

Graph Colouring Algorithm for Validating Labelled 2D Line Drawing Objects

M.Z. Matondang¹, A.A.Samah² H.Haron³ and H.A.Majid⁴

¹ Faculty of Computer and Information Systems, Universiti Teknologi Malaysia,
81310 UTM Skudai, Johor Bahru, Malaysia

² Faculty of Computer and Information Systems, Universiti Teknologi Malaysia,
81310 UTM Skudai, Johor Bahru, Malaysia

³ Faculty of Computer and Information Systems, Universiti Teknologi Malaysia,
81310 UTM Skudai, Johor Bahru, Malaysia

⁴ Faculty of Computer and Information Systems, Universiti Teknologi Malaysia,
81310 UTM Skudai, Johor Bahru, Malaysia

Abstract: Line labelling has been used to determine whether a two-dimensional (2D) line drawing object is a possible or impossible representation of a three-dimensional (3D) solid object. However, the results are not sufficiently robust because the existing line labelling methods do not have any validation method to verify their own result. In this research paper, the concept of graph colouring is applied to a validation technique for a labelled 2D line drawing. As a result, a graph colouring algorithm for validating labelled 2D line drawings is presented. A high-level programming language, MATLAB R2009a, and two primitive 2D line drawing classes, prism and pyramid are used to show how the algorithms can be implemented. The proposed algorithm also shows that the minimum number of colours needed to colour the labelled 2D line drawing object is equal to 3 for prisms and $n-1$ for pyramids, where n is the number of vertices (junctions) in the pyramid objects.

Keywords: *Graph colouring, line labelling, line drawing, validation.*

1. Introduction

Discussions on three-dimensional (3D) solid object reconstruction from two-dimensional (2D) single views reveal that there are several steps that need to be accomplished to transform a 2D drawing into a 3D view [1-3]. As the input of a reconstruction process, a 2D drawing could be a regular or an irregular line drawing. However, when the input is an irregular form or a sketch, then the input should be converted into a regular 2D line drawing first, before proceeding to the interpretation of the

drawing. However, it is assumed that the interpretation of a 2D single view into a 3D view is always an easy task. Generally, people assume that the interpretation of a drawing or an image as an object does not need conscious thought. Moreover, human vision itself appears effortless. For this reason, the validation of the drawing is rarely considered with regard to an object's transformation, especially when moving from a 2D view into a 3D view. Some mistakes arise because the drawing could be an impossible drawing or a possible drawing that cannot be represented as a common object, such as a matchbox. In other words, the validity of the 2D drawing (whether it represents an object or not) is important to know. A validation process filters a 2D drawing for a 3D reconstruction process, accepting the possible drawings and rejecting the impossible drawings. In this research paper, this validation process will be the main focus of the discussion.

Several previous studies show that line labelling has been used when representing 2D line drawings and in reconstructing 3D objects [4-8]. These studies have combined line labelling with geometric models to reconstruct 3D objects from 2D line drawings. Line labelling is also useful for identifying impossible objects and for validating 2D line drawings. The many line labelling scenarios used before provide motivation to use line labelling in the validation of 2D line drawings in the present research. However, in this research paper, the concept of graph colouring is applied to colour labelled 2D line drawings. The objective of the colouring process is the validation of the label of the 2D line drawing. We need to increase the speed of the validation process for 2D line

drawings, to determine whether a 2D line drawing contains a possible or an impossible 3D object. In addition, the colour labelling process is needed because the line labelling algorithm does not have any validation technique for its own result [9,10]. Readers interested in a detailed discussion about the line labelling algorithm are referred to [6-8,11]. The next section presents a discussion about the concept of graph colouring and some related issues.

2. Graph Colouring

In graph theory, a graph has a different meaning compared to statistical topics that utilise bar, circle and line graphs as representations of mathematical equations. Here, a graph is a model that consists of vertices and edges. Primarily, graphs illustrate real life problems and situations, such as social acquaintances, sports schedules, transportation routes, and computer networks [12].

By definition, a graph $G = (V, E)$ is a mathematical model consisting of a finite set of vertices $V = \{v_1, v_2, \dots, v_n\}$, which are represented by points, and a finite set of edges $E = \{e_1, e_2, \dots, e_m\}$, which are represented by line segments, where n and m are integers. Here, two vertices connected by an edge are said to be adjacent, and an edge is said to be incident to the vertices it connects. The degree (*deg*) of a vertex is the number of adjacent vertices. In this research paper, we discuss a special case in graph theory that is called graph colouring. Graph colouring is a special case of graph labelling, where colour is used as the label, so that there are no two adjacent vertices, edges, or faces that are assigned the same colour. If we are given a graph $G(V, E)$, then the chromatic number $\chi(G) = k$ is defined as the minimum numbers of colours needed to colour $G(V, E)$. Assume that $G(V, E)$ is a graph and for which $G^* = (V, C)$ is a mapping function $f : v \rightarrow c$ with $c \in C$ a finite set of colours, such that if $(v_1, v_2) \in E$, then $f(v_1) \neq f(v_2)$. This statement implies that adjacent vertices are not assigned the same colour [12].

There are three types of graph colouring, known as vertex, edge, and face colouring. In this research paper, the discussion is focused on edge colouring. The edge colouring of a graph, $G(V, E)$, is a colour assignment for each edge in $G(V, E)$, such that for each two adjacent edges e_j and e_k to a vertex v_i , those edges do not share the same colour. For a detailed discussion of graph colouring properties, readers can refer to [13] as an authoritative reference on graph colouring.

Some related literature and previous studies show that graph colouring is used to solve problems that may involve conflicts or items that need to be separated [12]. Several applications previously performed include separating chemicals during lab work, separating animals in a zoo, scheduling classes or exams, and (the most common application) colouring maps to separate distinct countries. Iturriaga-Velazquez [14] shows that the original problem involving the four-colour problem is the question of whether four colours are sufficient to colour the countries on a world map, never assigning the same colour to two countries with a common boundary. However, over time, graph colouring has been applied to many various fields of research. Marx [15] explained the applications of graph colouring to scheduling problems in his paper, while Gaceb et al. [16] carried out physical layout segmentation for postal sorting systems using a graph colouring application. Redl [17] used the graph colouring approach for university timetabling at the University of Houston, and Dobrolowski et al. [18] developed the Koala Graph Coloring Library, which is an open graph colouring library for real world applications. However, this research goes beyond previous studies and applications because we attempt to use the concept of graph colouring to develop an algorithm for the validation of labelled 2D line drawings. The proposed algorithm expedites the validation of 2D line drawings (to classify them as possible versus impossible objects), which is accomplished by a line labelling algorithm. This method is expected to provide better validation compared to the previous work of Matondang et al. [10], which is the only line labelling algorithm used to perform such a validation. The line labelling algorithm itself cannot be used as the validation technique to validate its own labelled 2D line drawing; in other words, the line labelling algorithm is not sufficiently robust for the labelling process to determine whether the line drawing is valid or invalid, representing a solid versus an impossible object, respectively. Therefore, the combination of both the line labelling algorithm and the graph colouring application can be useful to speed up and enhance the validation of a 2D line drawing. The graph colouring algorithm has been successful in many fields, which provides motivation to adapt it to the validation of 2D line drawings. The next section presents the proposed algorithm.

3. The Proposed Algorithm

This section presents the proposed algorithm for validating labelled 2D line drawings based on the concept of graph colouring. Our expected output is a valid, labelled 2D line drawing object with a different colour on each edge. Table 1 shows the algorithm.

Table 1: The proposed algorithm

Edge (line) colouring for labelled 2D line drawing object

- Step 1: input the number of vertices (junctions) in the labelled 2D line drawing given (object given).
%%start the edge labelling.
- Step 2: input every horizontal edge.
- Step 3: input every vertical edge.
*%%evaluate the greatest degree *deg in the object given.*
- Step 4: determine the greatest degree *deg for each vertex (junction) in the object given.
%%assume h, n as an integer number

```
for h=1:numberofvertex;  
    n = 1;  
    for i=1:numberofedge;  
        if labeledge(i,1) == V(h,1);  
            Degree(h,1) = n;  
            n = n + 1;  
        elseif labeledge(i,2) == V(h,1);  
            Degree(h,1) = n;  
            n = n + 1;  
        end  
    end  
end
```

- Step 5: colouring process
Assume m is an integer number

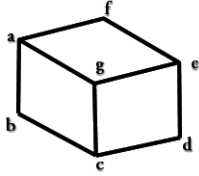
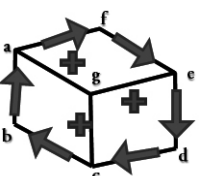
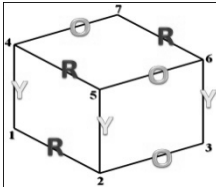
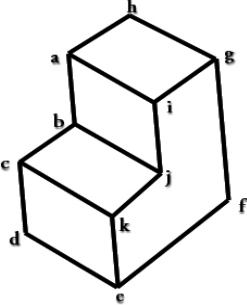
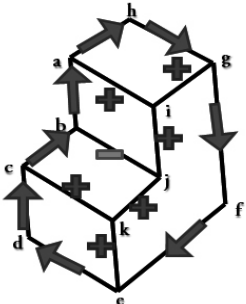
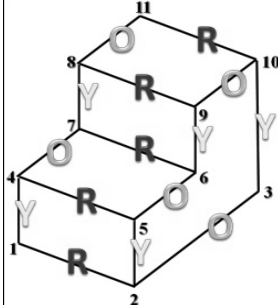
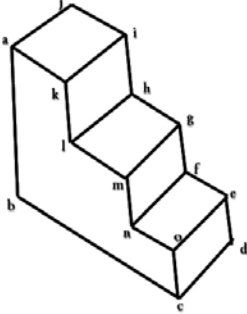
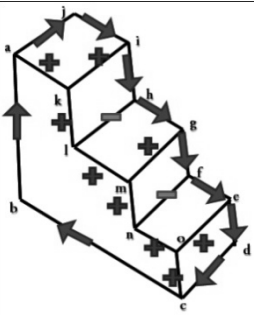
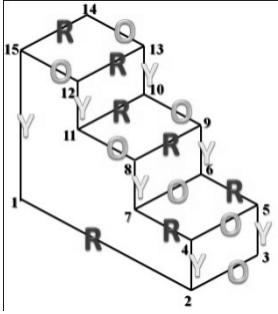
```
L = zeros(numberofedge,1);  
c = 0; %%assume c as the starting point for the iteration  
w = 1; %%assume w as the starting point for the vertex addressing in S  
for m=1:maxdegree  
    c = c + 1;  
    S = zeros(numberofvertex,1);  
        for j=1:numberofedge  
            if L(j,1)==0  
                if labeledge(j,1)~=S &  
                    labeledge(j,2)~=S  
                        L(j,1)=c;
```

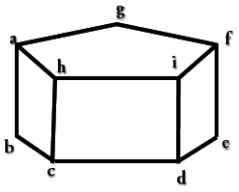
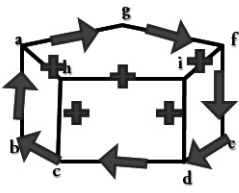
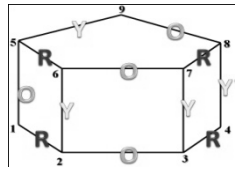
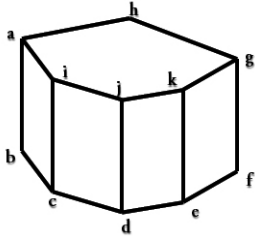
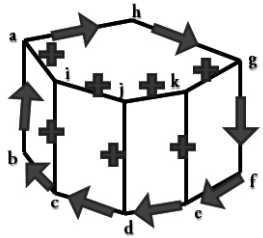
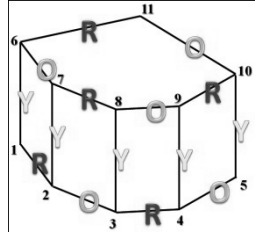
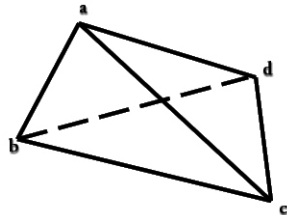
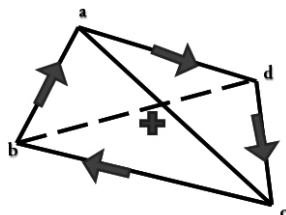
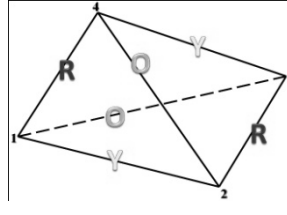
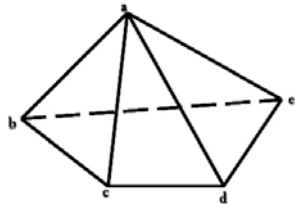
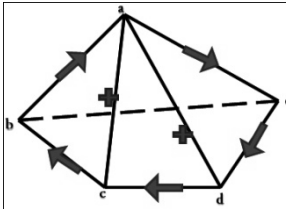
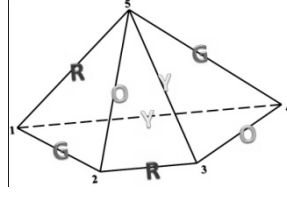
```
                S(w,1)=labeledge(j,1);  
                w = w + 1;  
                S(w,1)=labeledge(j,2);  
                w = w + 1;  
                elseif labeledge(j,1)==S  
                    ;  
                    L(j,1)=0;  
                    else labeledge(j,2)==S ;  
                    L(j,1)=0;  
                    end  
                end  
            end  
        end  
    end
```

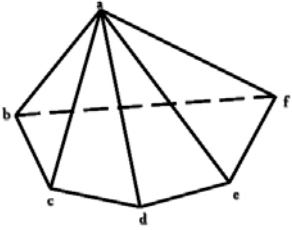
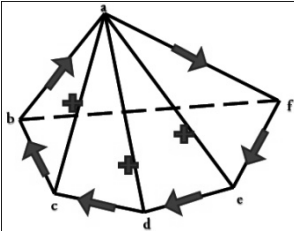
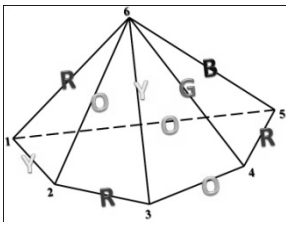
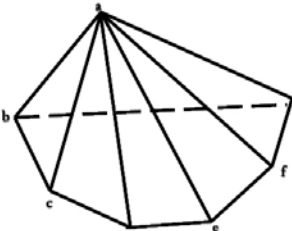
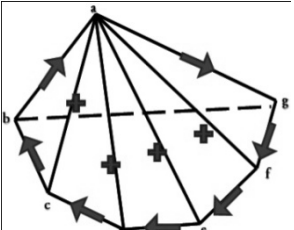
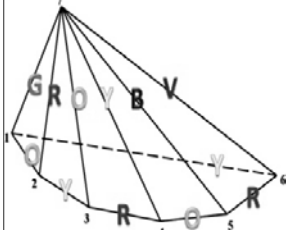
4. Experimental Results

In this section, we present results that show how the proposed algorithm has been implemented in two classes of 2D line drawing objects, namely the prism and pyramid. The results show the validation of the 2D line drawing using both the line labelling and the proposed graph colouring algorithm. However, assumptions have been made to simplify the implementation. The assumptions are as follows: First, the tested 2D line drawings are assumed to be an engineering sketch in the form of a 2D line drawing that represents a solid model. Second, the 2D line drawing is assumed to represent a valid solid model when all unwanted points or lines have been removed, and there are no unconnected points or lines. Third, the solid model is assumed to be a 2D line drawing with all of the informative lines shown. Fourth, there is only one hidden point in the backside of the solid model. These assumptions make the proposed algorithm more logical, or otherwise the engineering sketch is not seen as a solid model because the projection is parallel to the other faces of the object. In this case, it is impossible to interpret, reconstruct and represent the sketch as a solid model, and hence, the analysis of the accuracy of the results will become simpler.

Table 2: The experimental result

2D line drawing	Labelled 2D line drawing using line labelling algorithm	Valid-labelled of 2D line drawing with different colours at each edge	Role Model
 <p>Cube</p>		 <ul style="list-style-type: none"> • Red (R), Orange (O), Yellow (Y) 	<ul style="list-style-type: none"> • Label and colour (1,2) → red (2,3) → orange (4,5) → red (5,6) → orange (6,7) → red (7,4) → orange (1,4) → yellow (2,5) → yellow (3,6) → yellow • Number of visible vertices = 7 • Number of visible edges = 8 • Number of colours needed $k = 3$
 <p>L-block</p>		 <ul style="list-style-type: none"> • Red (R), Orange (O), Yellow (Y) 	<ul style="list-style-type: none"> • Label and colour (1,2) → red (2,3) → orange (4,5) → red (5,6) → orange (6,7) → red (7,4) → orange (8,9) → red (9,10) → orange (10,11) → red (11,8) → orange (1,4) → yellow (2,5) → yellow (8,7) → yellow (9,6) → yellow (3,10) → yellow • Number of visible vertices = 11 • Number of visible edges = 15 • Number of colours needed $k = 3$
 <p>Stairs</p>		 <ul style="list-style-type: none"> • Red (R), Orange (O), Yellow (Y) 	<ul style="list-style-type: none"> • Label and colour (1,2) → red (2,3) → orange (7,4) → red (6,5) → red (4,5) → orange (6,7) → orange (8,9) → red (9,10) → orange (10,11) → red (11,8) → orange (12,13) → red (13,14) → orange (14,15) → red (15,12) → orange (15,1) → yellow (11,22) → yellow (8,7) → yellow (10,13) → yellow (9,6) → yellow (2,4) → yellow (3,5) → yellow • Number of visible vertices = 15 • Number of visible edges = 21 • Number of colours needed $k = 3$

 <p>pentahedral-prism</p>		 <ul style="list-style-type: none"> Red (R), Orange (O), Yellow (Y) 	<ul style="list-style-type: none"> Label and colour (1,2) → red (2,3) → orange (3,4) → red (5,6) → red (6,7) → orange (7,8) → red (8,9) → orange (9,5) → yellow (6,2) → yellow (1,5) → orange (3,7) → yellow (4,8) → yellow Number of visible vertices = 9 Number of visible edges = 12 Number of colours needed $k = 3$
 <p>hexahedral-prism</p>		 <ul style="list-style-type: none"> Red (R), Orange (O), Yellow (Y) 	<ul style="list-style-type: none"> Label and colour (1,2) → red (2,3) → orange (3,4) → red (4,5) → orange (11,6) → red (7,8) → red (6,7) → orange (9,8) → orange (9,10) → red (10,11) → orange (1,6) → yellow (3,8) → yellow (4,9) → yellow (5,10) → yellow (2,7) → yellow Number of visible vertices = 11 Number of visible edges = 15 Numbers of colour needed $k = 3$
 <p>trihedral-pyramid</p>		 <ul style="list-style-type: none"> Red (R), Orange (O), Yellow (Y) 	<ul style="list-style-type: none"> Label and colour (4,1) → red (4,2) → orange (4,3) → yellow (1,2) → yellow (2,3) → red (3,1) → orange Number of visible vertices = 4 Number of visible edges = 5 Number of colours needed $k = 3$
 <p>kwartahedral-pyramid</p>		 <ul style="list-style-type: none"> Red (R), Orange (O), Yellow (Y) Green (G) 	<ul style="list-style-type: none"> Label and colour (5,1) → red (5,2) → orange (5,3) → yellow (5,4) → green (2,3) → red (3,4) → orange (4,1) → yellow (1,2) → green Number of visible vertices = 5 Number of visible edges = 7 Number of colours needed $k = 4$

 <p>pentahedral-pyramid</p>		 <ul style="list-style-type: none"> Red (R), Orange (O), Yellow (Y) Green (G), Blue (B) 	<ul style="list-style-type: none"> Label and colour (6,1) → red (6,2) → orange (6,3) → yellow (6,4) → green (6,5) → blue (2,3) → red (3,4) → orange (4,5) → red (5,1) → orange (1,2) → yellow Number of visible vertices = 6 Number of visible edges = 9 Number of colours needed $k = 5$
 <p>hexahedral-pyramid</p>		 <ul style="list-style-type: none"> Red (R), Orange (O), Yellow (Y) Green (G), Blue (B), Violet (V) 	<ul style="list-style-type: none"> Label and colour (7,2) → red (7,3) → orange (7,4) → yellow (7,1) → green (7,5) → blue (7,6) → violet (1,2) → orange (2,3) → yellow (3,4) → red (4,5) → orange (5,6) → red (6,1) → yellow Number of visible vertices = 7 Number of visible edges = 11 Number of colours needed $k = 6$

Based on the results shown in Table 2 and the assumptions that we made for implementing the proposed algorithm, we found the number of colours needed for a 2D line drawing's classification as a prism or a pyramid. For the prism class with n vertices (junctions), there are $n - 3$ vertices with $deg = 3$ and 3 vertices with $deg = 2$. The number of colours needed to colour the prism is equal to the maximum number of the degree $*deg$ and is valid only for prisms where the number of vertices is equal to $n \geq 6$.

For the pyramid class with n vertices (junctions), there are $n - 1$ vertices with $deg = 3$ and only one vertex with $deg = n - 1$. The number of colours needed to colour the pyramid is equal to $n - 1$ and is valid only for pyramids where the number of vertices is equal to $n \geq 4$.

5. Conclusions and Future Work

Line labelling has been used to determine whether a two-dimensional (2D) line drawing represents a possible or an impossible three-dimensional (3D) solid object. However, prior work is not sufficiently robust because the line labelling method does not have any validation method for its own result. In this research paper, a graph colouring method is applied as the validation technique for a labelled 2D line drawing object. As a result, the graph colouring algorithm for validating a labelled 2D

line drawing objects is presented. A high-level programming language, MATLAB R2009a, and two primitive 2D line drawing classes, prism and pyramid, are used to show how the proposed algorithm is implemented. Based on the experimental results, it is shown that for the prism class with n vertices (junctions), there are $n - 3$ vertices with $deg = 3$ and three vertices with $deg = 2$. The number of colours needed to colour the prism is equal to the maximum number of deg and is valid only for prisms having $n \geq 6$ vertices. Meanwhile, for the pyramid class with n vertices (junctions), there are $n - 1$ vertices with $deg = 3$ and one vertex with $deg = n - 1$. The number of colours needed to colour the pyramid is equal to $n - 1$ and is valid only for pyramids for which the number of vertices is equal to $n \geq 4$.

A suggestion for future research is to attempt to extend the algorithm for a more complex and general 2D line drawing that is not limited to objects in the prism and pyramid classes.

6. Acknowledgement

The authors would like to thank the Universiti Teknologi Malaysia (UTM) for UTM Short Term grant (Vot-77228), and Research Management Center (RMC) for the support in making this project a success.

References:

- [1] M. Z. Matondang, H. Haron and S. Thalib, "Three-dimensional visualization of two-dimensional data: the mathematical modeling," Proc. of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications, 13 - 15 June 2006, Penang, Malaysia, Computer Sciences and Applications. Ed. by Yahya Abu Hassan, Adli Mustafa and Zarita Zainuddin, Vol. 4.
- [2] M. Z. Matondang, Solid Model Reconstruction using Neural Network and the Mathematical Model Representation, Master Thesis, Universiti Teknologi Malaysia, 2009.
- [3] S. Mardzuki, Labeling Algorithm For Validation Of 2d Line Drawing, Master Thesis, Universiti Teknologi Malaysia, 2011.
- [4] T. Kanade, Recovery of the Three-Dimensional Shape of an Object from a Single View, *Artificial Intelligence*. 17 (1981) 409-460.
- [5] I. J. Grimstead and R. R. Martin, Creating Solid Model From Single 2D Sketches, *Proceedings Third Symp. On Solid Modeling Applications, ACM SIGGRAPH*. (1995) 233-337.
- [6] I. J. Grimstead, Interactive Sketch Input of Boundary Representation Solid Models, PhD Thesis, Univ. of Cardiff, UK, 1997.
- [7] P. A. C. Varley and R. R. Martin, Estimating Depth from Line Drawing, In Edt. K.Lee and Patrikalakis, *Proc. 7th ACM Symposium on Solid Modeling and Applications, SM'02*. ACM Press. (2002) 180-191.
- [8] P. A. C. Varley, R. R. Martin and H. Suzuki, Making the Most of using Depth Reasoning to Label Line Drawings of Engineering Objects, *ACM Symposium on Solid Modeling and Application*, (2004) 13-32.
- [9] S. Mardzuki, M. Z. Matondang, and H. Haron, Computational Approach in Validating Reconstructed Solid Model Based on Approximate Depth Value, *Proceeding the 5th International Conference on Information Technology and Applications (ICITA 2008)*, CAIRNS-Queensland Australia. (2008) 687 – 694.
- [10] M. Z. Matondang, S. Mardzuki and H. Haron, Transformation of engineering sketch to valid solid object, *Proc. of Intl. Conf. of The 9th Asia Pacific Industrial Engineering & Management Systems (APIEMS 08) Conference and The 11th Asia Pacific Regional Meeting of International Foundation for Production Research*, Bali – Indonesia. (2008) 2707–2715.
- [11] D. A. Huffman, Impossible Object as Nonsense Sentences, *Machine Intelligence*, Newyork: American Elsevier. 6 (1971) 295-323.
- [12] L. A. Robinson, Graph Theory for the Middle School, Master Thesis, Faculty of the Department of Mathematics, East Tennessee State University. UMI Number: 1436267. 2006.
- [13] T. R. Jensen and B. Toft, *Graph Coloring Problems*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons Inc., New York. A Wiley Interscience Publication, 1995.
- [14] C. C. Iturriaga-Velazquez, Map Labeling Problems, PhD Thesis, University of Waterloo, Ontario: Canada, 1999.
- [15] D. Marx, Graph Coloring Problems and Their Applications in Scheduling, *Periodica of Polytechnica Ser. El. Eng.* 489 (2004) 11-16.
- [16] D. Gaceb, V. Eglin, F. Lebourgeois, and H. Emptoz, Application of Graph Coloring in Physical Layout Segmentation, *IEEE 19th Intl. Conf. on Pattern Recognition*. (2008) 1-4.
- [17] T. A. Redl, *University Timetabling Via Graph Coloring: an alternative approach*, University of Houston, Houston, 2007.
- [18] T. Dobrolowski, D. Dereniowski, and L. Kuszner, *Koala Graph Coloring Library: An Open Graph Coloring Library for Real World Applications*, IT, Gdansk, Poland, May 2008.

M. Z. Matondang is a PhD student in computer science Universiti Teknologi Malaysia, Johor – Malaysia. He holds Master degree also in computer science from the same University in 2009 and Bachelor degree in mathematics from the Universitas Sumatera Utara, Medan, Indonesia in 2005. His researches interests are in operational research, soft computing and also in mathematical modeling.

A. A. Samah has received the Diploma and B.Sc. degree from University of Technology Malaysia in 1991 and 1993 respectively. In 1996, she obtained her M.Sc. from the University of Southampton, UK and recently in 2010, she received her PhD from Salford university, UK. Currently she is a lecturer in Faculty of Comp. Science and Information System, University of Technology Malaysia. Her research interests encompass Image Processing, Soft Computing Techniques and Operational and Simulation Modeling.

H. Haron has received the Dip, B.Sc. degree and PhD from University of Technology Malaysia in 1987, 1989 and 2004 respectively. He was awarded M.Sc. from the University of Brighton, UK in 1995. Currently he is an Associate Professor in Faculty of Comp. Science and Information System, University of Technology Malaysia. His research interests include Image Processing and Computer Aided Geometric Design.

H. A. Majid has received the Dip and B.Sc. degree from University of Technology Malaysia in 1993 and 1995 respectively. In 1998, he obtained his M.Sc. from the University of Salford, UK. Currently he is a lecturer in Faculty of Comp. Science and Information System, University of Technology Malaysia. His research interests focused on Image Processing, Operations Management and, Warranty and Maintenance.