NTNU – Trondheim
Norwegian University of
Science and Technology

Use of Clustering to Assist Recognition in Computer Vision

Ole Kristian Braut Grashei

Master of Science in Computer Science
Submission date: June 2013
Supervisor: Richard E. Blake, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Many important problems in computer visions are in NP time. Initial processing using polynomial time algorithms can help reduce the search space for the main NP problem. Clustering is a polynomial algorithm, recognition by graph matching is an NP problem. The thesis will investigate the use of clustering to assist recognition of objects.

Supervisor: Richard E. Blake

Abstract

In computer vision many problems are of non-deterministic polynomial time complexity. One of these problems is graph matching. Suboptimal solutions have been proposed to efficiently do graph matching. This thesis investigates the use of unsupervised learning to cluster structured graph data in polynomial time. Clustering was done on attributed graph nodes and attributed graph node-arc-node triplets, and meaningful results were demonstrated. Self-organizing maps and the minimum message length program Snob were used. These clustering results may help a suboptimal graph matcher arrive at an acceptable solution at an acceptable time. The thesis proposes some methods to do so, but implementation is future work.

Sammendrag

Mange problemer i datasyn er av ikke-deterministisk polynomisk kompleksitet. Et av disse problemene er graf matching. Suboptimale metoder har blitt foreslått til å gjennomføre graf-matching. Denne oppgaven undersøker bruken av ikke-veiledende læring til å klassifisere strukturert graf-data i polynomisk tid. Klassifisering av noder og av node-kant-node-grupperinger ble gjennomført, og meningsfulle resultater ble demonstrert. Selv-organiserende kart og minimums beskjedlengde programmet Snob ble brukt. Disse klassifiseringsresultatene kan hjelpe en suboptimal graf-matcher til å finne en akseptabel løsning på en akseptabel tid. Oppgaven foreslår noen metoder for å gjøre så, men implementasjon er fremtidig arbeid.

Preface

This project was defined in consultation with my supervisor, professor Richard E. Blake at the department of Computer and Information Science at the Norwegian University of Science and Technology. I would like to thank him for his time used to guide and help me during this project. This project was carried out during the 2013 spring semester.

I would also like to thank my family for support and encouragement during my years in education.

Ole Kristian Braut Grashei
June 11, 2013

Contents

List of Figures

List of Tables

1 Introduction

In many natural problems graphs can be used to model the knowledge. Protein structure[1], communication networks such as the Internet, social networks, linguistic structure, electronic circuits[2] and many more problems are modeled using graphs. Graphs provide an intuitive way of modeling knowledge and the relationship between entities that is close to real world representation. Unfortunately for many graph problems there are no fast solutions known. One of these problems is graph matching.

Graph matching is useful in many of the earlier mentioned areas e.g., predicting protein behavior[3] and in computer vision an unknown object can be matched with objects from a database. Graph matching can generally be said to be of non-deterministic polynomial complexity, although certain instances of the problem can be solved in polynomial time[4]. This means the problem of graph matching belongs to a class of problems believed to be computably infeasible. Symbolic vision systems based on graphs have for this reason met some resistance.

We know the human brain is excellent at object recognition and therefore sub-symbolic methods are very interesting in this domain. We observe that the brain is able to generalize over huge amount of data and recognize patterns, but its ability to do exact calculations are less impressive. Some approximations in a vision systems may therefore be acceptable and systems have been made that use approximation techniques together with graph representation of the objects[5, 6].

In this thesis we will use clustering algorithms that are known to be of polynomial complexity to assist a graph matching vision system. We hope that clustering will be able to effectively reduce the input to graph matching algorithms and therefore effectively reduce run-time, and making the graph matching approach seem more interesting.

We have selected two clustering techniques and applied them to a large dataset of graph data constructed from a series of laboratory images of a toy truck. The results were analyzed and we observe meaningful results by the measurement defined.

2 Background

In this section we introduce some necessary concepts before we move on to implementation and experimentation. A short introduction to graphs is given, where only the basics that will be needed for our reasoning are presented. Also an overview of the dataset we will use and the computer vision system we want to improve upon is given. An overview of the concepts of clustering and classification is given and then finally we investigate if any similar work has been done.

2.1 Complexity Classes

In computational complexity theory, the complexity class P is solvable in deterministic polynomial time. The complexity class NP is solvable in non-deterministic polynomial time. By the definition of the Turing machines P is a subset of NP. Some NP problems exists where there are no known P solutions, these are known as NP-complete. Some problems are known to be at least as hard as the hardest problems in NP, these are known as NP-hard.

One of the open questions of computer science today is whether P equals NP or not. If P do equal NP a wide variety of problems will have a much faster solution than we know of today.

For now we have no fast ways to solve problems that are NP-complete or NP-hard.

2.2 Graphs

A graph $G = (V, E)$ consists of edges and vertices. V is the set of vertices, sometimes also called nodes or leafs. E is the set of edges between the vertices, sometimes also called arcs or links. $|V|$ is the graph order (number of vertices), and $|E|$ is the size of the graph (number of edges). Edges may be directed or undirected.

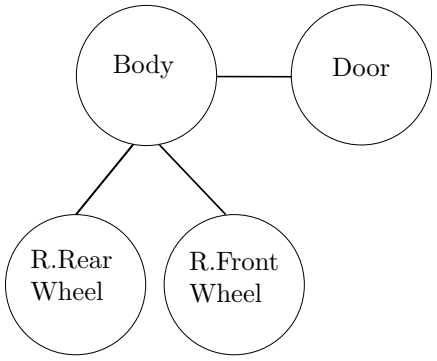


Figure 1: A simple vertex-labeled graph of a truck

Vertices and edges may contain information, if they have a simple label they are known as labeled. See fig. 1 for a vertex-labeled graph. The vertices and

edges may also contain more information as attributes, they are then known as attributed. Fig. 2 shows a vertex-attributed graph.

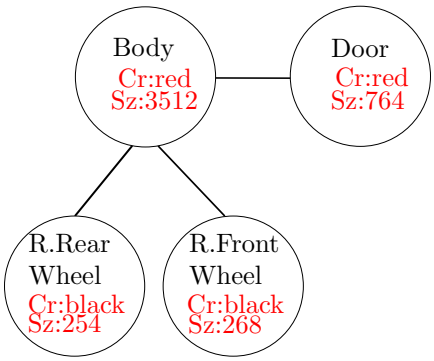


Figure 2: A vertex-attributed graph of a truck with size (Sz) and color (Cr)

2.2.1 Graph Matching

Matching two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $|V_1| = |V_2|$ can be defined as finding a one-to-one mapping $f : V_1 \to V_2$ such that $(u, v) \in V_1$ if and only if $(f(u), f(v)) \in V_2$. This is known as a graph isomorphism. If we have $|V_1| \neq |V_2|$ the matching will be a sub-graph isomorphism.

In a computer vision system with attributed graphs we cannot expect to have a model that is isomorphic to the data observed (perfect match). The segmentation process and feature extraction are inevitably subject to noise and inaccuracies. We are therefore concerned with inexact sub-graph matching, meaning that we will have graphs where no isomorphism exists and graphs that might be of different order. Graph isomorphism has yet not been classified within a complexity class, while inexact sub-graph matching is of NP-complete complexity [7]. For a discussion of graph matching problems and complexity see [4, 8].s Various methods have been researched for handling the NP-complete matching problem efficiently, [9] reviews graph matching in pattern recognition.

2.3 Computer Vision System for Object Recognition

For object recognition a computer vision system will do a matching of an unknown object to some known object in a database. In a structural computer vision system graphs are used for knowledge representation, and graph matching is used to classify the unknown object. See fig. 3 for an overview of such a system.

2.3.1 The Truck Dataset

In this thesis we will use a dataset that is generated from a series of photos of a toy truck (see fig. 4). The image series was created by adjusting the camera height and rotating the truck. We have images from 27 different camera height settings. For

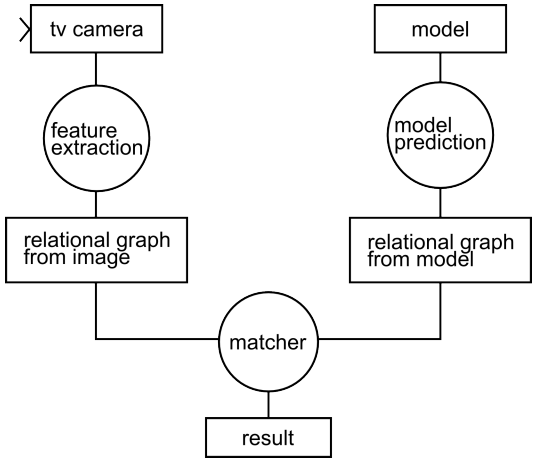
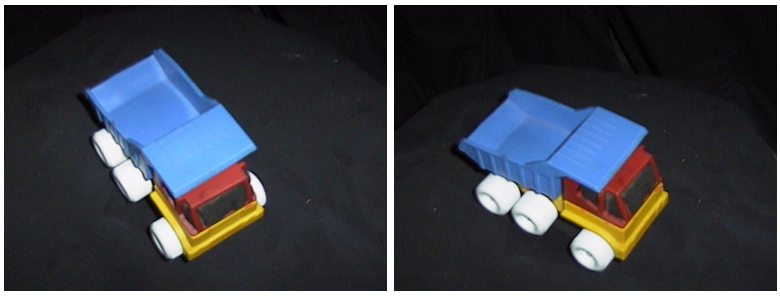


Figure 3: Activities for model directed computer vision. Diagram reproduced from [10].



(a) Truck Image 1004 (b) Truck Image 19006

Figure 4: The toy truck used as a base for our dataset from two views.

each of the 27 height settings, the truck was rotated at 19 intervals from 0 to 180 degrees. This yields in total 513 images of the truck. Each image and associated data are labeled with the two angles used to capture the image. The two angles are represented by numbers 1-27 and 01-19 the labels are built by joining them as a string with a 0 separating them, e.g. '19006'.This labeling is useful for evaluating clustering as we will see in section 3.

2.3.2 Image Segmentation and Graph Construction

Via image processing techniques the photos of the truck have been segmented, features extracted, and relational graphs constructed. See [11] for more information on the feature extraction. Fig. 5 shows the truck in fig. 4b after segmentation and node labeling. The color values in the segmented image are the mean color values for the area found by the feature extractor. Table 1 shows machine code generated

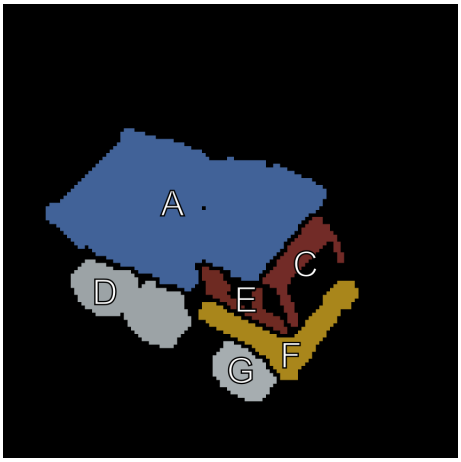


Figure 5: The toy truck segmented and labeled, coloring done by feature extractor.

for each of the segments shown in fig. 5.

Node Label	Machine Readable Code
A	[A[@id=c19006;ext:sqr:aa(2155):px(16049,111794,9096): mm(15570,43,2003,827):cg(51,69):cr(65,98,152)]]
C	[C[@id=c19006;ext:sqr:aa(208):px(5377,4639,167): mm(11747,54,2051,278):cg(83,54):cr(114,43,39)]]
D	[D[@id=c19006;ext:sqr:aa(462):px(20388,8955,253): mm(16087,85,783,62):cg(35,44):cr(156,163,166)]]
E	[E[@id=c19006;ext:sqr:aa(122):px(21479,4424,13): mm(11121,91,8,39):cg(67,44):cr(110,42,34)]]
F	[F[@id=c19006;ext:tri:aa(386):px(14109,20081,1463): mm(13046,55,37738,2479):cg(78,37):cr(169,134,27)]]
G	[G[@id=c19006;ext:sqr:aa(199):px(22603,587,111): mm(20638,43,331,30):cg(66,24):cr(166,173,176)]]

Table 1: Machine readable code representation of nodes with structural features.

The attributes in the code are:

1. External/internal contour flag, *ext* or *int*.
2. Shape classification, *sqr*, *cir* or *tri*.
3. Area, *aa*.
4. Three plane equation coefficients, *px*.
5. Four scale and rotation invariant moments, *mm*.
6. Centroid, *cg*.

7. Mean color values, *cr*.

Table 2 shows a node-arc-node triplet that is part of the relational graph. The graphs in our dataset are fully connected, meaning a relational triplet exists for all node pairs in both directions. The relation arc is attributed with a geometrical relation between the nodes. 16 different directions are used, North, North by North East, North East, North East by East and so on. The arc can also be attributed with an inside or outside flag if one of the contours severely intrude on the other.

Our dataset in this thesis are the nodes and graphs in the form shown in table 1 and table 2 for each of the 513 truck images.

Graph Component	Machine Readable Code
Leading Node	[A[@id=c19006;ext:sqr:aa(2155):px(16049,111794,9096): mm(15570,43,2003,827):cg(51,69):cr(65,98,152)]]
Relation (arc)	[[5][@rep= 1]]
Trailing Node	[C[@id=c19006;ext:sqr:aa(208):px(5377,4639,167): mm(11747,54,2051,278):cg(83,54):cr(114,43,39)]]

Table 2: Machine readable code fragment of an attributed relational graph; a node - arc - node triplet with attributes.

2.3.3 Sorted Table Graph Matcher

Blake proposes a way of handling the NP graph matching problem with the use of a sorted table [12]. The table is a list of every node-arc-node triplet pairing between the two graphs, with each row in the table is a column with the cost of the matching. A control vector column is also inserted into the table, this vector is a simple boolean structure. A one (true) means that this triplet pairing is part of the match, and a zero (false) means the pairing is not part of the match. The object of the matching is then to find a sequence of bits for the control vector that will satisfy the termination criteria. The termination criteria include acceptable run time, acceptable cost and acceptable way of matching, see [12] for a full description.

The table is sorted by cost, and the matching process will start biasing the lowest cost pairings. Finding the sequence of bits for the control vector is of NP complexity and therefore the termination criteria is needed. This method is sub optimal because when the termination criteria is met there is no guarantee for the solution to be a global minimum. See table 3 for an example of the sorted table described here.

2.4 Clustering

Clustering is the process of grouping a set of items together so that they are more similar to each other than they are to items of other groups. A supervised process is provided with the knowledge of what class every item should go into and would learn to classify items based on this information. Clustering, on the other hand, is

Arc/Nodes	Arc/Nodes	Cost	Control Vector
A→C	Z→X	320	1
B→C	Y→X	320	1
A→B	Z→Y	340	1
A→C	Y→X	570	0
B→C	Z→X	580	0
B→C	Y→Z	652	0
A→B	X→Y	794	0
A→B	Y→Z	878	0
A→C	Y→Z	902	0

Table 3: Eight top rows of a sorted table of associations. Reproduced from [12].

used when we have no class information on items and we want to discover underlying structures in the data set. Clustering is an unsupervised process as correct classification is not known and the task of the clustering algorithm is to suggest such a classification of the items.

Some methods will try to find representative items called exemplars in the data, the exemplars are then used to represent the cluster of which it belongs. Unknown items can be classified by comparing it to each of the exemplars by finding the most similar one. A common similarity measure when the data is real-valued is the negative Euclidean distance. Other similarity measures may perform better, but Euclidean distance is simple and works well [13]. By this method we can often reduce the search space required to find a good enough solution. This is one of the ways we might be able to exploit clustering in a computer vision system.

In this thesis we will use two different methods for clustering, minimum message length and self-organizing maps. Both methods have been chosen as they show good results in the literature and for their favorable unsupervised learning process.

2.4.1 Snob - Minimum Message Length Mixture Modeling

Minimum message length (MML) is based on evaluating a model based on how well it is able to compress a message containing the data. To communicate the data the full message must contain both the model and the message given the model. Therefore a more descriptive model will only be chosen if it reduces the total size of the entire message. The function of the model is to provide a good probability distribution for the data to be coded. A better model will reduce the second part of the message.

In a clustering context, the model will describe the different classes. More or less classes might lead to a better model and an overall shorter message. This is how MML will automatically find a appropriate number of clusters. Strict MML is NP-hard, but a heuristic can be used allowing an approximate solution to be found in polynomial time [14, 15].

MML effectively embodies Occam's razor as overly complex models with many parameters will be discarded, even though their better fit of the data will give a

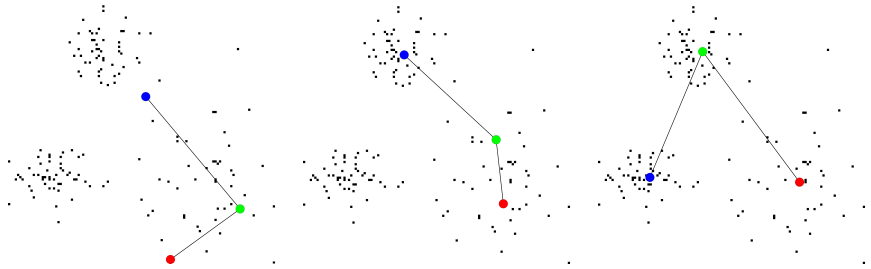
shorter seconds message [16].

MML was first described in 1968 by Wallace and Boulton [17]. Wallace's own implementation is the program known as Snob. Snob does Mixture Modeling (clustering) via MML, see [18] for an overview. In 1996 Upal and Neufeld compared several unsupervised classifiers and Snob was found to perform best [19].

In this thesis we have used Snob to run clustering experiments. The program is available at the Monash University website [20].

2.4.2 The Self Organizing Map

The self-organizing Map (SOM) is a type of artificial neural network (ANN) that is trained in an unsupervised process. SOMs are also known as Kohonen-maps as Kohonen introduced the concept in 1982 [21]. SOMs map m-dimensional input space vectors to an n-dimensional output space. One of SOMs features separating them from other ANNs is their property to preserve topology from input space to output space. For these two properties SOMs are well suited for visualization of high dimensionality data [22]. The SOM is a map of nodes in the output space



(a) t=0 (b) t=7 (c) t=10

Figure 6: SOM with three nodes in one dimension trained over input of three Gaussian distributions in two dimensions. (a) shows the network at random initialization, (b) after seven training runs, and (c) at convergence the SOM has found each distribution's center.

in n-dimensions, each node has a m-dimensional vector that are of the same dimension as the input space. Training is done by calculating a similarity between the training vector (input space) to every node's vector in the map. The most similar node in the map is said to be the winner. The winning node will then move its input space vector closer to that of the input vector by some learning rate. Topology is preserved by that the nodes in the neighborhood of the winner will also be moved toward the training vector. The training is done by starting with a large neighborhood for global organization. Gradually the neighborhood size and learning rate is lowered to allow local organization and finally convergence. After training each node will be an exemplar of a class, inputs can be compared to the nodes in the map, the winning node will then classify the input. With a learning rate of zero SOMs operates rigorously as the well known k-means clustering

algorithm, but SOMs may be better at exploring the full search space [23]. A full overview of the map can be found in [24].

The SOM algorithm can be used on large datasets [25], it scales linearly with the number of input samples and quadratically with number of nodes in the map [26].

See fig. 6 for a simple training run of a SOM. Three Gaussian distributions with different mean and variance was given as input. We can see that the SOM converges to the center of each distribution. Each data point can then be assigned to the cluster of the closest SOM node.

2.5 Previous Work

Graphs are studied widely in the literature, and much interests have been in the domain of structured pattern recognition. In 2004 Conte et al. did a review named Thirty Years of Graph Matching in Pattern Recognition [9]. They mention no work done with clustering directly working to a assist a graph matcher, but clustering was used to do the matching itself. In one paper clustering was used on spectral information extracted from graphs [27], and then graphs was matched if they was in the same cluster.

Graph kernels are used as a way to extract attributes from graphs that may be used for clustering, this method may help in clustering full graphs [28].

A system that use clustering of nodes, but do not include edges in the process, is described in [6]. Not including edges is a great sacrifice of accuracy for speed, and once edges are not include it is no longer a graph being worked on.

To our knowledge there have not been any work done with clustering of graph data including edges to directly assist a graph matcher.

3 Methods and Implementation

In this section we provide details to how we applied our methods to the clustering problem and we introduce a measure for clustering success.

3.1 Clustering With Snob

One of the advantages to Snob is the lack of need for configuration. This makes the method useful for an autonomous recognition system. Once data has been given as input according to documentation, Snob will run the clustering without any more specifications. Snob uses mixture modeling and supports a variety of data types, continuous, discrete, Poisson, and circular. Most of our data variables are of continuous type, although we have the direction of a relation between two nodes that is circular, and we have the inside/outside flag of a relation that is discrete.

Snob will produce the same result given the same input every time, but initial randomized classes can be used to obtain a different set of clusters. We use the random initial classes approach to get more then one clustering result to see if there are any trends.

3.2 Clustering with Self Organizing Maps

A basic SOM was implemented from scratch based on description by Kohonen[24]. The basic SOM use only continuous variables, our version was extended to also support circular variables for the direction attribute.

Support for discrete variables was not implemented as their behavior is not defined in the basic SOM framework. The basic operations in the SOM, moving the vector and calculating similarity is not well defined for discrete states. We therefore decided to ignore the discrete variable when clustering with a SOM. In our implementation we normalize the input vectors, it is not principally needed, but may increase numerical accuracy as the vectors will have same dynamic range [24]. See algorithm 1 for a pseudocode overview of our training algorithm.

```
1: Normalize training data to [0, 1]
2: Randomize SOM input space vectors to [0, 1]
3: for N training runs do
4:     for all training vectors do
5:         Find most similar node in SOM
6:         Move winning node and its neighborhood nodes closer to training vector
   by learning rate
7:     end for
8:     Reduce neighborhood size and learning rate
9: end for
```

Algorithm 1: Pseudocode for training a Self Organizing Map

When running a SOM certain decisions must be made: A distance measure must be chosen. We use the squared euclidean distance to compare our attribute

vectors. Note that the distance calculation has been adapted to support circular variables (angles).

A map size must be chosen, this may be of any size in any number of dimensions. As a general rule if we want the SOM to generalize over the training data we want less nodes in the map than the number of training vectors. We ran our experiments with different map sizes to gouge performance at various settings.

A learning rate and a neighborhood size must be chosen. We start with a somewhat arbitrary learning rate of 0.25 that seems to work well. Neighborhood size was chosen so that its radius is the double of the size of the map. This size ensures global organization from the start as the neighborhood is large enough to include every node from every position in the map. According to [23] the learning rate and neighborhood size must progressively be reduced to zero for robust performance. We do this in a linear fashion.

A number of training runs must be chosen. We used various numbers but it seems in our clustering we don't need a high number of runs.

3.3 A Success Measure via Step Distance

The best way to evaluate a clustering method would be to know the true class of every item clustered. The method's success rate could then be calculated. For our dataset or in general for an unsupervised automatic computer vision system, the true class of items are unknown. Our data contains no class labeling of the nodes, but we know the angles at which the pictures were taken. If we assume that items (nodes or node-arc-node triplets) from pictures taken at approximately the same angle will be more similar than items from pictures taken at less similar angles, we can exploit this to quantify our results.

Lets introduce truck A with label 14005, items $A1$ and $A2$, truck B with label 14006, items $B1$ and $B2$, and truck C with label 15006, items $C1$ and $C2$. After clustering of the items we get the following results:

Class 1 : $A1$, $B1$, $C1$

Class 2 : $A2$, $B2$

Class 3 : $C2$

We count the number of shared classes for the items for each truck. Truck A and truck B got items in two of the same classes (Class 1 and Class 2) while Both Truck A and B share only one class with Truck C. We then say that A is best matched with B by two items, and then second to C with one item in this clustering.

Now we can use the labels to compute a step distance of the angles as a similarity measure of two trucks. The step distance is equal to the sum of the distance in both camera height angle and truck rotational angle e.g., $step(A, B) = step(14005, 14006) = |14 - 14| + |5 - 6| = 1$ and $step(A, C) = step(14005, 15006) = |14 - 15| + |5 - 6| = 2$. By step distance A is more similar to B than it is to C and this goes well with the clustering above.

To measure the success of a clustering we can run this procedure for every truck used to extract data for the clustering process and obtain a mean step distance to

Object Label	Matching Nodes
25016	10
3016	10
26015	9
26016	9
11015	8
...	
2018	2
27003	1
27010	1
27014	1
27017	1

Table 4: Matching for object 27016 with a total of 20 nodes.

best match (B_1). We have also in this thesis used step distance to best of 5 matches (B_5) and to best of 20 matches (B_{20}). Mean step distances from several clusterings can be compared to measure their relative success. We also randomized the clusters and calculated mean step distance as a control. It should be noted that this is a synthetic measurement of clustering success and may or may not correlate with a real case correctness of the clustering.

Lets examine an example from one of our runs. At the top of table 4 are the best five matchings, and at the bottom are five matchings with only one or two node matches. In this example $B_1 = step(27016, 25016) = 2$, $B_5 = step(27016, 26016) = 1$ and $B_{20} = step(27016, 26016) = 1$.

The method of counting cluster matchings used here can in itself be used as a object matcher algorithm. We ran some test showing that mean step distance increased as we went from best match to second best match, and increase even more when testing the third best match and so on. This indicates that the method would be at least a better object matcher then a random guess on our dataset.

4 Experiments and Results

In this section we describe the experiments we have done and the results we obtained. Runtime numbers have been given, but they are mainly illustrational. It should be noted that the two methods were run on separate hardware setups. All Snob experiments used Ubuntu Linux 3.2.0-41-generic 64-bit with an Intel Xeon E5440 CPU at 2.83GHz (shared server). All SOM experiments used Microsoft Windows 7 (6.1) Home Premium Edition 64-bit Service Pack 1 with an Intel Pentium SU4100 CPU at 1.3GHz.

4.1 Clustering Nodes

In this subsection we have clustered individual nodes. The nodes are areas found in the segmentation process. Ideally each area (node) represents a part of the truck (a wheel, cargo area, etc.), but the segmentation process is not flawless and sometimes a logic part of the truck might be segmented into several areas. In total we have 3785 nodes from 513 truck images, this averages 7.38 nodes per truck.

See tables 5 for Snob and table 6 for SOM clustering results of the full individual nodes dataset. Table 7 gives results after randomizing the content of clusters.

Clusters Found	B_1	B_5	B_{20}	Runtime (s)
93	7.61	2.82	1.32	35
98	8.1	2.94	1.31	30
102	7.82	2.89	1.31	26

Table 5: Snob all Nodes Clustering.

Map Size	Clusters Found	Training Runs	B_1	B_5	B_{20}	Runtime (s)
6^3	196	10	8.170	3.074	1.394	6.24
70^2	1408	10	9.238	3.674	1.710	138
12^2	138	100	8.226	2.589	1.228	24.85
12^2	138	10	7.955	2.908	1.288	4.1
10^2	99	10	8.068	2.836	1.318	2.15
8^2	62	10	8.873	2.982	1.287	1.41
7^3	290	10	8.183	2.846	1.333	7.12

Table 6: SOM all Nodes Clustering.

4.1.1 Nodes from Two trucks

In this experiment we clustered nodes from two trucks. We expected to see similar truck parts cluster together e.g., wheels with other wheels. Since we only used data from two images, the step distance method is not applicable on this clustering as our

Classes	B_1	B_5	B_{20}
98	16.54	8.17	3.79
93	16.67	7.98	3.78
100	16.91	7.93	3.66
138	16.62	8.37	3.91
62	15.44	7.58	3.78
290	16.19	8.44	4.63
196	16.71	7.85	4.27
5	22.00	15.70	10.79
50	15.60	8.16	3.88
500	16.13	9.09	4.40
1000	17.34	9.25	3.58
5000	16.09	8.52	7.9

Table 7: Random Nodes Clustering.

dataset is too small. We can instead use manual inspection since the datasets are only 13 and 14 nodes and will fit in print. We clustered nodes from two dissimilar trucks (1004 and 19006) and two similar trucks (19005 and 19006). Snob was not able to produce a meaningfull result for this small dataset, only one class was found. The results are therefor only from the SOM. See table 8 and table 9 for a visualization of the classes.

4.2 Clustering Node-Arc-Node Triplets

In this experiment we clustered Node-Arc-Node triplets. These triplets represent two parts of a truck and the relation between them, e.g., a wheel below the truck bed. We have in total 25922 triplets, each with 17 variables. Each truck averages 50.5 triplets.

In the previous section we clustered data from only two trucks, and used human inspection to check the results data for meaning. It would be interesting to do the same with triplets from two trucks, however as trucks have on average of over fifty triplets it would be infeasible to include the visualization of the results in this report.

See tables 10 and 11 for Snob and SOM clustering results of the full arc-node-arc triplet dataset. Table 12 gives results after randomizing the content of clusters.

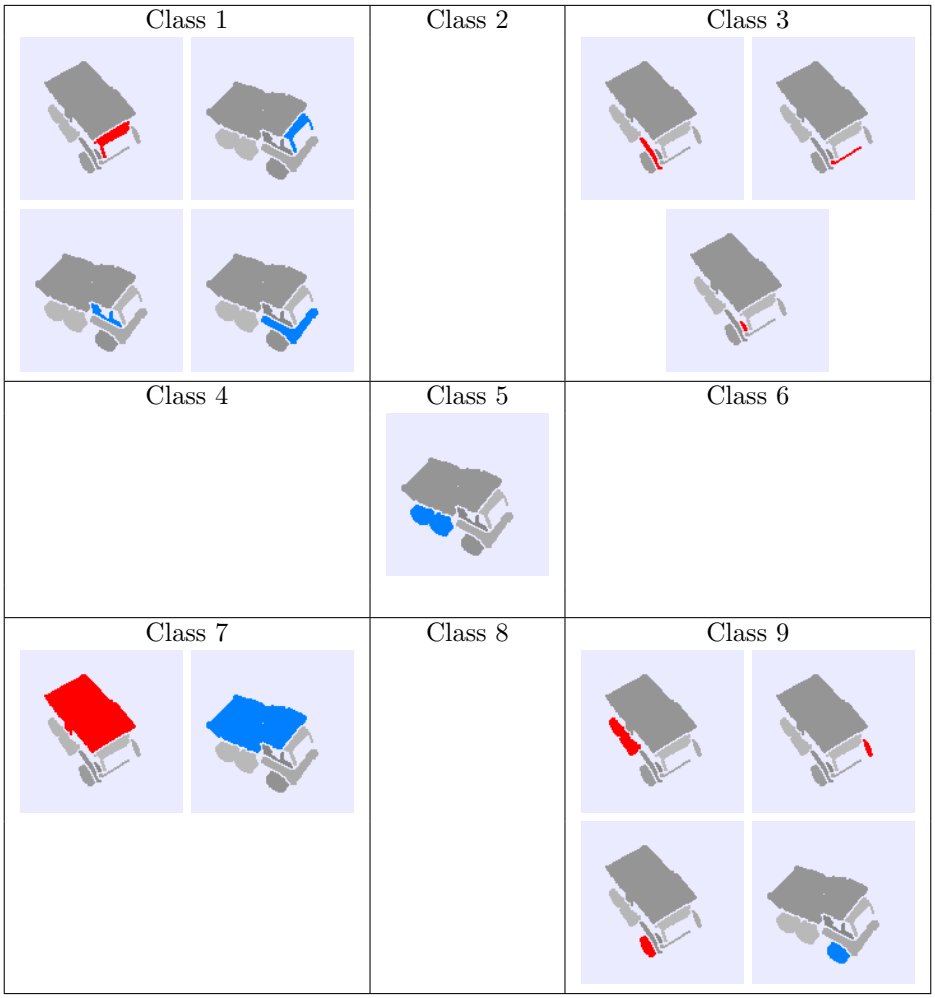


Table 8: SOM clustering of the nodes from two dissimilar trucks. Colored parts are the nodes, red for truck 1004 and blue for truck 19006.

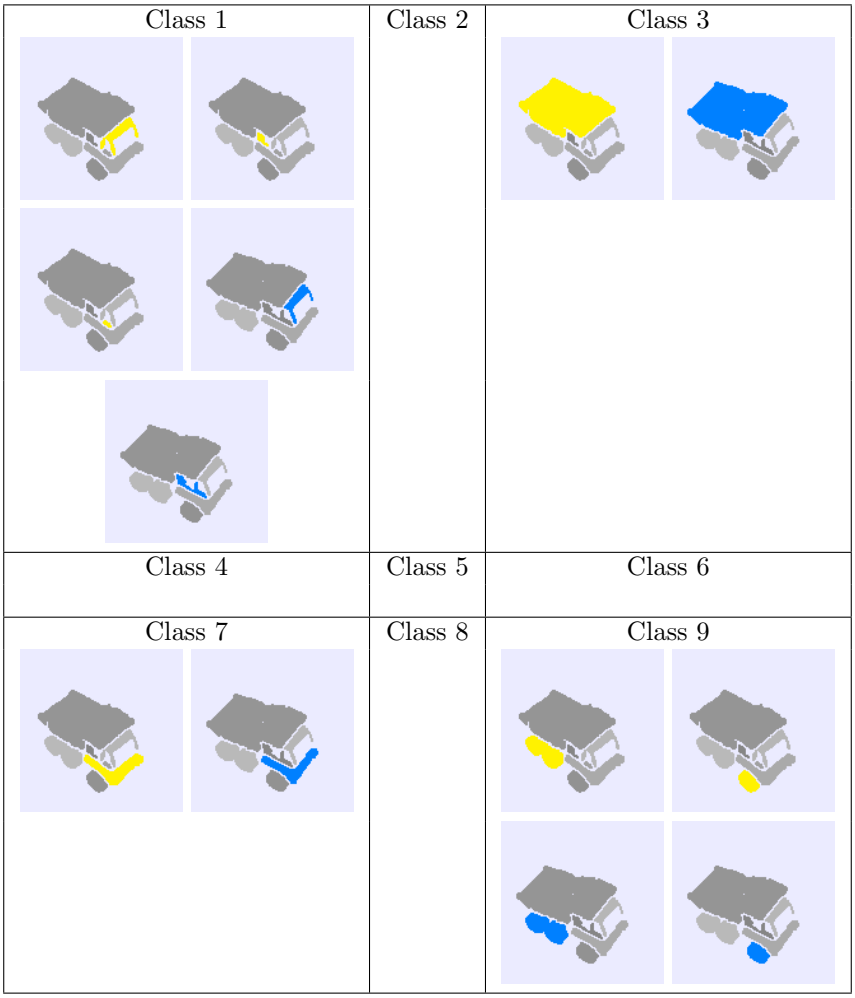


Table 9: SOM clustering of the nodes from two similar trucks. Colored parts are the nodes, yellow for truck 19005 and blue for truck 19006.

Clusters Found	B_1	B_5	B_{20}	Runtime (s)
181	7.79	2.68	1.25	5061
125	9.42	3.49	1.46	3147
333	7.66	2.5	1.15	3252
66	10.22	3.53	1.48	19632
96	8.48	3.01	1.44	6780

Table 10: Snob Triplet Clustering.

Map Size	Clusters Found	Training Runs	B_1	B_5	B_{20}	Runtime (s)
8^4	2555	10	6.573	2.207	1.086	1420
8^4	2518	50	6.402	2.279	1.086	8235
10^4	4399	10	6.630	2.090	1.105	3296
3^6	529	10	7.386	2.505	1.146	342
8^3	469	10	7.097	2.384	1.156	218
17^2	285	10	8.359	2.622	1.191	92
3^5	222	10	7.626	2.694	1.211	212
14^2	191	10	8.366	2.692	1.230	63
13^2	168	50	8.220	2.749	1.246	316
2^8	231	10	7.827	2.819	1.251	91
13^2	167	10	8.259	2.686	1.251	57
4^4	230	10	7.809	2.908	1.271	127
15^2	218	10	8.550	2.858	1.275	89
16^2	249	10	7.620	2.788	1.292	109
2^7	118	10	8.548	3.413	1.298	65
11^2	121	10	8.938	3.144	1.314	40
10^2	100	10	8.694	2.743	1.345	34
12^2	144	10	8.209	2.815	1.376	68
8^2	64	10	10.047	3.458	1.425	33
3^4	81	10	10.571	4.179	1.682	72
4^3	64	10	11.228	4.700	2.029	29
6^2	36	10	14.513	7.493	3.417	16
5^2	25	10	15.967	9.310	4.148	11
4^2	16	10	20.359	14.571	7.292	8
3^2	9	10	21.290	16.772	11.507	5

Table 11: SOM Triplet Clustering.

Classes	B_1	B_5	B_{20}
10	22.00	18.53	12.71
66	17.55	10.55	7.05
100	17.26	10.90	7.39
18 1	17.96	10.67	6.91
333	17.88	10.48	6.31
1000	17.62	9.91	5.41
10000	16.76	8.48	4.17
20000	16.04	8.53	4.12
40000	16.78	8.90	4.12
200000	15.93	8.92	7.72

Table 12: Random Triplet Clustering.

5 Discussion and Conclusion

In the previous section we have shown the clustering of structured image data to be successful. In this section we will discuss the various results and performance of the techniques used. We will also explore some possibilities for exploiting the clustering in NP graph matching problem. Som future work will be discussed and a conclusion will be given.

5.1 Individual Nodes

When clustering individual nodes we see consistent results from snob. About 100 clusters are found, the varying runtime is due to the different initialization done. An initial set of randomized clusters lets Snob arrive faster to the final results than when no initialization is done.

The SOM produces its best result when we use 144 nodes in the map. It should be noted that after the training has been done, it is not guaranteed that every node in the map will be a cluster representative for the training data used. In this instance we see only 138 effective clusters after classification even though the map has 144 nodes.

Using 100 training runs as opposed to only 10 improves the results for B_5 and B_{20}, but worsens B_1. The increased runtime of using more training runs seems not to be worth it in our experiments. Our results are generally good with only 10 training runs. We did some runs with a SOM too large for this dataset consisting of 4900 nodes, that is more nodes in the SOM than the number of training vectors used. This large of a map should theoretically lead to over fitting, and the results reflect this, the large map performed worst of all the SOMs.

For 100 nodes, which is about the amount of clusters snob found, the SOM produces a clustering giving very similar B_1, B_5 and B_{20}. Compared to random clustering both Snob and SOM produces better scores, this helps validate the results.

5.2 Two Trucks

5.2.1 Dissimilar Trucks

The two trucks clustered in table 8 have different segmentations. Truck 1004 (fig. 4a) have 8 segments, and truck 29006 (fig. 4b) have 6 segments. We see that because of the angle the front of truck 1004 have been segmented into several small areas. These small areas are exclusive to truck 1004 and so they are all clustered to class 3 without any nodes from the other truck.

In general this clustering would not be perfect by human standards, but we do spot a trend toward similar nodes intraclass. We see that the cargo areas are both in class 7 and most of the wheels go into class 9, except for 1 set of wheels that go in class 5. Even though one set of wheels missed their class, the topology conserving property of SOMs have likely helped keeping them not too far apart in the map from the other wheels.

5.2.2 Similar Trucks

Once we clustered two similar trucks as in table 9 we see here a perfect clustering. Class 1 got all cabin nodes, class 3 got all the cargo area nodes, class 7 got all the bumper nodes and class 9 got all wheels.

5.3 Node-Arc-Node Triplets

This was the largest clustering experiment done. As we bring two nodes and the relation into the clustering we have a quite more complex set of data then in the previous nodes only experiment. Because the relation is brought into the clustering we are in effect clustering small parts of the graphs. The clusters brought forward here are likely to be of help to the sorted table graph matcher as it is matching on arc-node-arc triplets.

Snob varies quite a lot in the number of clusters it produces here depending on the initialization, this is likely due to a quite complex data set. We see in table 10 that a run producing 333 clusters give the best average step distances. If we look at 12 we see that step distances are generally lower for randomized clusterings with a high number of classes. Therefore we should not jump to the conclusion that the 333 classes run is really a better clustering then the one that produced 66 classes. It is only by our average step distance synthetic measure that it performs better, but as our measure is correlated to number of classes, comparing clusterings of different class size via step distance might not be straight forward. The average step distance method seems mostly useful to compare clusterings of the same size.

For the triplet dataset a large variety of SOM configurations was used, we wanted to see how the SOM settings affected runtime and the results. We see in 11 that also here a larger map gives better results, but as discussed earlier it might be mostly due to the step distance method favoring a high number of classes. Using more training runs, as in 50 instead of 10, does not produce a much better clustering. It seems with our settings a low number of training iterations are sufficient.

Dimensionality seems not to matter much compared to the total number of nodes in total. A map of size 3^5 produced a slightly better results compared to one of size 2^8 even though it has less nodes. A map of two dimensions seems be be sufficient, but we don't have enough data to conclude on the perfect number of dimensions.

In general our SOM implementation seems to produce equally good clusterings at a much lower runtime compared to Snob. It should be noted Snob is an old program and possibly not optimized for speed. The SOM is viable for large datasets as it can be parallellized and shortcuts can be taken [25].

5.4 Application to Graph Matching

We have shown the clustering to give meaningful results, now we discuss some possible methods for improving a graph matcher system

There are two levels where the clustering can help the matching process. The first is at selecting appropriate graphs to be matched with the unknown graph. It is preferable that a good enough match is found in the least amount of time. Clustering can be done on all items, i.e., triplets or nodes in the database to find clusters. Items from the unknown graph can be assigned to the clusters, and the graphs that have items in the same classes can be prioritized for the matching process. This should in effect reduce the average runtime of the graph matching process as the least likely matches are put at the end of the matching queue.

The second level where clustering can be used is directly in the process of matching two graphs. In the sorted table graph matcher, we can manipulate the cost of each triplet matching. For instance if two triplets doesn't belong to the same cluster their cost of match may be increased. This method might work if clustering are done with Snob or another clustering method that does not explicitly use a similarity measure such as distance to do the clustering. Cost of match is also a similarity measure of the two triplets.This means that the cost in the match table and likeliness of the triplets to be in the same cluster are correlated. Therefore this straightforward method might not improved the matching process.

Instead of the similarity measure other metrics can be used. Tightness of a cluster could be tried to directly adjust the cost. Tightness is a vague term used here, but for our methods there are certain ways a tightness or homogeneity might be found. For Snob the tightness of a cluster could be measured using the message length needed to encode that cluster, a shorter message length means a tighter cluster. The std. dev. of the distributions used for a model is another way, low std. devs. means a tighter cluster. For the SOM a std. dev. of a clusters members variables could be used.

The topology preserving feature of the SOM can be exploited in a search process, as neighboring clusters are more similar to each other then those far apart. This can be taken advantage of if a good enough graph match is not found while being supported by the first level SOM clustering. The SOM clusters can be expanded to include its neighbors and therefore incrementally increasing search space. If a large number of nodes are used in a SOM the neighbors might in fact be quite similar, and in a clustering context they can themselves be clustered [26].

5.5 Conclusion and Future Work

We have in thesis demonstrated clustering of structural features in a computer vision context. Both self-organizing maps and minimum message length approaches produce meaningful clustering. The SOM seems more flexible as a magnitude of parameters can be adjusted, but this comes at the expense of more involvement needed for an expert. Methods for automating some of the parameters for the SOM has been proposed in the literature and may increase the self-sufficiency of such a system.

Especially interesting is the clustering of so called node-arc-node triplets that are in fact subgraphs of the larger graphs in use. The clusters formed can then be used in a graph matching system to reduce runtime required for an acceptable match, and methods for doing so have been proposed. Clustering in itself has been used before to classify images, but not in the context of helping a graph matcher that is a process of NP runtime.

Future work would be to implement the use of clusters to support the graph matching system. If the basic clustering techniques used here are able to improve upon the NP system, further investigation into better clustering methods for the domain at hand may also be investigated.

References

[1] Saraswathi Vishveshwara, K. V. Brinda, and N. Kannan. Protein structure: Insights from graph theory. *Journal of Theoretical and Computational Chemistry*, 01(01):187–211, 2002.

[2] J.R. Burch, E.M. Clarke, D.E. Long, K.L. McMillan, and D.L. Dill. Symbolic model checking for sequential circuit verification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(4):401–424, 1994.

[3] Bonnie Berger and Tom Leighton. Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete. *Journal of Computational Biology*, 5(1):27–40, 1998.

[4] E. Bengoetxea. *Inexact Graph Matching Using Estimation of Distribution Algorithms*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, Dec 2002.

[5] Guido Del Vescovo and Antonello Rizzi. Automatic classification of graphs by symbolic histograms. In *Granular Computing, 2007. GRC 2007. IEEE International Conference on*, pages 410–410. IEEE, 2007.

[6] A Rizzi and G Del Vescovo. Automatic image classification by a granular computing approach. In *Machine Learning for Signal Processing, 2006. Proceedings of the 2006 16th IEEE Signal Processing Society Workshop on*, pages 33–38. IEEE, 2006.

[7] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[8] Simon Thoresen. *An efficient solution to inexact graph matching with application to computer vision*. PhD thesis, Norwegian University of Science and Technology, 2007.

[9] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[10] Richard E Blake and Algimantas Juozapavicius. Convergent matching for model-based computer vision. *Pattern recognition*, 36(2):527–534, 2003.

[11] Richard E Blake and Peter Boros. The extraction of structural features for use in computer vision. In *Proceedings of the Second Asian Conference on Computer Vision, Singapore*, 1995.

[12] RE Blake. Graph matching with a sorted table: a suboptimal method that is effective and allows problem partitioning. In *Proceedings of ICARCV*, volume 90, pages 956–960, 1990.

[13] Kuang-Chiung Chang, CHENG WEN, Ming-Feng Yeh, and Ren-Guey Lee. A comparison of similarity measures for clustering of qrs complexes. *Biomedical Engineering: Applications, Basis and Communications*, 17(06):324–331, 2005.

[14] GE Farr and CS Wallace. The complexity of strict minimum message length inference. *The Computer Journal*, 45(3):285–292, 2002.

[15] Christopher S Wallace. *Statistical and inductive inference by minimum message length*. Springer Science+ Business Media, 2005.

[16] Rohan A. Baxter. Minimum message length inference: Theory and applications. Technical report, Monash University, Australia, 1996.

[17] C. S. Wallace and D. M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968.

[18] Chris S. Wallace and David L. Dowe. Mml clustering of multi-state, poisson, von mises circular and gaussian distributions. *Statistics and Computing*, 10(1):73–83, 2000.

[19] M Afzal Upal and E Neufeld. Comparison of unsupervised classifiers. In *Proceedings of the First International Conference on Information, Statistics and Induction in Science*, pages 342–353. Citeseer, 1996.

[20] Monash university data mining centre software page - snob. http://www.datamining.monash.edu.au/software/snob/. Accessed May 2013.

[21] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[22] Teuvo Kohonen, Erkki Oja, Olli Simula, Ari Visa, and Jari Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–1384, 1996.

[23] Fernando Bação, Victor Lobo, and Marco Painho. Self-organizing maps as substitutes for k-means clustering. In *Computational Science–ICCS 2005*, pages 476–483. Springer, 2005.

[24] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[25] Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojarvi, Jukka Honkela, Vesa Paatero, and Antti Saarela. Self organization of a massive document collection. *Neural Networks, IEEE Transactions on*, 11(3):574–585, 2000.

[26] Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *Neural Networks, IEEE Transactions on*, 11(3):586–600, 2000.

[27] Serhiy Kosinov and Terry Caelli. Inexact multisubgraph matching using graph eigenspace and clustering models. In Terry Caelli, Adnan Amin, RobertP.W. Duin, Dick Ridder, and Mohamed Kamel, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 133–142. Springer Berlin Heidelberg, 2002.

[28] Miguel Angel Lozano and Francisco Escolano. Graph matching and clustering using kernel attributes. *Neurocomputing*, 2013.