Classification of 3D Surface Data Using the Concept of Vertex Unique Labelled Subgraphs

Wen Yu*, Frans Coenen *, Michele Zito * and Kwankamon Dittakan *
*Department of Computer Science
The University of Liverpool
Ashton Building, Ashton Street, Liverpool, L69 3BX, UK
Email: {yuwen,coenen,michele,dittakan}@liverpool.ac.uk

***Abstract*—An overview is presented on the use of the concept of Vertex Unique Labelled Subgraph (VULS) mining for the use of localised classification of regions in 3D surfaces represented in terms of grid graphs. A VULS is a sub graph within some larger graph G that has a unique ("one-of") vertex labelling associated with it. Given a 3D surface represented as a grid graph, we can identify a number of different forms of VULS that may be discovered: (i) all, (ii) minimal, (iii) frequent and (iv) frequent minimal. Algorithms for discovering (mining) these are presented in the paper. The paper also presents the Backward Match Voting (BMV) algorithm for predicting (classifying) vertex labels associated with an "unseen" graph using a given collection of VULS. The operation of the VULS mining algorithms, and the BMV algorithm, is fully described and evaluated. The evaluation is conducted using satellite image data where the ground surface is represented as a 3D surface with the z dimension describing grey scale value. The idea is to predict vertex labels describing ground type. A statistical analysis of the results, using the Friedman test, is also presented so as to demonstrate the statistical significance of the VULS based 3D surface regional classification idea. The results indicate that the VULS concept is well suited to the task of 3D surface regional classification.**

***Keywords*-Data Mining; Graph Mining; Vertex Unique Labelled Subgraph Mining; Vertex Classification;**

I. Introduction

Three Dimensional (3D) surface data occurs in the context of many environments. Obvious examples are applications that use map data such as geological studies. Other applications include manufacturing processes where 3D parts are produced; one example is sheet metal forming ([1]). There are many other domains where we can represent the data in a 3D x-y-z format such that 3D surfaces can be defined. For example image data can be defined in terms of a surface where the z coordinate represents (say) intensity or gray scale value. Given a 3D surface we may wish to classify this in a global manner, surface s belongs to class c; or we might wish to classify the surface in a local manner, region r in surface s belongs to class c. For example, with respect to the latter we might wish to say that a certain region within some given map data features a particular from geology, or data at a particular locality on a sheet metal surface has some specific distortion associated with it, or data in a certain area within an image depicts a particular object. In this paper we propose a mechanism for conducting such local 3D surface classification using Vertex Unique Labelled Subgraphs (VULS). The concept of VULS Mining (VULSM) was first proposed by the authors in [2] and further developed in [3], [4]. A VULS is a subgraph with particular structure and edge labelling that has a unique vertex labelling associated with it. The idea is to represent the 3D surface of interest in terms of a grid graph with a grid size of d. Each vertex in the graph is then defined by an x-y coordinate and has associated with it a z value (vertices thus represent regions within a surface). Each vertex also has some class label c assiciated with it, taken from a set of classes $C = \{c_1, c_2, \dots\}$. Given a pre-labelled training set, VULS can be identified using a VULS mining process. The resulting set of identified VULS can then be used to label vertices in previously unseen grid graphs.

In the context of VULS based classification there are two challenges. The first, in common with other forms of graph mining, is that VULS mining is computationally expensive. The second is that the identified set of VULS U should be as comprehensive as possible. We express this in terms of *coverage*, which is measured as follows:

$$coverage = \frac{|V'|}{|V|} \tag{1}$$

where V' is the set of vertices contained in U and V is the complete set of vertices contained in some pre labelled input graph G. To achieve good coverage a comprehensive training set is required. In the context of computational complexity, instead of identifying all VULS, we can also attempt to identify some appropriately descriptive (in terms of coverage) subset of the complete set of VULS. To this end we can identify a number of different forms of VULS: (i) minimal VULS, (ii) frequent VULS and (iii) frequent minimal VULS. A minimal VULS is a VULS ϕ such that none of its subgraphs are VULS. A frequent VULS is a VULS ϕ whose occurrence count within an input graph G is greater than some pre-specfied threshold σ. A frequent minimal VULS is then a VULS ϕ which is both frequent and minimal.

To illustrate the operation of the proposed approach we consider the classification of regions within satellite imagery for the purpose of land usage analysis [5], [6], [7], [8].

The rest of this paper is organised as follows. An overview of the proposed VULS mining and classification process is presented in Section II. The algorithms for mining VULS, minimal VULS, frequent VULS and frequent minimal VULS are described in section III. The Backward-Match-Voting algorithm (for predicting vertex labels in previously unseen data) is then presented in Section IV. An experimental analysis of the proposed approach, in the context of satellite images, is then given in section V. Section VI presents some conclusions and summarises the work and main findings.

II. System Overview

An overview of the proposed VULS based 3D surface classifier generation process is presented in this section. A block diagram is given in Figure 1. The input is a set of 3D surfaces (grayscale satellite images with respect to the work described in this paper). These images are then translated into grid graph format; in the case of our satellite images the z coordinate describes the average grayscale value with respect to each grid square. Once the satellite images have been translated into a grid-graph format VULS mining can be applied. Recall that we can mine either: (i) the complete set of VULS, (ii) minimal VULS, (iii) frequent VULS or (iv) frequent minimal VULS. The VULS mining process is described in further detail in Section III below. To apply (or test) the resulting set of VULS, in the context of classification (vertex label prediction), we match the mined VULS to subgraphs in new data so that the labels associated with the VULS can be used to label the new graph. This is achieved using what we have termed the Backward Match Voting (BMV) algorithm which is thus described in further detail in Section IV.

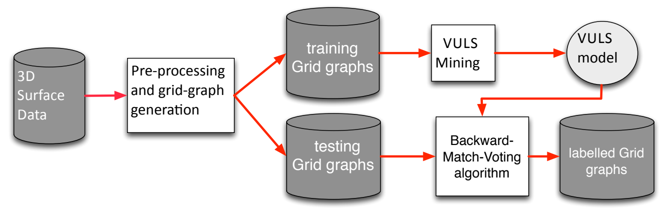

Fig. 1: The VULS generation and application/testing process

III. VULS Mining

The mining of the complete set of VULS encompasses traversing the entire search space starting with $k = 1$ edge subgraphs and continuing until some user-specified Maximum number of edges (Max) is reached. Note that in the context of classification very large VULS will be of little use as they will be "overfitted" to the training set and therefore unlikely to appear in any unseen data. Note also that if a k edge subgraph is not a VULS this does not necessarily mean that its $k+1$ edge super graphs will also not be VULS (and vice versa). Indeed as k approaches max the likelihood of finding VULS increases. Thus the process for finding all VULS is exhaustive, involving exponential time complexity as the size of the input graph increases.

The high level pseudo code for the VULSM algorithm is presented in Algorithm 1. The input is a labelled graph G and a value max for the maximum size of an identified VULS. Note that the input graph G is encoded using minimal Depth First Search (DFS) lexicographical ordering, a canonical form also used with respect to the well known gSpan algorithm [9].

Algorithm 1 : VULSM algorithm for mining All VULS

```
1: Input:
2: $G$ = The input graph
3: $max$ = The maximum size of a VULS
4: Output:
5: $U$ = The set of all VULS contained in $G$

6: $U = \emptyset$
7: $G_1$ = the set of 1 edge (candidate) subgraphs contained in $G$

8: procedure $main(G, max)$
9:     $k = 1$
10:    while $(k < max)$ do
11:        $U = U \cup idVULS(G_k)$ (Algorithm 2)
12:        $coverage$ = Coverage calculated using Equation 1
13:        if $coverage \equiv 100\%$ then
14:            exit
15:        end if
16:        $k = k + 1$
17:        $G_k$ = candidateGeneration$(G_1, G_{k-1}, k)$ (Algorithm 3)
18:    end while
19: end procedure
```

The output is the set of identified VULS U. Recall that we limit the size of the searched-for VULS using the parameter max; if we do not do this the entire input graph may ultimately be identified as a VULS which, in the context of the intended applications of VULS (prediction), will not be very useful. At the start of the VULSM procedure, the set U will be empty (line 6). Note also that $max \leq |E_G|$ as otherwise we will attempt to generate candidate subgraphs that are bigger than the input graph. We proceed in a breadth first manner commencing with one edge candidate VULS, then two edge candidate VULS, and continuing until max edge candidate VULS are arrived at. On each iteration we first identify the k edge VULS contained in the set of candidate VULS G_k (line 11), using the $idVULS$ procedure (Algorithm 2), and include them in the set U. We then check the coverage (lines 12 to 15) using Equation 1. Note that coverage checking requires isomorphism checking which, for any reasonably sized input graph G, is computationally expensive. Although not shown in Algorithm 1, as the algorithm progresses, we mark vertices in the input graph that are covered so that as the algorithm progresses we only need to check the vertices that are uncovered so far. If the entire input graph G is covered (coverage = 100%) the procedure stops. Otherwise we continue and generate the $k+1$ edge candidate VULS (line 17) using the $candidateGeneration$ procedure (Algorithm 3), and so on.

Algorithm 2 shows the $idVULS$ procedure (and the associated $isVULS$ procedure). The $idVULS$ procedure takes as input a set of k edge candidate VULS (G_k) and returns a set

Algorithm 2 : VULS identification procedure

```
1: procedure $idVULS(G_k)$
2:     $U' = \emptyset$
3:     for all $g \in G_k$ do
4:         if $isVULS(g)$ then
5:             $U' = U' \cup g$
6:         end if
7:     end for
8:     return $U'$
9: end procedure

10: procedure $isVULS(g)$
11:     $isaVULS$ = true
12:     $S$ = List of lists of labels that may be assigned to
        each vertex in $g$
13:     for all $s \in S$ do
14:         if $|s| \neq 1$ then
15:             $isaVULS = false$
16:             break
17:         end if
18:     end for
19:     return $isaVULS$
20: end procedure
```

(U') of discovered k edge VULS (if any). Each candidate subgraph g in G_k is processed in turn (line 3). Note that each subgraph g in G_k has the form $\langle V_g, E_g, L_{E_g} \rangle$ (no vertex label set). If a candidate graph g is identified as a VULS it is appended to the set U' (line 5). To determine whether a candidate VULS g is a VULS the $isVULS$ procedure is called. The $isVULS$ procedure is given in the second part of Algorithm 2 (lines 10 to 20). In line 12 a list of sets of possible labels for each vertex in the current candidate VULS g is determined ($S = \{s_1, s_2, \ldots s_{|v_g|}\}$); note that S is generated with respect to the original input graph G and that there is a one to one correspondence between the sets in S and the vertices in V_g. If the size of any of the sets $s_i \in S$ is greater than 1 (line 14) then g is not a VULS and the procedure returns *false*. Otherwise the procedure returns *true*.

Algorithm 3 : Candidate Generation procedure

```
1: procedure $candidateGeneration(G_1, G_{k-1}, k)$
2:     $G_k = \emptyset$
3:     for each $g \in G_{k-1}$ do
4:         for each edge $e \in G_1$ do
5:             $G_{temp}$ = Set of k edge subgraphs produced by
               extending $g$ with $e$ using a modified form of "right most
               extension"
6:             $G_k = G_k \cup G_{temp}$
7:         end for
8:     end for
9:     Return $G_k$
10: end procedure
```

Algorithm 3 gives the $candidateGeneration$ procedure. This is similar to that found in frequent subgraph mining algorithms such as gSpan, but using a modified form of "right most extension" (using unlabelled vertices instead of labelled vertices). The input to the procedure is the set of one edge subgraphs G_1 identified at the beginning of the process (Algorithm 1, line 7), the set of candidate VULS from the previous iteration (G_{k-1}) and the current candidate VULS of size k. Then for each g in G_{k-1}, and each one edge subgraph e in G_i, we compute the k edge subgraphs by applying our modified "right most extension" principle. In other words we grow an existing candidate graph g by adding edges to its right most nodes.

Algorithm 1 can be easily modified to mine: (i) minimal VULS, (ii) frequent VULS, and (iii) frequent minimal VULS. For minimal VULS mining, on each iteration, we only extend non-minimal VULS instead of all subgraphs to form further candidate VULS. For (frequent VULS mining, on each iteration, we only extend frequent subgraphs instead of all subgraphs to form further candidate VULS ready for the next stage processing (if a VULS ϕ is infrequent none of its super graphs will be frequent). As noted above a frequent VULS is one whose occurrence count exceeds some threshold value σ. In the case of transaction graph mining, we can express σ in terms of a proportion of the number of transaction graphs under consideration (a subgraph is frequent if it appears in $x\%$ of the available set of transaction graphs). In single graph mining this is not so straight forward, σ can of course be predefined in terms of a fixed value but this does not take account of the overall graph size. With respect to the work described in this paper the value σ is calculating dynamically as the average of the sum of the *support* values for the complete set of candidate VULS identified on each iteration as shown in Equation 2 where $g_i \in G_k$ and *support* is a function that returns the occurrence count (support value) of g_i in G.

$$\sigma = \frac{\sum_{i=1}^{i=|G_k|} support(g_i)}{|G_k|} \tag{2}$$

For frequent minimal VULS mining we simply combine the minimal and frequent VULS mining procedures.

IV. Classification Using VULS

The proposed classification mechanism, founded on the VULS concept, where by the vertices in a new unseen graph G_{new} are labelled, is presented in this section. Essentially the process is one of matching subgraphs in a G_{new} with subgraphs in the set U of previously identified VULS. An issue with this approach is that vertices in G_{new} may be matched to more than one vertex in the set U of VULS with different vertex labels. In this case the conflicting label issue needs to be resolved. We can identify a potential number of mechanisms for doing this but we have opted for a voting mechanism. A variety of voting mechanisms also exist, including: (i) majority voting and (ii) weighted voting. We adopted the first as the nature of the most appropriate weighting mechanisms to use was difficult to determine (as indicated by some initial experiments not reported here). We refer to the proposed approach as the Backward Match Voting (BMV) algorithm because: (i) we work "backwards" from the maximum value for k to $k = 1$ (because this is more efficient as potentially larger numbers of vertices in G_{new} will be covered early on in the process), (ii) we "match" graph structures and edge labelings in U with graph structures and edge labelling in G_{new} so as to label the vertices in G_{new} using the labelling from U, and (iii) where more than one vertex label is assigned to a vertex in G_{new} we use a majority "voting" scheme.

The Backward Match Voting algorithm is presented in Algorithm 4. The algorithm comprises a single main procedure ($main$) and a sub-procedure. The algorithm takes as input a collected set U of VULS and a new graph G_{new} which has known edge labels but unknown vertex labels. The algorithm also utilises the parameter max set to the same value as that used to generate the VULS in U so that the algorithm does not try to find matches beyond the maximum size of the VULS presented as part of the input. The output is a vertex labelled graph G_{new}. The algorithm, as noted previously, starts with VULS of size max and iteratively proceeds with VULS of decreasing size until one edge VULS are reached or 100% coverage of the vertices in G_{new} is obtained, whichever happens first. On each iteration a set, L, is generated (line 13) to hold vertex labels extracted from relevant VULS in U. Note that each set in L corresponds to a vertex in G_{new} that does not yet have a label associated with it. We then, on each iteration, process the VULS of size k and populate L. We then process each set L_i in L. If the number of labels in a given set L_i is greater than one (Line 18) the voting mechanism is invoked and the most popular label assigned to the corresponding vertex V_i in G_{new}. If a set L_i has only one label (line 21) this is the label assigned to the corresponding vertex V_i in G_{new}. In some case a set L_i will be empty Whatever the case once L has been processed the coverage is calculate, if coverage is equivalent to 100% the process stops (line 28), otherwise we continue with the $k-1$ subsets. At the end of the process any remaining vertex in G_{new} that does not yet have a label attached to it is assigned a default label (lines 32 to 37).

V. Experiments and Performance Study

This section describes the evaluation of the proposed VULS based classifier mechanism in terms of effectiveness and efficiency. Because of the novelty of the proposed VULS concept there are no alternative algorithms that can be used for comparison purposes. However, it was possible to compare the operation of the proposed mechanism in the context of grid size d ($\{8, 16, 32\}$ pixels) and varying vertex degree values (4 or 8). It was also possible to evaluate classification effectiveness using the four different forms of proposed VULS: all , minimal, frequent and frequent minimal. Thus 24 ($3 \times 2 \times 4$) VULS mining configurations were considered. A fixed max parameter setting of 4 was used because earlier experiments, not reported in this paper, had demonstrated that this produced the best results. The evaluation was conducted by considering

Algorithm 4 : Backward Match Voting (BMV) algorithm to predict vertex labels in a vertex-unlabelled graph

```
1: Input:
2: $U$ = Set of identified VULS
3: $G_{new}$ = Edge labelled previously unseen input graph (with unlabelled vertices)
4: $Max$ = the Maximum VULS size (consistent with training procedure)
5: Output:
6: Graph $G_{new}$ with labelled vertices

7: defaultVertexLabel = A default vertex label
8: coverage=0
9: $V$ = Set of vertices in $G_{new}$ ($\{v_1, v_2, \ldots, v_{|G_{new}|}\}$)
10: $k = max$

11: procedure $main$($G_{new}$, $max$, $U$)
12:     while ($k >= 1$) do
13:         $L$ = Set of empty sets related to vertices in $G_{new}$ that do not already have a vertex label
14:         for all $g \in U$ where $|g| \equiv k$ do
15:             $L_{V_c}$ = Updated set of labels in $L_{V_c}$ where vertices in $g$ match $G_{new}$
16:         end for
17:         for all $L_i \in L$ do
18:             if $|L_i| > 1$ then
19:                 $v_i.label$ = most frequent label in $L_i$
20:             else
21:                 if $|L_i| \equiv 1$ then
22:                     $v_i.label$ = label in $L_i$
23:                 end if
24:             end if
25:         end for
26:         $coverage$ = compute coverage using Equation 1 with respect to $G_{new}$
27:         if $coverage \equiv 100\%$ then
28:             exit
29:         end if
30:         $k = k - 1$
31:     end while
32:     if $coverage \not\equiv 100\%$ then
33:         for all $v_i \in G_{new}$ do
34:             if $v_i.label \equiv null$ then
35:                 $v_i.label$ = DefaultVertexLabel
36:             end if
37:         end for
38:     end if
39: end procedure
```

a specific case study directed at a rural area of Ethiopia which is described in Sub-section V-A below. Ten fold Cross-Validation (TCV) was used throughout. The recorded metrics were: (i) Area Under the Receiver Operating Characteristic Curve (AUC), (ii) coverage and (iii) run time. To determine the statistical significance of the AUC results the Friedman test combined with the Nemenyi's post-hoc test was applied. A variant of $Dem\check{s}ar$s significance diagrams was used to visualise the statistical significance results obtained. Note that the Friedman statistic was computed using the approach presented in [10]. Image preprocessing and preparation was conducted using the MATLAB (matrix laboratory) workbench[2]. The presented VULSM algorithms were implemented using the JAVA programming language. All experiments were conducted using a 2.7 GHz Intel Core i5 with 4 GB 1333 MHz DDR3 memory, running OS X 10.8.1 (12B19).

A. Case Study Application Domain

For evaluation purposes a set of satellite images describing part of the Ethiopian hinterland was used. Image grey scale pixels values were interpreted as z values and consequently each image represented a 3D surface. Clearly other forms of imagery than satellite imagery could equally well have been used; or indeed other forms of 3D surface. The satellite images were extracted from Google Earth[1]. An example image is given in Figure 2. In total 10 images were downloaded and hand annotated with land usage labels describing regions with in each satellite. For the evaluation presented in this short paper the label (class) set was $C = \{HouseHold, NonHouseHold\}$ (thus $|C| = |L_V| = 2$). To identify regions a segmentation process was adopted. The satellite images were converted into a grid graph, as described earlier in this paper, with each node representing a segmented region. If a node encompassed more than one region the majority region was selected. The vertex labelling is inspired by the work presented in [11] which used the same data set for the purpose of population estimation. in Edges were labelled according to the grey scale difference between the end vertices, and were directed from low grayscale value to high. The edge labels were discretised using equal range discretisation. Different numbers of ranges were used, $\{2, 3, 4, 5, 6, 7, 8, 9\}$, for the edge label discretisation.

B. AUC results

Table I shows the AUC results obtained. The 24 VULS models are represented by the columns. The highest AUC value obtained in each column is highlighted in grey. From the table it can be seen that the best average AUC result was obtained when using a grid size of $d = 8$ and grid graphs with degree 8 (an average AUC of 0.760). Thus successfully distinguishing "HouseHold" nodes (grid squares) from "Field" nodes (grid squares). Regardless of VULS model (all VULS, minimal VULS, frequent VULS, or frequent minimal VULS), as the grid size d decreased, the recorded AUC values tended

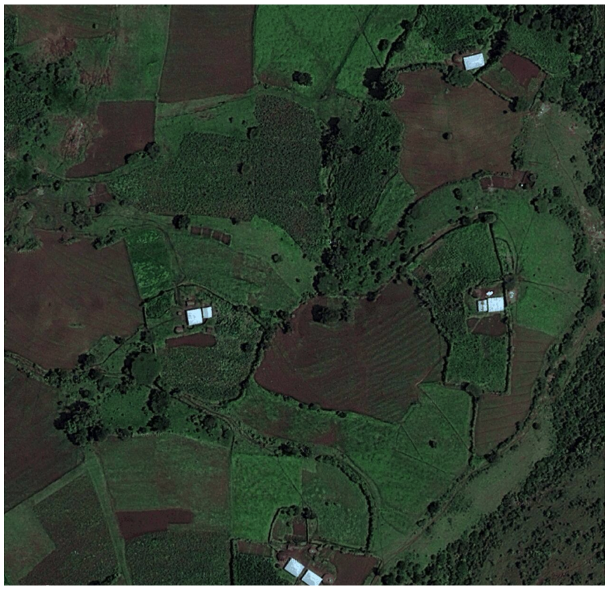
Fig. 2: Example satellite image

to increase. This was because as the grid size decreases the number of grid squares increases, thus the number of vertices increases and consequently the grayscale intensity difference (represented by edge labels) between end vertices becomes substantial, therefore labelling of vertices associated with VULS is more straight foward and the final performance tends to be better.

C. Coverage results

Tables II shows the coverage results obtained. As expected the best coverage (100%) is achieved when using all VULS or minimal VULS mining. This is an interesting result indicating that, at least in the case of this reported experiment, that the identified set of minimal VULS incorporated all those VULS from the complete set of VULS required to produce good coverage. Using a grid size $d = 32$ and grid graphs with degree 4 produced the best overall performance in terms of average coverage (100%). From the table it can also be seen that in the case of frequent VULS and frequent minimal VULS the coverage was often 0%. This is because fewer VULS are identified using these models, consequently fewer vertices are covered, thus lower coverage results are produce. Although in some cases, when using frequent VULS, coverage does on occasion reach 100% it is conjectured that if a lower threshold value σ were used more frequent VULS might be identified and consequently better coverage (and possibly AUC) results produced. However, this is an item for further investigation and future work.

D. Run time results

Tables III shows the recorded run time (seconds) results obtained. For convenience the recorded run time is made up of both the training and testing phases, thus the application of both the appropriate VULSM algorithm and the Backward

[2]http://www.mathworks.com
[1]http://www.google.co.uk/intl/en_uk/earth/index.html

TABLE I: AUC results

Graph	VULS						Minimal VULS						FVULS						Frequent Minimal VULS					
	$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$	
	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8
SIE2V2	0.500	0.500	0.500	0.500	0.752	0.756	0.567	0.545	0.778	0.75	0.661	0.452	0.566	0.507	0.608	0.608	0.71	0.500	0.517	0.519	0.608	0.608	0.500	0.500
SIE3V2	0.500	0.500	0.500	0.557	0.691	0.805	0.54	0.616	0.851	0.863	0.719	0.795	0.508	0.548	0.696	0.508	0.747	0.687	0.692	0.615	0.696	0.697	0.500	0.500
SIE4V2	0.500	0.500	0.508	0.608	0.764	0.814	0.499	0.541	0.668	0.53	0.815	0.865	0.497	0.569	0.608	0.608	0.674	0.629	0.565	0.568	0.607	0.608	0.674	0.579
SIE5V2	0.500	0.500	0.558	0.608	0.748	0.777	0.508	0.545	0.502	0.556	0.812	0.893	0.569	0.549	0.664	0.608	0.686	0.628	0.646	0.565	0.664	0.665	0.686	0.578
SIE6V2	0.500	0.500	0.558	0.608	0.731	0.764	0.547	0.613	0.591	0.707	0.809	0.788	0.498	0.55	0.608	0.608	0.735	0.579	0.56	0.615	0.608	0.697	0.735	0.686
SIE7V2	0.499	0.500	0.608	0.608	0.752	0.748	0.55	0.549	0.582	0.733	0.691	0.681	0.497	0.560	0.608	0.615	0.666	0.554	0.646	0.567	0.664	0.615	0.666	0.566
SIE8V2	0.500	0.500	0.508	0.687	0.744	0.702	0.56	0.512	0.563	0.739	0.743	0.75	0.548	0.57	0.665	0.608	0.674	0.629	0.57	0.564	0.672	0.673	0.674	0.628
SIE9V2	0.500	0.500	0.508	0.565	0.768	0.711	0.55	0.663	0.674	0.682	0.739	0.726	0.545	0.576	0.658	0.615	0.678	0.628	0.58	0.560	0.664	0.697	0.678	0.628
average	0.500	0.500	0.531	0.593	0.7438	0.760	0.540	0.573	0.651	0.695	0.749	0.7438	0.529	0.554	0.639	0.597	0.696	0.604	0.597	0.572	0.648	0.658	0.639	0.583

TABLE II: Coverage resuts

Graph	VULS						MinVULS						FVULS						FminVULS					
	$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$	
	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8
SIE2V2	100	100	100	100	99.883	90.322	100	100	100	100	99.834	90.488	98.75	97.188	99.766	99.844	97.891	0	89.063	89.531	99.766	99.844	0	0
SIE3V2	100	100	100	100	99.98	100	100	100	100	100	99.98	100	99.375	99.531	99.453	99.922	99.502	99.785	96.719	98.438	99.297	99.531	0	0
SIE4V2	100	100	100	100	100	100	100	100	100	100	100	100	100	99.375	99.727	99.844	99.736	99.844	98.594	99.063	99.609	99.844	99.736	89.863
SIE5V2	100	100	100	100	100	100	100	100	100	100	100	100	99.375	99.844	99.57	99.727	99.6	99.766	97.656	98.594	99.57	99.648	99.6	89.785
SIE6V2	100	100	100	100	100	100	100	100	100	100	100	100	100	99.844	99.766	99.844	99.316	99.922	98.438	98.438	99.727	99.531	99.316	99.619
SIE7V2	100	100	100	100	100	100	100	100	100	100	100	100	99.844	99.688	99.727	99.766	99.814	99.941	97.656	98.906	99.492	99.766	99.814	99.834
SIE8V2	100	100	100	100	100	100	100	100	100	100	100	100	100	99.844	99.492	99.805	99.736	99.844	98.281	98.438	99.453	99.609	99.736	99.824
SIE9V2	100	100	100	100	100	100	100	100	100	100	100	100	99.688	99.688	99.727	99.766	99.658	99.805	98.125	99.688	99.57	99.531	99.658	99.785
average	100	100	100	100	99.983	98.790	100	100	100	100	99.977	98.811	99.629	99.375	99.654	99.815	99.407	87.363	96.817	97.637	99.561	99.663	74.733	72.339

TABLE III: run time (seconds) where $|L_V|$=2 for graphs

Graph	VULS						MinVULS						FVULS						FminVULS					
	$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$		$d = 32$		$d = 16$		$d = 8$	
	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8	Deg4	Deg8
SIE2V2	1.76	7.49	4.45	48.36	14.03	50.66	1.14	2.17	1.86	3.56	6.15	39.94	1.89	1.52	9.65	4641.82	4.95	12.82	1.08	0.98	1.17	3.09	1.8	10.13
SIE3V2	1.71	19.5	5.89	20.27	22.8	111.99	0.93	1.58	1.86	13	7.23	32.74	1.45	2.29	6.41	3057.38	49.33	27231.6	1.22	1.02	1.41	2.41	2.2	10.13
SIE4V2	1.88	9.71	5.74	92.86	29.22	147.92	1.22	2.15	3.01	17.8	7.47	52.49	1.76	1.23	5.08	1419.99	46.32	26119.6	0.91	1.2	1.12	2	2.56	9.89
SIE5V2	1.93	6.36	8.71	63.09	45.66	233.3	1	2.08	2.08	8.45	6.72	204.46	2.4	2.47	4.07	16.66	42.73	23364.87	0.96	0.94	1.25	1.6	2.94	8.03
SIE6V2	1.71	4.99	6.58	93.57	35.36	278.18	1.17	1.79	2.73	4.16	6.83	128.76	1.95	4.38	2.74	2.99	38.03	21909.67	0.81	1.41	1.17	1.49	2.43	5.29
SIE7V2	1.5	4.76	4.89	43.17	32.15	549.85	1.17	1.94	2.93	7.06	7	108.58	2.35	1.66	2.68	3.72	26.81	6096.72	1	1.19	1.24	1.32	2.48	3.54
SIE8V2	1.46	7.27	5.49	46.11	33.06	170.14	1.03	1.88	2.13	10.39	5.8	126.57	2.47	6.62	3.08	6.21	20.29	596.78	0.85	1.52	1.26	1.61	2.31	2.88
SIE9V2	1.55	6.9	3.96	36.72	28.03	248.75	1.11	1.81	2.31	6.71	4.75	215.4	2.48	8.81	4.65	12.57	18.25	541.12	0.96	1.12	1.38	1.66	2.31	2.98
average	1.688	8.373	5.714	55.519	30.039	223.849	1.096	1.925	2.364	8.891	6.494	113.618	2.094	3.623	4.795	1145.168	30.839	13234.148	0.974	1.173	1.250	1.898	2.379	6.609

Match Voting (BMV) algorithm. Inspection of the table indicates that, understandably, using minimal VULS, frequent VULS and frequent minimal VULS requires less average run time than when using all VULS. Minimal VULSM and frequent VULSM is significantly more efficient than when mining all VULS. General speaking, the fastest approach was when using frequent minimal VULS, however, as already noted, this produced very few VULS and poor coverage (and consequently poor AUC results) so we can discount this. The second fastest approaches were when minimal VULS and frequent VULS were used. Reference to the previous results presented above show that the use of minimal VULS also produced good coverage and acceptable AUC results. From Table III it can also be noted that in most cases the recorded run time when using graphs with a degree of 8 was greater than when using graphs with a degree of 4. Furthermore, there are some cases where the used of frequent VULS required much more run time than when using all VULS, especially the cases highlighted in grey in Table III. This is because, although frequent VULSM requires less time than all VULSM, much more run time was required with respect to the BMV algorithm, thus producing a substantial overall run time result.

E. Statistical Comparison of VULSM Models

To determine whether the AUC results presented in the foregoing section were statistically significant or not, significance testing was conducted using the Friedman test and the Nemenyi post hoc test. The Friedman's test [10], [12], [13], [14], [15] as applied to the VULS concept was based on the Average Ranked (AR) performances of the 24 different VULS based classifiers. For this purpose the eight data used to produce the AUC results presented in Table I were used complemented with a further sixteen data sets generated again using a range of values for the number of discretised edge labels ($\{2, 3, 4, 5, 6, 7, 8, 9\}$), and $|L_V| = |C| = 3$ and $|L_V| = |C| = 4$. Where $L_V = 3$ the class set was $\{HouseHold, Forest, Field\}$; where $L_V = 4$ the class set was $\{HouseHold, Forest, BrownField, GreenField\}$. Thus a total 24 data sets (without assuming any specific data distribution) and the 24 different VULS mining based classification approaches were considered for the purpose of the significance testing. Recall that using different values for the grid size d was interpreted as representing a different VULS based classification approach, as was using different degree values (4 or 8).

All 24 VULS based classification approaches under consideration were ranked according to their AUC performance in ascending order with respect to each data set. The mean rank of each classifier j, AR_j, was then computed across all the data sets. With D representing the overall number of data sets, K the overall number of classifiers, and r^i_j the rank of classifier j with respect to data set i, the Friedman test statistic was calculated as follows:

$$\chi^2_F = \frac{12D}{K(K+1))}\left[\sum_{j=1}^{K} AR^2_j - \frac{K(K+1)^2}{4}\right]$$

$$AR_j = \frac{1}{D}\sum_{i=1}^{D} r^i_j$$

where χ^2_F is distributed according to the Chi-Square distribution with $K-1$ degrees of freedom ($D = 24$ and $K = 24$).

The null hypothesis, H_0, being tested was that there was no statistically significant difference between the operation of the VULS approaches. In other words that the performance differences observed with respect to the above reported AUC results was not statistically significant, but simply due to random chance. If the value of χ_F^2 is above a given threshold, then the null hypothesis that there is no difference in the operation of the classifiers can be rejected. The smallest level of significance that can result in the rejection of the null hypothesis is represented by a threshold value called the p-value. The p-value not only provides information about whether a statistical hypothesis test is significant or not, it also indicates something about "how significant" the result is. Note also that the smaller the p-value, the stronger the evidence against H_0. If H_0 can be rejected, a so-called post hoc test can be applied to detect which specific approaches differ significantly. In this respect $Dem\check{s}ar$ [15] recommended the use of the Nemenyi test. The Nemenyi post-hoc test [16] was thus applied so as to identify significant differences (if any) between the individual approaches. Using the Nemenyi post-hoc test the performances of two or more approaches (classifiers) is significantly different if their average ranks differ by at least a Critical Difference value (CD), given by:

$$CD = q_{\alpha,\infty,K}\sqrt{\frac{K(K+1)}{12D}} \quad (3)$$

The value $q_{\alpha,\infty,K}$ is based on the "studentized" range statistic and is tabulated in standard statistical textbooks. With respect to the evaluation presented here the outcome from applying the Friedman test and the Nemenyi post-hoc tests are presented using a modified version of $Dem\check{s}ar$'s significance diagrams [15], [14]. These diagrams display the ranked performances of the VULS approaches, along with their critical difference, to clearly indicate those approaches whose operation is significantly different from the other approaches (classifiers) in terms of recorded AUC value.

The Friedman statistic derived from the recorded AUC values for the 24 data sets considered are summarized in Table IV where the "AR" column gives the average AUC ranked performance and the "AR+CD" gives the average AUC value plus the calculated critical difference. The values in the "AR+CD" column can be used to determine wether one classification approach is statistically different from another.

Recall that the performance of two classifiers is significantly different if the corresponding AR values differ by at least the Critical Difference (CD). Note that in this case the chi-square (χ_F^2) value, using 23 ($K-1 = 24-1 = 23$) degrees of freedom, is 104.238, whilst the p-value (threshold) is $5.888E-11$. The F-distribution with 23 ($K-1 = 24-1 = 23$) and 529 ($(K-1)*(D-1) = (24-1)*(24-1) = 529$) degrees of freedom, $F(23,529)$ is 5.354. The critical value for $F(23,529)$, with a critical difference level of $\alpha = 0.05$, is 1.550. Thus, from the foregoing we can note that the p-value ($5.888E-11$) is smaller than 0.005 and that the F-distribution (5.354) is larger than the corresponding F-distribution critical value (1.550). Referring back to Table IV the average rank results comparing different VULS classifiers proposed in this

TABLE IV: Average Rankings of the VULS based classification aproaches

Algorithm	AR	AR+CD
d32-Deg4-VULS	10.917	16.166
d32-Deg8-VULS	8.917	14.166
d16-Deg4-VULS	11.063	16.312
d16-Deg8-VULS	8.333	13.583
d8-Deg4-VULS	13.313	18.562
d8-Deg8-VULS	7.667	12.916
d32-Deg4-MinVULS	8.917	14.166
d32-Deg8-MinVULS	7.958	13.208
d16-Deg4-MinVULS	8.750	14.000
d16-Deg8-MinVULS	8.625	13.875
d8-Deg4-MinVULS	15.958	21.208
d8-Deg8-MinVULS	13.833	19.083
d32-Deg4-FVULS	18.063	23.312
d32-Deg8-FVULS	13.750	19.000
d16-Deg4-FVULS	12.438	17.687
d16-Deg8-FVULS	15.125	20.375
d8-Deg4-FVULS	12.792	18.041
d8-Deg8-FVULS	15.521	20.770
d32-Deg4-FminVULS	14.354	19.604
d32-Deg8-FminVULS	15.917	21.166
d16-Deg4-FminVULS	14.167	19.416
d16-Deg8-FminVULS	13.563	18.812
d8-Deg4-FminVULS	14.125	19.375
d8-Deg8-FminVULS	15.938	21.187

paper were all significant ($p < 0.005$). Therefore we can reject H_0 (that the observed performance differences among the VULS approaches is not simply a matter of chance) and apply a Nemenyi post-hoc test to detect which particular VULS models differ significantly further.

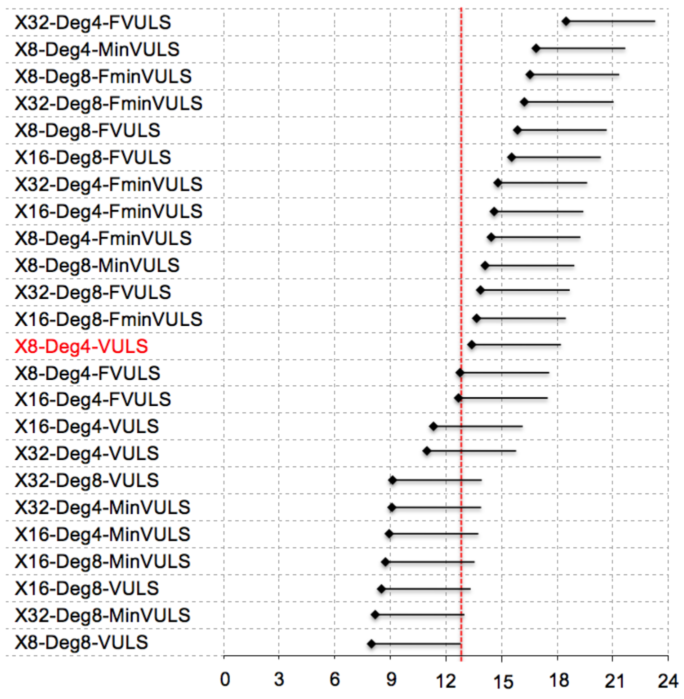

Fig. 3: Significance diagram to support the comparisons of the performance of the 24 VULS based classification approaches using Nemenyis post hoc test with $\alpha = 0.05$.

The significance diagram presented in Figure 3 corresponds to the information presented in Table IV and displays the AUC performance rankings for the models, along with Nemenyi's Critical Difference (CD) tail. The CD value for the diagram is equal to 5.250. The diagram shows the VULS models listed in ascending order of their ranked AUC performance (y-axis); whilst the mean AUC performance values are displayed along the x-axis. A vertical dashed line has been inserted in the figure to indicate the end of the best performing approachs tail so as to allow for comparison with respect to other approaches. From the Figure it can be seen that the best performing approach was that using the all VULS model with grid size of $d = 8$ and graphs featuring a vertex degree of 8 (a recorded AR value of 7.667). Using directed minimal VULS (with grid size of $d = 16$ or $d = 32$ and graphs featuring a vertex degree of 4 or 8) comparable performance was achieved; however, as established in Sub-section V-D the generation and usage of minimal VULS is much more efficient than when using the all VULS model. All the Frequent Minimal VULS approaches produced substantially the worst performance.

VI. Conclusions and further Study

In this paper a mechanism for predicting local class labels associated with 3D surface regions has been presented. The mechanism is founded on the idea of VULS mining where a VULS is a subgraph that has a unique labelling associated with it, in the context of some given training data set, that can be used to predict class labels to be associated with regions in previously unseen data. Four different types of VULS model were considered: (i) all, (ii) minimal, (iii) frequent and (iv) frequent minimal. This paper described the algorithms required to generate these different forms of VULS. The paper also presented the Backward Match Voting algorithm whereby detected VULS can be used to predict labels associate with unseen graphs. The algorithms were evaluated using a satellite image data set describing two villages located in a rural part of the Ethiopian hinterland. The results produced by the reported experiments indicate that the proposed VULS models tended to perform best when dealing with graph sets where the vertices have a degree of 8 and a grid size of $d = 8$. It was also noted that when using minimal VULS comparable performance with that obtained using all VULS was achieved, but in a more efficient manner. Using frequent VULS did not produce good results because very few frequent VULS were identified. It is suggested that better results using frequent VULS might be obtained using an alternative way of determining the threshold σ and this is thus a suggested avenue for future work.

The intension behind the evaluation was to investigate how well VULS can perform in the context of a satellite image application. Techniques for translating image data into the required grid graph format were not considered, but the manner in which this is conduced will clearly have some influence on the final VULS based classification performance. To date only simple 3D surface data has been considered. Alternative techniques for generating the desired grid graphs are seen as desirable. In the context of image data, such as the satellite data used for evaluation purposes in this paper, techniques for translating images in to graph formats exist (such as the Waxman model [17], K-means clustering, K-Nearest Neighbour clustering and so on). It is anticipated that use of these techniques will improve the performance of the propose VULS based classification approaches, this is thus a second anticipated avenue for future work.

References

[1] M. Firat, B. Kaftanoglu, and O. Eser, "Sheet metal forming analyses with an emphasis on the springback deformation," *Journal of Materials Processing Technology*, vol. 196, no. 1-3, pp. 135–148, 2008.

[2] W. Yu, F. Coenen, M. Zito, and S. E. Salhi, "Vertex unique labelled subgraph mining," in *Thirty-third BCS SGAI International Conference on Artificial Intelligence (BCS SGAI2013)*, 2013, pp. 21–38.

[3] W. Yu, F. Coenen, M. Zito, and S. E. Salhi, "Minimal vertex unique labelled subgraph mining," in *15th International Conference on Data Warehousing and Knowledge Discovery (DaWak 2013)*, 2013, pp. 317–326.

[4] W. Yu, F. Coenen, M. Zito, and S. E. Salhi, "Vertex unique labelled subgraph mining for vertex label classification," in *The 9th International Conference on Advanced Data Mining and Applications (ADMA 2013)*, 2013, pp. 542–553.

[5] M. J. Carlotto, "Spectral shape classification of landsat thematic mapper imagery," *Photogrammetric engineering and remote sensing*, vol. 64, pp. 905–914, 1998.

[6] M. J. Barnsley and S. L. Barr, "Distinguishing urban land-use categories in fine spatial resolution land-cover data using a graph-based, structural pattern recognition system," *Computers, Environment and Urban Systems*, vol. 21, no. 34, pp. 209 – 225, 1997, remote Sensing of Urban Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0198971597100011

[7] I. Walde, S. Hese, C. Berger, and C. Schmullius, "Graph-based mapping of urban structure types from high-resolution satellite image objects x2014;case study of the german cities rostock and erfurt," *Geoscience and Remote Sensing Letters, IEEE*, vol. 10, no. 4, pp. 932–936, July 2013.

[8] I. Walde, S. Hese, C. Berger, and C. Schmullius, "Graph-based urban land use mapping from high resolution satellite images," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-4, pp. 119–124, 2012. [Online]. Available: http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/I-4/119/2012/

[9] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proceedings of the 2002 International Conference on Data Mining*, 2002, p. 721.

[10] S. Garca and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, no. 12, pp. 2677–2694, 2008.

[11] K. Dittakan, F. Coenen, R. Christley, and M. Wardeh, "Population estimation mining using satellite imagery," in *Data Warehousing and Knowledge Discovery*, ser. Lecture Notes in Computer Science, L. Bellatreche and M. Mohania, Eds. Springer Berlin Heidelberg, 2013, vol. 8057, pp. 285–296. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40131-2_25

[12] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[13] I. Brown, "An experimental comparison of classification techniques for imbalanced credit scoring data sets using sas," in *SAS Global Forum 2012,Data Mining and Text Analytics*, 2012.

[14] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings." *IEEE Trans. Software Eng.*, vol. 34, no. 4, pp. 485–496, 2008.

[15] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[16] P. Nemenyi, *Distribution-free Multiple Comparisons*. Princeton University, 1963. [Online]. Available: http://books.google.co.uk/books?id=nhDMtgAACAAJ

[17] B. M. Waxman, "Routing of multipoint connections," *Selected Areas in Communications, IEEE Journal on*, vol. 6, no. 9, pp. 1617–1622, 1988.