Related work

Object Detection is a very important part of any computer vision application. There are a number of applications from face detection and pedestrian detection to image and video retrieval. Object detection and recognition is an integral part of many common applications such as video surveillance, image captioning, video summarization, etc. Many techniques are used to detect the objects in an image. Some techniques use feature extractors such as SIFT [38, 39] and HOG [41]. Some use bounding boxes [12] and Contour detection. Deep learning techniques are starting to solve these problems but they are easily fooled and do not capture the structure of the image. 48 (2015) show how it is possible to produce images which are not recognizable to the human eye but DNNs classify as familiar objects.

Graphs are useful when one wants to represent the connectivity or structure of objects. Graphical approaches have been studied for many years, and yet there are still a number of unsolved problems. Applications such as document processing, scene processing, image retrieval [6, 11, 13, and 16] and video summarization [45] could benefit from such connectivity information. However, due to the complexity of working with graphs, traditional Computer Vision techniques often use a 'Bag of features' [43, 46] approach and so are missing information on object structure. For humans, features being in the wrong position degrade recognition accuracy. Imagine two images of bicycles, one being with the right position and orientations and other being with only the right components and wrong locations. When we classify this image using a 'bag of features' approach both images will be classified as bicycles, but the second image is not the correct form of a cycle. Being able to utilize such structure or connectivity information will be of significant value in image understanding. One approach to representing structure in deep networks is the development of Capsules [54] by Geoffrey Hinton and his group. Capsules take advantage of the fact that spatial relationships can be modelled by matrix multiplies.

We know that biology makes extensive use of connectivity and other kinds of structures when doing object recognition. In the work described here, we are applying sparse distributed representations to the problem of graph isomorphism which is required if graphical information is to become a part of the pattern recognition process.

Sparse distributed representation (SDR) is a technique that has been proposed as one technique that is used by cortical circuits to represent data. The best description of this technique can be found in an excellent paper by Jeff Hawkins and his team at Numenta [1, 36]. They have made extensive use of SDRs and are continually improving their techniques.

Hierarchical temporal memory [1, 34] is a hierarchical, unsupervised technique, which makes extensive use of SDRs for processing the input data from a variety of sources [33, 36]. SDRs are now being used in a number of commercial applications. One example is an application developed by cortical.io, [1, 2 and, 3], which performs natural language processing using the Numenta HTM algorithm.

3. Our Approach

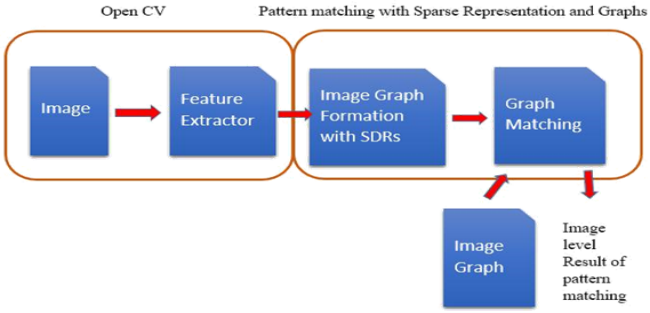

Figure 3: Data Flow Pipeline

Graphical representations of object structure have significant potential in helping to recognize complex objects in still and moving images. However, to use graphs effectively requires the ability to efficiently capture the graph structure from recognized features. This is a problem that has been studied and many techniques have been developed. As mentioned earlier, the OpenCV library has a number of state of the art feature extractors. The second problem, finding isomorphism in graphs and subgraphs, is significantly more computationally intensive. It is hypothesized that biology uses a number of computational techniques, but the most intriguing is Sparse Distributed Representation.

In this section, we describe the image processing pipeline. Figure 3 shows the data flow pattern matching using the combination of graphs and SDRs. In section 3.1, we describe the process of object detection and the features extracted using OpenCV. In section 3.2, we describe how to form a hierarchical graph from the detected objects as nodes with a fixed-radius nearest neighbors algorithm. Then in section 3.3, we discuss the possibility of representing graphs in SDRs and leveraging the massive parallelism, for example in massively parallel associative memories, that is enabled by SDRs. We present two algorithms using SDRs for exact and approximate matching in section 3.4. Finally, we show a specific example of applying SDRs to the simple 2D Blocks World images such as triangles and rectangles, which can be combined in different ways to form complex images.

3.1 Object detection

Object detection in still or moving images is a complex task. Given an image or a region of interest (ROI), the goal of object detection is to find the locations of objects in the image and to classify them. Object detection is a widely studied problem for which numerous methods have been proposed [11, 22]. Object detection has many applications such as facial recognition, pedestrian detection, etc. [21, 23].

In this paper, we are using OpenCV to locate the objects in the image. Contours, instead of bounding boxes, are used to find objects and are generated using OpenCV's findContours() function, they are defined by a simple, joined curve of continuous data points along the object boundary. Finding contours works best when there is a reasonable contrast between the objects and the background. It also helps if the background is not cluttered and the objects do not partially occlude one another, though approximate SDR matching does handle partial occlusion.

Figure 4 shows the detected contours of the coins in green and each contour is an object. With the help of OpenCV contours, we can also determine other characteristics of the objects such as moments, area, perimeter, and bounding rectangles. For our 2D blocks world application, we compute the object attributes from contour features such as centers, height, width, angle with the x and y-axis, etc.

Figure 4: Contours detected for the objects in an image

3.2 Hierarchical Graph Construction

Traditional computer vision techniques do not capture the locality and connectivity of the objects [30]. Traditional (pre-Deep Network) systems find complex information associated with each feature, using feature detection algorithms such as SIFT. Then the discovered features are matched somewhat independently to a set of features associated with each object, which has been termed a "bag of features" approach. And even if the arrangement or orientations are distorted in an image besides just considering the presence of some certain objects, the methods give unsatisfactory results [22, 26, 28, and, 29].

In this paper, we are representing the images using graphs so that we get better accuracy with simple or complex images. We are constructing a hierarchical graph for an image with the help of a fixed-radius nearest neighbors algorithm. It is clear that biological vision, at least in mammals, takes advantage of the geometric relationships of the features with each other, which we refer to as the "structure" of the object. It appears that visual cortex at the lowest level of the processing hierarchy stores information about tiny sections of the visual field such as edges and corners [1]. These low-level patterns are recombined at higher levels for more complex components.

In this paper, we assume that all the detected contours in the image are parts of much more complex objects. This assumes a simple structural hierarchy. The number of levels can vary based on the application. In our 2D blocks world, we are only assuming three levels of hierarchy. The first level of the graph is the OpenCV detected parts, which are treated as nodes, the second level's nodes are the objects made of these parts based on their proximity with each other and the third level is the image itself.

The first level of the graph (showing the spatial relationships between components) is constructed using a fixed-radius nearest neighbors algorithm applied to the parts' centers. We calculate the connected components of the graph. In graph theory, a connected component of an undirected graph is a subgraph in which any two vertices are connected to each other by more than one path, and which is connected to no additional vertices in the super-graph [53]. Figure 5 shows an example of a graph with three connected components. These connected components are the objects present in the image. Therefore, the number of connected components is equal to the number of objects present in the image.

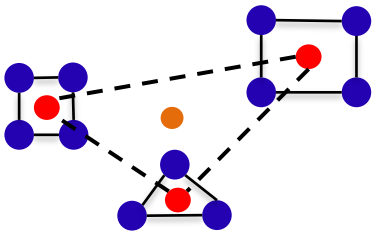

Figure 5: Hierarchical graph with 3 connected components

The second level of the graph is constructed between these objects by applying the fixed-radius nearest neighbors algorithm to the new centers which are calculated from the spatial arithmetic mean of the old centers. The levels can also be increased by applying the fixed-radius nearest neighbors algorithm to the new calculated centers from arithmetic means for the previous level centers.

3.3 Hierarchical Sparse Distributed Representation

An SDR (Sparse distributed representation) is a large binary vector with mostly 0's [52]. Each bit generally carries some semantic meaning, so if two SDRs have more than a few overlapping 1's, then those two SDRs have similar meanings [52]. We can encode any type of data into an SDR while observing this aspect of the data. When a new input is presented, it should contribute to similarity. However, there is no single fixed approach to encoding ("sparsifying") the data into an SDR. An effective encoder should capture as much information on data as possible, which will be different for different types of data. Purdy, Scott [52] discusses several objectives, which should be considered while encoding the data, and it also presents a few encoder examples.

In section 3.2, we described how we generate a hierarchical graph for the image. In this section, we will describe how we are encoding graph information into SDRs. Sharing representations in a hierarchy leads to a generalization of expected behavior. The patterns learned at each level are reused when combined in novel ways at higher levels [1]. The higher levels inherit the properties of lower level components. It makes the computation faster and also reduces memory requirements [1].

For the graphs we are using, SDRs are determined bottom up. First, we compute the SDRs for the lowest level and then take a union of them to form the higher levels. As we mentioned earlier, we are only considering three levels in this paper. For level one, all the detected contours in the image, which are the components at that level, will have a separate and distinct SDR. The fields and length of the SDR are fixed for all the nodes and levels. We compare and operate on SDRs bit-by-bit, with each bit having a semantic meaning so we do need the SDRs of the same dimensionality.

The significance of SDR in a graph is that a single node's SDR will be able to store its own information as well as its neighbors'. The neighbors are defined from the one-hop connectivity. While designing the encoder, we fix the number of nodes a node can be connected to. In this paper, we design an encoder which encodes and stores the graph nodes' attributes into the SDR. Here we will be dealing with the block polygons in a simple 2D "blocks world" image space.

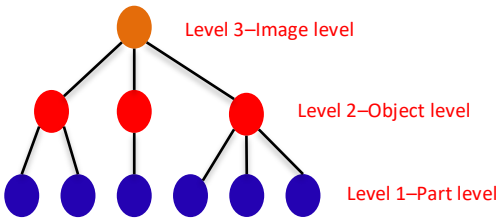

Figure 6: Figure 5 corresponding graph hierarchy

The attributes are defined as the number of edges, the height-width ratio, and connectivity (number of neighbors). To store the relative positions of neighbors, we compute an angle between the node and the neighbor node. The final SDR of a node consists of five fields as described in Figure 7. An SDR has two fields, one for the node and another for the connected neighbors. Each field has sub-fields to store the node's attributes and the neighbor node's relative positions.

The two considerations for encoding the data into the SDR are described below:

a. SDRs should be sparse. The sparsity for encoders can vary but should be relatively fixed for a given application of an encoder [52]. A very rough rule of thumb is that the number of 1's should be the log_2 of the dimension. For this, we assign each field of an SDR a fixed number of bits assuming b and keeping

only *w* bits ON. This way, each dimension is sparsified by a *w/b* factor.

b. The use of SDRs should be mostly independent of the indexing scheme representing the graph, for example, the adjacency list or matrix. A single SDR should have a reasonable knowledge of its surroundings, regardless of the predefined indexing. Having a certain level of independence in the indexing is important for the usefulness of SDR for pattern matching. When storing the neighbors' attributes into the SDR, we process them in the clockwise direction, keeping a particular, invariant, geometric coordinate as the reference.

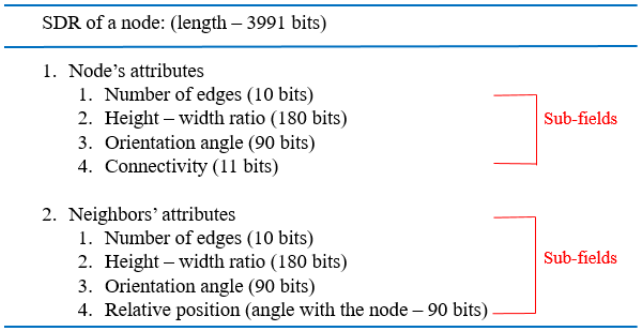

Figure 7: A graph node's SDR organization

would be (assuming x for the number of edges and y for the connectivity):

$$x = 0000010 \ldots 0$$
$$y = 000010 \ldots 0$$

2. For height, width and angle, we assume *b* bits for each field. Among these *b* bits, we set *w* bits 'ON', making the sparsity *w / b*. Starting from the calculated index *i*, we will take *w* consecutive bits and set them 'ON'.

For a given value *v,* bucket *i* (that the number falls into) is calculated from the approach described below [52]:

a. Calculate the range as $\boldsymbol{range = maxValue - minValue}$

b. Choose the number of buckets into which we will split the values.

c. For a given value *v*, the index *i* is computed as:

$$i = floor[buckets * \frac{v - minValue}{range}] \quad (1)$$

The final SDR of a node of a graph's lowest level is a fixed length binary representation assuming length as *l* and the number of 'ON' bits as *o*.

$$l = s + b + c + c\,(s + 2b) \quad (2)$$
$$o = 1 + w + 1 + c\,(1 + 2w) = (2 + w) + c\,(1 + 2w) \quad (3)$$

Node's attributes				Neighbor 1's attributes				Neighbor 2's attributes			
Number of edges	Height-width ratio	Orientation angle	Connectivity	Number of edges	Height-width ratio	Orientation angle	Relative position	Number of edges	Height-width ratio	Orientation angle	Relative position
001…00	00…1…00	00…1…00	0010000	0001…00	00…1…00	00…1…00	00…1…00	00001…00	00…1…00	00…1…00	00…1…00

Figure 8: A Sparse Distributed Representation

This way, we can compare the SDRs of two different nodes and find similarity metrics between the two. Therefore, as desired, the usefulness of SDRs becomes independent of how the nodes in the graph were originally indexed.

To understand the encoder algorithm, assume a 2D planar labeled graph, *G*, having *n* nodes, where every node is connected sparsely to, at most *m,* other nodes, and the nodes are labeled with their attributes (such as number of edges, height, width and connectivity, information that is easily attainable from OpenCV). Each node will have a distinct SDR. Lengths of the SDR's fields and the number of ON bits to represent the data in the fields are fixed.

Each field is converted into a sparse representation using two criteria:

1. For the number of edges and connectivity of the nodes, we assume *s* and *c* bits respectively. This means that we are limiting the maximum number of sides a polygon can have and to how many other nodes it can be connected. Among these *s* and *c* bits, we only set one bit 'ON'. The sparsity of these fields is *1/s* and *1/c* respectively. This single set index bit is the number next to itself as we number the fields' index from 0.

For example: If a pentagon node is connected to four other nodes then the set bits will be the 5th *s* bit and the 4th *c* bit. And, the fields

Example: Let's say we have a graph of 3 nodes. For a node (triangle) which is connected to the other 2 nodes (rectangle and pentagon), figure 8 represents the SDR (the maximum number of neighbors are assumed as 6).

3.3.1 Sparse distributed representation for higher levels

After calculating the SDRs for each and every node of a level 1 graph, we move up to the hierarchy. For level 2, to determine the SDRs of the nodes, we combine the SDRs of level 1 by taking the connected component nodes. We perform 'union' operations for every node present in level 2 and these union SDRs represent a hierarchical graph's structure. For example, assume we have a graph that consists of nine objects and three connected components. We compute three SDRs for level 2 by performing 'union' operations on the objects which belong to the connected components. These three SDRs are the fixed-length binary representation of level two graph.

By the union property, a single SDR is able to store a dynamic set of elements, so when we see the final SDR after performing the union, it has the information presented in the component node SDRs. We can also represent the whole graph in a single SDR by taking the union of all its nodes' SDRs. This resultant SDR will have relevant information about the graph and represents our level 3, which is the entire image.

Even if we have more than three levels in the graph we still only need to take this bottom-up approach: calculate the SDR for the lowest level and then start combining (union operation) the SDRs for higher levels motivated by the approach our brain takes when processing a piece of new visual information.

The SDRs of image-graphs have three important characteristics, which allow them to achieve their goal of fast pattern matching in graphs.

- Each bit in an SDR has semantic meaning.
- Computations with SDRs are independent of the indexing in graphs and their components.
- SDRs are also sparse enough to reduce spatial complexity.
- The SDRs form a representation that contains the "structure" of the object and so is useful in downstream object recognition.

3.4 Graph Matching

Object recognition is the primary operation of any computer vision system. One obvious method of recognizing an object is by comparing it to a database of known objects, template matching is an example of this approach. One way to incorporate more flexibility into the recognition process is to represent objects by graphs, which incorporate the structural information in the image. For example, in computer vision, graphs have been shown to be a useful tool for representing images. Labeled graphs can capture significant amount of information on the "structure" of objects. Using graphs, object recognition requires graph matching [8, 27, 28, 29, and, 30]. Graph isomorphism, which is also known as exact graph matching, in the area of image recognition. This problem is known to be solved in non-polynomial time, but here we are proposing a new method for solving approximate graph isomorphism to reduce the complexity of pattern matching by combining graph analytics and sparse distributed representations. The algorithm is heuristic.

Graph isomorphism can only be applied when the number of nodes in the graphs are the same. Therefore, we check the number of nodes in the graphs' level 1, if equal; we check the isomorphism between the level 1 SDRs. If not, we move to level 2 and calculate the sub-graphs of the bigger graph. The sub-graphs respect the hierarchy. We check the isomorphism for all the sub-graphs whose number of nodes are equal to the smaller graph's nodes. If the smaller graph exists in the bigger graph, the graph is sub-graph isomorphic.

The computational savings come at the cost of capturing and representing more complete information in the SDR. Although, SDR vectors are large the operations using SDRs depend on the number of active bits, which are much fewer than the total number of bits. This is an advantage of sparse representations. SDR vectors contain most of the information about the objects' geometries and the structure of an image. More information can be added based on the application and the dataset. Increasing information improves robustness. However, this comes with the cost of more false positives [2]. We realize the match between the SDRs using SDR's union property and a threshold Θ. Decreasing Θ also results in more false positives. One advantage of the union property is that there is no risk of false negatives since the overlap gives the perfect match if the SDR is within the set. However, it does increase the chance of false positives [2], by increasing the number of active bits in the resultant SDR.

With the help of SDRs, we have developed a powerful heuristic search for graph isomorphism in O(l) time, l is the SDR length which is a constant in our case. A variation of exact match isomorphism is called subgraph isomorphism. Here one must determine whether a graph contains a subgraph, which is isomorphic to another graph. This problem is also known to not be solvable in polynomial time. Here, we choose k nodes' subgraph out of a big graph of n nodes in $O(n^k)$ time and with the help of SDRs, do the matching in $O(1)$. The k nodes' subgraphs respect the hierarchy. For efficient image matching, an SDR should be invariant to position, scale, brightness and, rotation of an object. In this paper, our SDR provides both scale and position invariance. The graph-matching algorithm using our SDR is shown in Figure 9. In the future, we can apply this technique of merging graph matching and SDRs to find a solution for probabilistic matching. We can also use the techniques to find matching patterns in an image using associative memory.

Algorithm Graph Matching

Input First level image graphs (G_A, G_B) and their 2D SDR arrays (S_A, S_B) with n_A and n_B nodes.

Output Whether graphs are matched or not. If matched, they are isomorphic or sub-graph isomorphic.

```
1:  procedure PatterMatching
2:
3:      /* If number of nodes are equal - check graph isomorphism*/
4:      if n_A == n_B then
5:          Ans ← GraphIsomorphism(S_A, S_B)
6:      /*else - check sub-graph isomorphism */
7:      else if n_A > n_B then
8:          calculate level 2 graph G_A2 and a 2D SDR array S_A2 for graph G_A
9:          for i in len(S_A2) do
10:             if number of nodes in S_A2[i] = n_B then
11:                 Ans ← GraphIsomorphism(S_A2[i], S_B)
12:             end if
13:         end for
14:     else then
15:         calculate level 2 graph G_B2 and a 2D SDR array S_B2 for graph G_B
16:         for i in len(S_B2) do
17:             if number of nodes in S_B2[i] = n_A then
18:                 Ans ← GraphIsomorphism(S_A, S_B2[i])
19:             end if
20:         end for
21:     end if
22:
23:     if Ans == 'Yes' then
24:         "Graphs are sub-graph isomorphic"
25:     else then
26:         "Graphs are not sub-graph isomorphic"
27:     end if
28: end procedure
29:
30: procedure GraphIsomorphism
31:     calculate union of the SDR arrays and create 1D SDRs
32:     Take dot product of S_A and S_B to get the overlap score for the SDRs to
    determine the similarity.
33:     We realise a match between the SDRs if their overlap exceeds some
    threshold θ.
34: end procedure
```

Figure 9: Graph matching algorithm

4. Experiments

We start with 2D Blocks World images in our experiment. Here we aim to show that our method detects objects and generates the hierarchical graph in an image (section 4.1) which is used to create sparse distributed representations of all the components of the graphs (section 4.2). Further, the resulting SDRs for the images are

used in graph matching (section 4.3). The algorithm in Figure 9 is tested on a number of 2D blocks world images which were generated randomly with some specific directions to meet our application's requirement. Here, we show the results of applying the algorithm to a few images.

4.1 Object Detection and Graph Generation

In this section, we are showing the detected objects and their generated graphs for the blocks world images. In figure 10, we show two images with only one object made of composite parts. In figure 10 a) is the image and b) is the generated graph. Here the graph is only between the object's parts and how they are connected to each other. Figure 11 shows two image graphs with more than one objects, made of some parts, far enough to be separate objects. The lowest level (level 1, represented by blue) of the graph represents connectedness between the basic detected parts, which, in turn, make complex objects in the image. The second level (represented by red) shows the graph between more complex objects. The third level is the image itself. Because of our simple Blocks World images and to illustrate algorithm operation, we assume three levels of hierarchy. However, the number of levels can be increased with the complexity of an image.

4.2 Sparse Distributed Representation

The generated SDRs are large binary vectors representing the important attributes of the objects. Each detected part in the image has an SDR of length *l*. The length of the SDRs is large compared to the active number of bits. For limiting the size of the SDR, we assume that maximum connected nodes and maximum number of edges for a node are 10. The height and width can be in a range from 1 to 360. To represent the very sparse SDRs, we show only the indices of ON bits. The computation with SDRs is memory and time efficient as the computation happens only with the active bits. Figure 12 shows the SDR of object 0 in figure 11's image graph 2. Each and every bit has semantic meaning. Starting some bits represent the object information and rest of the bits represent the connected neighbors' information and their relative position with the object.

```
[   3   97   98   99  100  101  102  282  294  316  317  318  319  320
  321  322  323  408  409  410  411  412  413  421  422  423  424  425
  426  581  582  583  584  585  586  590  591  592  593  594  595  626
  627  628  629  630  631  664  686  687  688  689  690  691  692  693
  777  778  779  780  781  782  791  792  793  794  795  796  977  978
  979  980  981  982 1005 1006 1007 1008 1009 1010 1022 1023 1024 1025
 1026 1027]
```

Figure 12: SDR with active bits indices.

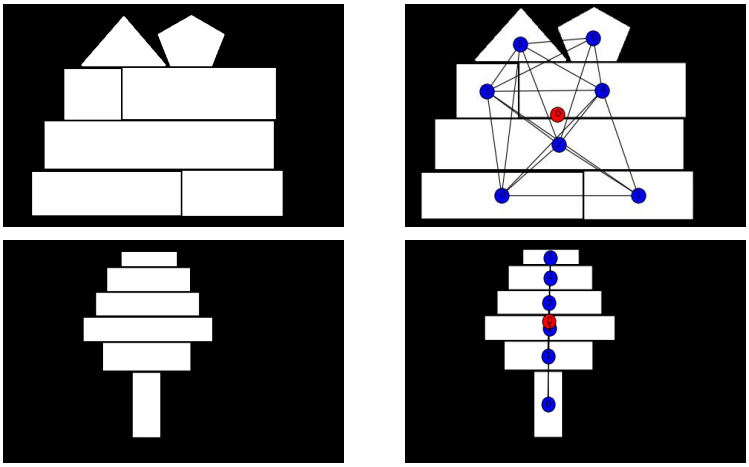

Figure 10: Images with one object and generated graphs.

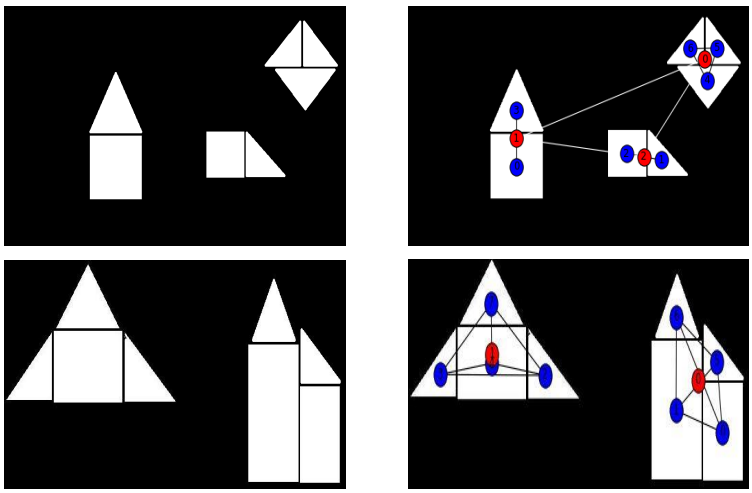

Figure 11: Images with multiple objects and their generated graphs

4.3 Graph Matching

In this section, we calculate the match between the generated graphs using SDRs overlap. In figure 13, we take two graphs and check whether the first graph contains a graph, which is isomorphic to the second graph. This demonstrates whether the object present in the second image exists somewhere in the first image. Here, we also show that this check is independent of the graph/object indices. As one can see in the images, some of the detected parts in the second image are indexed differently indices in the first image, which does not affect the final result. This match also demonstrates the scale and position invariance. For the graphs in figure 13, image 1 and image 2 have two and one object respectively in level 2 represented by red color. We take one object SDR of graph 1 at a time and compare it with the graph 2 SDR, which realizes a match.

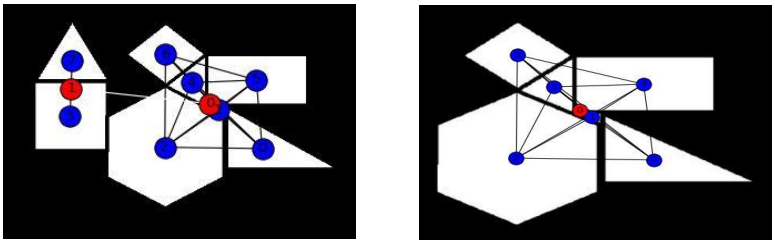

Figure 13: Two graphs with sub-graph isomorphism

For the given images in figure 13 the SDR overlap exceeds the threshold. We conclude that graph 2 is sub-graph isomorphic to graph 1 which also means that the object in the second image exists in the first image.

It should be noted, that such matching can be done in a straightforward manner by cortical-like associative memories.

4.4 Result Analysis

Table 1: Result Analysis: Techniques and their complexity

Algorithm	Complexity
Graph Isomorphism	NP-intermediate
Sub-graph Isomorphism	NP-Complete
Approximate Graph Isomorphism w/SDR	*O(1)*
Approximate Sub-graph Isomorphism w/ SDR	Choosing a *k* node subgraph out of a big graph with *n* nodes – $O(n^k)$ and matching subgraph with k nodes is *O(1)*

5. Conclusion

Object recognition continues to be the most important capability in computer vision. Traditional object recognition techniques were based on capturing complex features, but the features were mostly treated as unrelated in any way, the "bag of features" approach. The actual structure of the features with respect to each other was rarely attempted, though there has been some work in this area [43, 46]. The bag of features approach loses important information about the structural relationships of the features with respect to each other, for example, the spatial relationship between the limbs of an animal or the formation and shape of vehicles. The structure captured by our SDR contains important information that may help with object recognition and complex variations of it are most likely used in primate vision. Deep networks appear to limit how much structure they capture. And, they are easily fooled with minor modification to test images [48]. These failures often have to do with a common pattern in an arbitrary position a "bag of features" kind of mistake.

Graph techniques, when paired with biologically inspired characteristics, have the potential to be an effective method for object recognition. These techniques leverage the information about the connectedness between the features, i.e., the "structure" of an image rather than the traditional methods in which we have no connectivity between features and objects of the image.

In this paper, we have presented a novel technique to perform object detection and pattern matching in images with the help of graph algorithms and Neuromorphic computing techniques. With these techniques, we can identify connections in images and represent those as graphs. This enables us to use many graph-based algorithms for this pattern matching in images. We showed that we can perform approximate graph matching in *O(1)* time with the SDR representations, and further choose *k* nodes subgraph in $O(n^k)$ and perform subgraph matching with *O(1)*, whereas the classic techniques take a non-polynomial amount of time. Moreover, we can also identify partial matching in images based on inherent properties of SDRs. This work shows a way of using graph-based techniques for object recognition related tasks in images and demonstrates the use of Neuromorphic computing techniques for providing orders of magnitudes of speedups.

The next step in this work is to map the derived SDR to a biologically inspired associative memory, which will allow us to do approximate object mapping in the case where parts of the objects are occluded or noisy.

ACKNOWEDGEMENT

This work was supported in part by the Center for Brain Inspired Computing (C-BRIC), one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

REFERENCES

[1] Hawkins, J., Ahmad, S. and Dubinsky, D., 2010. Hierarchical temporal memory including HTM cortical learning algorithms. Technical report, Numenta, Inc,

Palo Alto, http://www.numenta.com/htmoverview/education/HTM_CorticalLearningAlgorithms.pdf.
[2] Ahmad, S. and Hawkins, J., 2015. Properties of sparse distributed representations and their application to hierarchical temporal memory. arXiv preprint arXiv:1503.07469.
[3] Rinkus, G.J., 2010. A cortical sparse distributed coding model linking mini-and macrocolumn-scale functionality. Frontiers in neuroanatomy, 4(17).
[4] Zhu, S. and Hammerstrom, D., 2002, November. Simulation of associative neural networks. In Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on (Vol. 4, pp. 1639-1643). IEEE.
[5] Albarelli, A., Bergamasco, F., Rossi, L., Vascon, S. and Torsello, A., 2012, November. A stable graph-based representation for object recognition through high-order matching. In Pattern Recognition (ICPR), 2012 21st International Conference on (pp. 3341-3344). IEEE.
[6] Xirouhakis, Y., Tirakis, A. and Delopoulos, A., 1999. An efficient graph representation for image retrieval based on color composition.
[7] Ullmann, J.R., 1976. An algorithm for subgraph isomorphism. Journal of the ACM (JACM), 23(1), pp.31-42.
[8] Zhu, Y., Qin, L., Yu, J.X., Ke, Y. and Lin, X., 2013. High efficiency and quality: large graphs matching. The VLDB Journal, 22(3), pp.345-368.
[9] Suga, A., Fukuda, K., Takiguchi, T. and Ariki, Y., 2008, December. Object recognition and segmentation using SIFT and graph cuts. In Pattern Recognition, 2008. ICPR 2008. 19th International Conference on (pp. 1-4). IEEE.
[10] Prasad, B.D.C.N., PESNK, P. and Sagar, Y., 2010. A study on associative neural memories. Int J Adv Comput Sci App, 1, pp.124-133.
[11] Prabhu, Nikita, and R. Venkatesh Babu. "Attribute-graph: A graph based approach to image ranking." Proceedings of the IEEE International Conference on Computer Vision. 2015.
[12] Zhang, Zizhao, et al. "Revisiting graph construction for fast image segmentation." Pattern Recognition 78 (2018): 344-357.
[13] Johnson, Justin, et al. "Image retrieval using scene graphs." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
[14] Li, Li-Jia, et al. "Object bank: A high-level image representation for scene classification & semantic feature sparsification." Advances in neural information processing systems. 2010.
[15] Aditya, Somak, et al. "From images to sentences through scene description graphs using commonsense reasoning and knowledge." arXiv preprint arXiv:1511.03292 (2015).
[16] Schuster, Sebastian, et al. "Generating semantically precise scene graphs from textual descriptions for improved image retrieval." Proceedings of the fourth workshop on vision and language. 2015.
[17] Xu, Danfei, et al. "Scene graph generation by iterative message passing." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Vol. 2. 2017.
[18] Li, Yikang, et al. "Scene graph generation from objects, phrases and region captions." ICCV. 2017.
[19] Plaza, Laura, Elena Lloret, and Ahmet Aker. "Improving automatic image captioning using text summarization techniques." International Conference on Text, Speech and Dialogue. Springer, Berlin, Heidelberg, 2010.
[20] Stenroos, Olavi. "Object detection from images using convolutional neural networks." (2017).
[21] Zhang, Feihu, et al. "Segment graph based image filtering: fast structure-preserving smoothing." Proceedings of the IEEE International Conference on Computer Vision. 2015.
[22] Zitouni, Hilal, et al. "Re-ranking of web image search results using a graph algorithm." Pattern Recognition, 2008. ICPR 2008. 19th International Conference on. IEEE, 2008.
[23] Song, Yi-Zhe, et al. "Finding semantic structures in image hierarchies using Laplacian graph energy." European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2010.
[24] Hsieh, Liang-Chi, et al. "Canonical image selection and efficient image graph construction for large-scale flickr photos." Proceedings of the 17th ACM international conference on Multimedia. ACM, 2009.
[25] Malmberg, Filip, et al. "A graph-based framework for sub-pixel image segmentation." Theoretical Computer Science 412.15 (2011): 1338-1349.
[26] Lézoray, Olivier, and Leo Grady. Image processing and analysis with graphs: theory and practice. CRC Press, 2012.
[27] Sanfeliu, Alberto, et al. "Graph-based representations and techniques for image processing and image analysis." Pattern recognition 35.3 (2002): 639-650.
[28] Shokoufandeh, Ali, and Sven Dickinson. "Graph-theoretical methods in computer vision." Theoretical aspects of computer science. Springer, Berlin, Heidelberg, 2002.
[29] Camilus, K. Santle, and V. K. Govindan. "A Review on Graph Based Segmentation." International Journal of Image, Graphics & Signal Processing 4.5 (2012).
[30] Lézoray, Olivier, and Leo Grady. Image processing and analysis with graphs: theory and practice. CRC Press, 2012.
[31] Gruen, Armin, Emmanuel P. Baltsavias, and Olof Henricsson, eds. Automatic extraction of man-made objects from aerial and space images (II). Birkhäuser, 2012.
[32] Johnson, Justin, Agrim Gupta, and Li Fei-Fei. "Image generation from scene graphs." arXiv preprint (2018).
[33] De Sousa Webber, Francisco. "Semantic Folding Theory And its Application in Semantic Fingerprinting." arXiv preprint arXiv:1511.08855 (2015).
[34] George, Dileep. How the brain might work: A hierarchical and temporal model for
learning and recognition. Stanford: Stanford University, 2008.
[35] Hawkins, Jeff, and Sandra Blakeslee. On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines. Macmillan, 2007.
[36] Hawkins, Jeff, and Dileep George. Hierarchical temporal memory: Concepts, theory and terminology. Technical report, Numenta, 2006.
[37] Pan, Jia-Yu, et al. "Gcap: Graph-based automatic image captioning." Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on. IEEE, 2004.
[38] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision60.2 (2004): 91-110.
[39] Lowe, David G. "Object recognition from local scale-invariant features." Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, 1999.
[40] Li, Li-Jia, et al. "Object bank: A high-level image representation for scene classification & semantic feature sparsification." Advances in neural information processing systems. 2010.
[41] Oliva, Aude, and Antonio Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope." International journal of computer vision 42.3 (2001): 145-175.
[42] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.
[43] Nowak, Eric, Frédéric Jurie, and Bill Triggs. "Sampling strategies for bag-of-features image classification." European conference on computer vision. Springer, Berlin, Heidelberg, 2006.
[44] Jiang, Yu-Gang, Chong-Wah Ngo, and Jun Yang. "Towards optimal bag-of-features for object categorization and semantic video retrieval." Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007.
[45] Ngo, Chong-Wah, Yu-Fei Ma, and Hong-Jiang Zhang. "Video summarization and scene detection by graph modeling." IEEE Transactions on Circuits and Systems for Video Technology15.2 (2005): 296-305.
[46] Faheema, A. G., and Subrata Rakshit. "Feature selection using bag-of-visual-words representation." Advance Computing Conference (IACC), 2010 IEEE 2nd International. IEEE, 2010.
[47] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. CVPR, 2006.
[48] Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
[49] Yu, Tianshu, et al. "Joint Cuts and Matching of Partitions in One Graph." arXiv preprint arXiv:1711.09584 (2017).
[50] Zanfir, Andrei, and Cristian Sminchisescu. "Deep Learning of Graph Matching." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[51] Cui, Yuwei, Subutai Ahmad, and Jeff Hawkins. "The HTM spatial pooler—a neocortical algorithm for online sparse distributed coding." Frontiers in computational neuroscience11 (2017): 111.
[52] Purdy, Scott. "Encoding data for HTM systems." arXiv preprint arXiv:1602.05925 (2016).
[53] Wikipedia contributors. "Connected component (graph theory)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 May. 2018. Web. 3 Jan. 2019
[54] Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." Advances in neural information processing systems. 2017.
[55] Felzenszwalb, Pedro F., et al. "Object detection with discriminatively trained part-based models." IEEE transactions on pattern analysis and machine intelligence 32.9 (2009): 1627-1645.