

JVLC

**Journal of
Visual Language and
Computing**

Volume 2022, Number 1

Copyright © 2022 by KSI Research Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

DOI: 10.18293/JVLC2022-N1

Journal preparation, editing and printing are sponsored by KSI Research Inc.

**Journal of
Visual Language and Computing**

Editor-in-Chief

Shi-Kuo Chang, University of Pittsburgh, USA

Co-Editors-in-Chief

Gennaro Costagliola, University of Salerno, Italy

Paolo Nesi, University of Florence, Italy

Gem Stapleton, University of Brighton, UK

Franklyn Turbak, Wellesley College, USA

An Open Access Journal published by

KSI Research Inc.

156 Park Square Lane, Pittsburgh, PA 15238 USA

JVLC Editorial Board

Tim Arndt, Cleveland State University, USA

Paolo Bottoni, University of Rome, Italy

Francesco Colace, University of Salerno, Italy

Maria Francesca Costabile, University of Bari, Italy

Martin Erwig, Oregon State University, USA

Andrew Fish, University of Brighton, United Kingdom

Vittorio Fuccella, University of Salerno, Italy

Angela Guercio, Kent State University, USA

Erland Jungert, Swedish Defence Research Establishment, Sweden

Kamen Kanev, Shizuoka University, Japan

Robert Laurini, University of Lyon, France

Jennifer Leopold, Missouri University of Science & Technology, USA

Mark Minas, University of Munich, Germany

Brad A. Myers, Carnegie Mellon University, USA

Joseph J. Pfeiffer, Jr., New Mexico State University, USA

Yong Qin, Beijing JiaoTung University, China

Genny Tortora, University of Salerno, Italy

Kang Zhang, University of Texas at Dallas, USA

Journal Production Associate Editors

Jorge-Luis Pérez-Medina, Universidad de Las Américas, Ecuador

Yang Zou, Hohai University, China

Journal of Visual Language and Computing

Volume 2022, Number 1

February 2022

Table of Contents

Regular Papers

| | |
|--|----|
| Graphical Animations of the Lim-Jeong-Park-Lee Autonomous Vehicle Intersection Control Protocol. | 1 |
| <i>Dang Duy Buia, Win Hlaing Hlaing Myinta, Duong Dinh Trana and Kazuhiro Ogata</i> | |
| Precise Embodying Tree Structures onto Their Latent Feature Vectors | 16 |
| <i>Tiansi Dong</i> | |
| Analysis and Design of Land Cover Usage from Satellite Images using Machine Learning Algorithms. | 25 |
| <i>Rajeswara Rao Duvvada, Shaik Ayesha Fathima and Shaik Noorjahan</i> | |
| An Innovative Methodology of the Algorithmic Composition Knowledge Base for the Chinese Calligraphy Painting Interaction: from Sonification to Musification | 36 |
| <i>Chih-Fang Huang and Jenny Ren</i> | |

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

Graphical Animations of the Lim-Jeong-Park-Lee Autonomous Vehicle Intersection Control Protocol^{*,**}

Dang Duy Bui^a, Win Hlaing Hlaing Myint^a, Duong Dinh Tran^a and Kazuhiro Ogata^{a,*}

^aSchool of Information Science, Japan Advanced Institute of Science and Technology (JAIST), 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

ARTICLE INFO

Article History:

Submitted 3.1.2021

Revised 6.1.2021

Second Revision 8.1.2021

Accepted 10.21.2021

Keywords:

graphical animation
autonomous vehicle intersection protocol
state machine
state picture design

ABSTRACT

SMGA is a tool that mainly supports humans in visually perceiving the characteristics/properties of systems/protocols. Those characteristics could be used as lemmas to formally verify that the systems/protocols enjoy desired properties. The core task of the tool is to design a state picture that helps humans comprehend the systems/protocols better and then conjecture the characteristics. To demonstrate that SMGA can be applied to a wider class of systems/applications, we have graphically animated the Lim-Jeong-Park-Lee autonomous vehicle intersection control protocol with SMGA. The state machine formalizing the protocol uses composite data. We have revised SMGA so as to handle composite data. We design a flexible state picture for the protocol so that it is possible to deal with different initial states when the number of vehicles is less than or equal to a given number. In the conference version of the present paper, the visual representations of vehicles (or vehicle statuses) on each lane did not preserve the actual order of the vehicles on the lane. We have also revised SMGA so that it is possible to make a state picture design that preserves such an order. In the conference version, any information on conflict and concurrent lanes for each lane was not displayed. We have revised the state picture design so that such information can be visualized. Some characteristics are guessed by observing graphical animations based on the state picture design, and the characteristics are confirmed with model checking. The paper also summarizes several lessons learned as tips on how to design a state picture with composite data.

© 2022 KSI Research

1. Introduction

SMGA [15] has been developed to visualize graphical animations of systems/protocols. The main purpose of SMGA is to help human users be able to perceive non-trivial characteristics of systems/protocols by observing its graphical animations because humans are good at visual perception [10]. Those characteristics could be used as lemmas to formally verify that the systems/protocols enjoy some desired properties. It implies the usefulness of the tool since

*This work was partially supported by FY2020 grant-in-aid for new technology research activities at universities (SHIBUYA SCIENCE CULTURE AND SPORTS FOUNDATION).

**The present paper is an extended and revised version of the paper [14] presented at DMSVIVA 2021.

*Corresponding author

✉ bddang@jaist.ac.jp (D.D. Bui); winhlainghlaingmyint@jaist.ac.jp (W.H.H. Myint); duongtd@jaist.ac.jp (D.D. Tran); ogata@jaist.ac.jp (K. Ogata)

ORCID(s): 0000-0002-2700-1762 (D.D. Bui); 0000-0001-7092-2084 (D.D. Tran); 0000-0002-4441-3259 (K. Ogata)

DOI reference number: 10.18293/JVLC2022-N1-004

lemma conjecture is a challenging problem in formal verification. Several case studies of some protocols have been conducted with SMGA, such as Alternating Bit Protocol (ABP) [15], a communication protocol, Qlock and MCS protocols [3, 16, 5, 6] shared-memory mutual exclusion protocols, and Suzuki-Kasami protocol [4], a distributed mutual exclusion protocol, to demonstrate its power.

Nowadays, autonomous vehicles or self-driving cars are a trend of the era. They have many potentials that make humans more convenient. The Lim-Jeong-Park-Lee autonomous vehicle intersection control protocol (the LJPL protocol) [12] is one possible way to handle traffic control management for intersections through which vehicles/cars pass. It also has been formally specified in Maude by Moe et al. [2]. Thus, it is ready to graphically animate the LJPL protocol with SMGA so as to demonstrate that SMGA can be applied to protocols of autonomous vehicles/self-driving cars. Our motivations of the work described in the present paper are two-fold: (1) we would like to show that SMGA

can be applied to a wide class of protocols/systems, and (2) we would like to be prepared by conjecturing non-trivial characteristics for future formal proofs that the LJPL protocol enjoys some desired properties. The LJPL protocol was taken for motivation (1) as written. Moe et al. [2] conducted some model checking experiments for the LJPL protocol but did not do any formal proofs for the protocol. Standard model checking cannot guarantee that protocols/systems surely enjoy desired properties in general, although it is good at finding counterexamples. It would be necessary to use theorem proving so as to guarantee that protocols/systems surely enjoy desired properties. Formal proofs that the LJPL protocol enjoys desired properties is one piece of our future work for which we must use several lemmas. Motivation (2) is for this purpose.

We need to carefully make a state picture design in order to produce good graphical animations because it is considered a core task of the tool [5]. To make a state picture design, we first specify the protocol in Maude. The specification, however, contains some observable components whose values are composite (over one component value inside). It is non-trivial to deal with composite data with SMGA. One possible way to visualize such a composite value is to add extra observable components each of which stores a copy of a (component) value that composes the composite value, where observable components are what constitute states in our way to specify state machines and extra observable components play roles as auxiliary/ghost variables used in formal verification. This option may make state expressions unnecessarily complex. We did not take this option but have revised the tool so that users can design a state picture to be able to display (each component of) a composite value explicitly. Even if the number of vehicles is fixed, the LJPL protocol has multiple initial states. Given a natural number n , we make a flexible state picture design so that any initial states in which the number of vehicles is up to n can be handled. Some characteristics are then guessed by observing graphical animations generated from the design. The characteristics are also confirmed with Maude [7]. Some lessons learned are summarized as tips on how to design a good state picture with observable components whose values are composite.

In the conference version [14] of the present paper, the visual representations of vehicles (or vehicle statuses) on each lane did not preserve the actual order of the vehicles on the lane. We have also revised SMGA so that it is possible to make a state picture design that preserves such an order. In the conference version [14], any information on conflict and concurrent lanes for each lane was not displayed. We have revised the state picture design so that such information can be visualized.

The rest of the paper is organized as follows. Sect. 2 mentions some preliminaries such as state machines, Maude, and SMGA. Sect. 3 introduces the LJPL protocol. Sect. 4 describes formal specification of the LJPL protocol in Maude. In Sect 5, we describe how to graphically animate the LJPL protocol. We discuss some ideas of how to handle

observable components whose values are composite. Some characteristics of the LJPL protocol are then guessed by observing its graphical animations and confirmed by model checking in Sect. 6. Sect. 7 summarizes lessons learned as tips on how to design a state picture with composite data. Sect. 8 mentions some related work. Finally, we conclude the present paper in Sect. 9.

All of the state pictures and state sequences for SMGA presented in this paper are available at <https://bddang.bitbucket.io/>.

2. Preliminaries

A state machine $M \triangleq \langle S, I, T \rangle$ consists of a set S of states, a set $I \subseteq S$ of initial states, and a binary relation $T \subseteq S \times S$ over states. $(s, s') \in T$ is called a state transition and may be written as $s \rightarrow_M s'$. The set $R \subseteq S$ of reachable states with respect to M is inductively defined as follows: (1) for each $s \in I$, $s \in R$ and (2) for each $(s, s') \in T$, if $s \in R$, then $s' \in R$. A state predicate p is an invariant property with respect to M if and only if $p(s)$ holds for all $s \in R$. A finite sequence $s_0, \dots, s_i, s_{i+1}, \dots, s_n$ of states is called a finite computation of M if $s_0 \in I$ and $(s_i, s_{i+1}) \in T$ for each $i = 0, \dots, n - 1$.

In this paper, to express a state of S , we use a braced associative-commutative collection of name-value pairs. Associative-commutative collections are called soups, and name-value pairs are called observable components. That is, a state is expressed as a braced soup of observable components. The juxtaposition operator is used as the constructor of soups. Let $oc1, oc2, oc3$ be observable components, and then $oc1\ oc2\ oc3$ is the soup of those three observable components. A state is expressed as $\{oc1\ oc2\ oc3\}$. There are multiple possible ways to specify state transitions. In this paper, we use Maude [7], a programming/specification language based on rewriting logic, to specify them as rewrite rules. Maude makes it possible to specify complex systems flexibly and is also equipped with model checking facilities (a reachability analyzer and an LTL model checker). A rewrite rule starts with the keyword `r1`, followed by a label enclosed with square brackets and a colon, two patterns (terms that may contain variables) connected with \Rightarrow , and ends with a full stop. A conditional one starts with the keyword `cr1` and has a condition following the keyword `if` before a full stop. The following is a form of a conditional rewrite rule:

```
cr1 [lb] : l => r if ... /\ c_i /\ ...
```

where lb is a label and c_i is part of the condition, which may be an equation $lc_i = rc_i$ or a matching equation $lc_i := rc_i$. The negation of $lc_i = rc_i$ could be written as $(lc_i \neq rc_i) = \text{true}$, where $= \text{true}$ could be omitted. For a given subject term t , if there exist a sub-term t' of t and a substitution σ such that $t' = \sigma(l)$ and the condition $\dots /\ c_i /\ \dots$ holds under σ , t' in t is replaced with $\sigma'(r)$, where σ' is a substitution obtained by σ and substitutions calculated by matching equations. For $lc_i := rc_i$, lc_i may have new variables that do not appear in l and the other matching equations so far, while rc_i does not

have such new variables. $lc_i := rc_i$ holds if and only if there exists a substitution σ_i such that $\sigma_i(\sigma''(lc_i)) = \sigma''(rc_i)$, where σ'' is a substitution obtained by σ and substitutions calculated by the matching equations so far.

Maude provides the search command that allows finding a state reachable from t such that the state matches p and satisfies c :

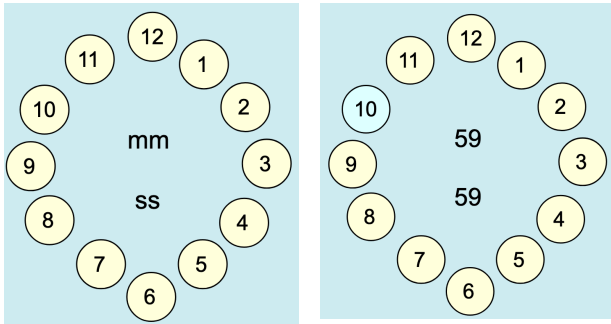
search [n,m] in MOD : $t \Rightarrow^* p$ such that c .

where MOD is the name of the module specifying the state machine, and n and m are optional arguments stating a bound on the number of desired solutions and the maximum depth of the search, respectively. n typically is 1 and t typically represents an initial state of the state machine.

State Machine Graphical Animation (SMGA) is a tool developed by Nguyen and Ogata [15]. SMGA does not automatically produce visual state picture design but it allows human users to design a good state picture. An input requires a state picture designed by humans and a state sequence generated by Maude. An output is graphical animations based on series of pictures in which each state is displayed based on the state picture design. There are two ways to visualize the observable component at each state that temporarily called (1) text display and (2) mark display. For example, one state that simulates a clock is specified as follows:

{(hh: 10) (mm: 59) (ss: 59)}

where hh, mm, and ss are observable components receiving 10, 59, and 59 as their values, respectively. The following figure displays a state picture design (on the left-hand side), and a state picture (on the right-hand side) of the example in which hh is presented as (2) (mark display), mm and ss are presented as (1) (text display).



3. Lim-Jeong-Park-Lee Autonomous Vehicle Intersection Control Protocol

Unlike the traditional traffic light mechanism, which has a drawback in choosing the optimal time and cycles of light signals with the different numbers of vehicles in different lanes of various intersections, the LJPL protocol provides an efficient solution to manage the traffic of intersections. Suppose that vehicles run on the right-hand side of a street and each side of a street has two lanes as shown in Figure 1. We also suppose that when a vehicle is crossing the intersection, if it is running on the right lane of its moving direction, then

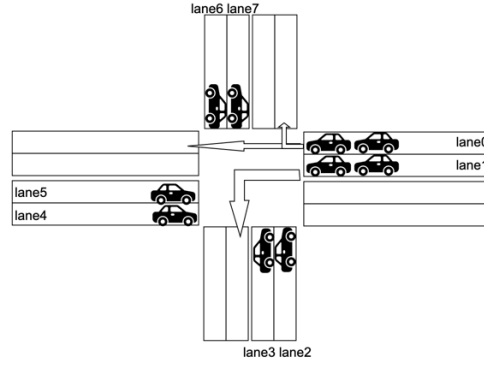


Figure 1: An example of intersection

it can only go straight or turn right. On the other hand, if the vehicle is crossing the intersection on the left lane, it can only turn left.

For each lane i , two relations between it and the other lanes are introduced as follows:

- *conflict lanes*: the conflict lanes of lane i are lane j for $j = (i + 2) \% 8, (i + 5) \% 8, (i + 6) \% 8, (i + 7) \% 8$ if $i = 0, 2, 4, 6$; and $j = (i + 1) \% 8, (i + 2) \% 8, (i + 3) \% 8, (i + 6) \% 8$ if $i = 1, 3, 5, 7$ (where $\%$ is the modulo operation). For example, lane0 and lane2 are conflict, meaning that vehicles on lane0 and those on lane2 are not allowed to go through the intersection simultaneously.
- *concurrent lanes*: the concurrent lanes of lane i are lane j for $j = (i + 1) \% 8, (i + 3) \% 8, (i + 4) \% 8$ if $i = 0, 2, 4, 6$; and $j = (i + 4) \% 8, (i + 5) \% 8, (i + 7) \% 8$ if $i = 1, 3, 5, 7$. For example, lane0 and lane4 are concurrent, meaning that vehicles on those two lanes are allowed to go through the intersection simultaneously.

Each vehicle has as its status one of the following five values: running, approaching, stopped, crossing, or crossed as its status. Let us note that Lim et al. do not use running & approaching statuses in the paper [12] and they use passing & passed statuses instead of crossing & crossed, respectively. There are eight queues of vehicles, each of them is associated with one of eight lanes. When a vehicle is far enough from the intersection, its status value is running, and when the vehicle is approaching the intersection shortly enough, its status changes to approaching, and its ID is enqueued into the queue associated with its lane. A vehicle is supposed to never change the lane and never pass the vehicles in front of it after its status changes to approaching. After a vehicle becomes approaching as its status, its status becomes stopped, which means that the vehicle stops in front of the intersection. Furthermore, the vehicle becomes lead if there is not any other vehicle whose status value is stopped in front of it; otherwise, it becomes non-lead. Note that there are two possible cases such that the vehicle becomes lead: (1) the vehicle is the top of the queue (i.e., there

is no other vehicle in front of it on the lane), and (2) there is another vehicle in front of it on the lane whose status value is crossing.

Every lead vehicle checks if there is no vehicle on any conflict lane crossing the intersection and the time given to it is earlier than those given to the lead vehicles on the conflict lanes (i.e., it arrives at the intersection earlier than any others). If so, the lead vehicle is allowed to go through the intersection, and its status changes to crossing. At the same time all non-lead vehicles following the lead vehicle and statuses are stopped are also allowed to go through the intersection together with the lead and their statuses also change to crossing. When a vehicle has crossed the intersection, the status of the vehicle becomes crossed and its ID is dequeued from the queue.

Some information needs to be exchanged among lead vehicles to synchronize the protocol (e.g., comparing the arrival times). Algorithm 1 and Algorithm 2 describe how a lead vehicle whose status value is stopped on a lane exchanges information with the lead vehicles on the four conflict lanes.

Algorithm 1. Basic IVC protocol (active thread)

```

1: begin at each round
2:    $vehicle_{target} \leftarrow selectVehicle();$ 
3:    $send(information_{local}, vehicle_{target});$ 
4:    $information_{target} \leftarrow receive(vehicle_{target});$ 
5:    $updateInformation(information_{local},$ 
                      $information_{target});$ 
6: end

```

Algorithm 2. Basic IVC protocol (passive thread)

```

1: repeat
2:    $vehicle_{target} \leftarrow waitForVehicle();$ 
3:    $information_{target} \leftarrow receive(vehicle_{target});$ 
4:    $updateInformation(information_{local},$ 
                      $information_{target});$ 
5:    $send(information_{local}, vehicle_{target});$ 
6: until forever

```

where IVC stands for inter-vehicle-communication [12] and $information_v$ for $v = local, target$ consists of the following information:

- *lane*: Lane number from 0 to 7
- *arrivalTime*: Arrival time for its own vehicle
- *arrivalTime_{lead}*: Arrival time for the lead vehicle
- *lead*: True or false
- *conflictLane*: List of conflict lanes
- *concurrentLane*: List of concurrent lanes
- *concurrentLanePassing*: List of concurrent lanes for passing vehicles
- *status*: stopped, passing, or passed

Let us repeat again that in the present paper, we use crossing & crossed instead of passing & passed; and we add running & approaching as the *status* values.

The LJPL protocol itself is described as Algorithm 3:

Algorithm3. Mutual exclusion algorithm via IVC

```

1: begin initialization
2:    $inforVehicles_i[j] \leftarrow null, \forall i \in \{1, \dots, max_{lane}\},$ 
    $\forall j \in \{1, \dots, max_{vehicle}\};$ 
3: end
4: begin when entering the intersection
5:    $lane \leftarrow getLaneNum();$ 
6:    $arrivalTime \leftarrow getCurrentTime();$ 
7:   if no vehicle on the lane,
   where  $status == stopped$  then
8:      $lead \leftarrow true;$ 
9:      $arrivalTime_{lead} \leftarrow arrivalTime;$ 
10:  else
11:     $lead \leftarrow false;$ 
12:  endif
13:   $status \leftarrow stopped;$ 
14: end
15: begin at each cycle
16:  update  $inforVehicles_i[j]$ 
   according to Algorithm 1 and Algorithm 2;
17:  check  $inforVehicles_{conflictLane}$ 
   for passing the intersection;
18:  if passingCondition() then
19:     $status \leftarrow passing;$ 
20:    move and cross the intersection;
21:  endif
22: end
23: begin when exiting the intersection
24:   $status \leftarrow passed;$ 
25: end
26: function passingCondition()
27:  if  $\forall arrivalTime_{lead} \in inforVehicles_{conflinctLane}$ 
    $> arrivalTime_{lead}$  and
    $\forall status \in inforVehicles_{conflictLane}$ 
    $== stopped$  then
28:    return true;
29:  else if  $\exists status \in inforVehicles_{concurrentLane}$ 
    $== passing$  and
    $\exists ! lane_i \in \{lane_n, \forall n \in \{0, \dots, max_{lane}\}$ 
    $| status == passing\}$  and
    $\forall arrivalTime_{lead}$ 
    $\in inforVehicles_{concurrentLanePassing}$ 
    $> arrivalTime_{lead}$  then
30:    return true;
31:  else
32:    return false;
33: end function

```

4. Specification of LJPL Protocol in Maude

As written, in this paper, a state is expressed as a braced soup of observable components. Let b is a Boolean value, q

is a queue of vehicle IDs (i.e., a queue of natural numbers because natural numbers are used as vehicle IDs). Let vid , lid , t , lt are natural numbers, where vid and lid represent a vehicle ID and a lane ID, respectively, while t and lt represent the time. To formalize the LJPL protocol as a state machine M_{LJPL} , we use the following observable components:

- $(clock : t, b)$ - it says that the current time is t . $clock$ represents the global clock shared by all vehicles. Initially, the first parameter of $clock$ is set to 0 and will increment. However, if time is allowed to increase without any constraints, the reachable state space will quickly explode. That is the reason why we introduce the second component b such that t only can increment when b is true. That is, whenever b is true, t can increment and b becomes false, and when a vehicle obtains the current time t (without changing t), b becomes true.
- $(v[vid] : lid, vstat, t, lt)$ - it says that the vehicle vid is running on the lane lid , its current *status* is $vstat$, it arrives at the intersection at the time t , and the lead vehicle of the lane lid reaches the intersection at the time lt .
- $(lane[lid] : q)$ - it says that the queue of vehicles running on lane lid is q .
- $(gstat : gstat)$ - it says whether all vehicles concerned have crossed, where $gstat$ is either fin or $nFin$. When it is fin , all vehicles concerned have crossed the intersection.

Each state in S_{LJPL} is expressed as $\{obs\}$, where obs is a soup of those observable components. We suppose that five vehicles (from 0 to 4) participate in the LJPL protocol such that two vehicles are running on lane0, one vehicle is running on lane1, and two vehicles are running on lane5. The initial state of I_{LJPL} namely $init$ is defined as follows:

```
{(gstat: nFin) (clock: 0, false) (lane[0]: oo)
(lane[1]: oo) (lane[2]: oo) (lane[3]: oo)
(lane[4]: oo) (lane[5]: oo) (lane[6]: oo)
(lane[7]: oo) (v[0]: 0, running, oo, oo)
(v[1]: 0, running, oo, oo) (v[2]: 1, running, oo, oo)
(v[3]: 5, running, oo, oo) (v[4]: 5, running, oo, oo)
(v[oo]: 0, stopped, oo, oo) (v[oo]: 1, stopped, oo, oo)
(v[oo]: 2, stopped, oo, oo) (v[oo]: 3, stopped, oo, oo)
(v[oo]: 4, stopped, oo, oo) (v[oo]: 5, stopped, oo, oo)
(v[oo]: 6, stopped, oo, oo) (v[oo]: 7, stopped, oo, oo)}
```

Initially, $gstat$ is set to $nFin$, the value of the global clock is 0. Since the second value of the $clock$ observable component is $false$, the abstract notion of the current time cannot increment. Each queue associated with each lane only consists of oo (denoting ∞), saying that there is no vehicle on the lane close enough to the intersection. $v[0]$ & $v[1]$ represent the two vehicles running on lane0, $v[2]$ represents the vehicle running on lane1, and $v[3]$ & $v[4]$ represent the two

vehicles running on lane5. There are eight $v[oo]$ observable components that are used to represent dummy vehicles.

12 rewrite rules are used to specify T_{LJPL} . Let OCs and OCs' be Maude variables of observable component soups, T , T' and T'' be Maude variables of natural numbers, and B is a Maude variable of Boolean values. When all vehicles have crossed the intersection, the state does not change anything, which is specified by the following two rewrite rules:

```
r1 [stutter] : {(gstat: fin) OCs}
=> {(gstat: fin) OCs} .
```

```
cr1 [fin] : {(gstat: nFin) OCs}
=> {(gstat: fin) OCs} if fin?(OCs) .
```

where $fin?(OCs)$ returns true iff all vehicles in OCs have crossed the intersection.

The rewrite rule $tick$ is defined to specify the behavior of the global clock:

```
r1 [tick] :
{(gstat: nFin) (clock: T, true) OCs} =>
{(gstat: nFin) (clock: (T + 1), false) OCs} .
```

The rewrite rule says that if the second value of the $clock$ observable component is $true$, the abstract notion of the current time T increments and the second value becomes $false$.

Two rules are used to specify a set of transitions that change a vehicle status from running to approaching as follows:

```
r1 [approach1] : {(gstat: nFin) (clock: T, B)
(lane[LI]: oo) (v[VI]: LI, running, oo, oo) OCs}
=> {(gstat: nFin) (clock: T, true)
(lane[LI]: VI) (v[VI]: LI, approaching, T, oo)
OCs} .
```

```
r1 [approach2] : {(gstat: nFin) (clock: T, B)
(v[VI]: LI, running, oo, oo)
(lane[LI]: (VI' ; VS)) OCs}
=> {(gstat: nFin) (clock: T, true)
(lane[LI]: (VI' ; VS ; VI))
(v[VI]: LI, approaching, T, oo) OCs} .
```

where LI , VI , and VI' are Maude variables of natural numbers, VS is a Maude variable of queues of natural numbers and ∞ , and $;$ is the constructor of queues. Note that $;$ is declared as an associative binary operator and each natural number or oo is declared as a singleton queue. Thus, $VI' ; VS ; VI$ denotes the queue obtained by putting VI into the queue denoted as $VI' ; VS$ at the end. The first rewrite rule specifies the case in which there is no vehicle close enough to the intersection on the lane where the vehicle is running, while the second one deals with the case in which there exists at least one vehicle close enough to the intersection on the lane.

Three rewrite rules are used to specify a set of transitions that change a vehicle status from approaching to stopped as follows:

```
r1 [check1] : {(v[VI]: LI,approaching,T,oo)
(gstat: nFin) (lane[LI]: (VI ; VS)) OCs}
=> {(gstat: nFin) (v[VI]: LI,stopped,T,T)
(lane[LI]: (VI ; VS)) OCs} .
```

```
r1 [check2] : {(v[VI]: LI,approaching,T'',oo)
(gstat: nFin) (v[VI']: LI,stopped,T,T')
(lane[LI]: (VS' ; VI' ; VI ; VS)) OCs}
=> {(gstat: nFin) (v[VI]: LI,stopped,T'',T')
(v[VI']: LI,stopped,T,T')
(lane[LI]: (VS' ; VI' ; VI ; VS)) OCs} .
```

```
r1 [check3] : {(v[VI]: LI,approaching,T'',oo)
(gstat: nFin) (v[VI']: LI,crossing,T,T')
(lane[LI]: (VS' ; VI' ; VI ; VS)) OCs}
=> {(gstat: nFin) (v[VI]: LI,stopped,T'',T')
(v[VI']: LI,crossing,T,T')
(lane[LI]: (VS' ; VI' ; VI ; VS)) OCs} .
```

where VS' is a Maude variable of queues. The first rewrite rule specifies the case in which vehicle VI is the top of the queue (i.e., VI will be lead on the lane). The second one deals with the case in which there exists another vehicle VI' in front of the vehicle VI such that VI' is stopped (VI will be non-lead on the lane). The last one specifies the case in which there exists another vehicle VI' in front of the vehicle VI such that the *status* of VI' is crossing (VI will be lead on the lane).

Two rewrite rules *enter1* and *enter2* are used to specify a set of transitions that change a lead vehicle *status* from stopped to crossing. *enter1* deals with the case in which the ID of the lane on which the lead vehicle is located is even and *enter2* deals with the case in which it is odd. *enter1* is defined as follows:

```
cr1 [enter1] : {(v[VI]: LI,stopped,T,T)
(gstat: nFin) (lane[LI]: (VI ; VS)) OCs}
=> {(gstat: nFin) (lane[LI]: (VI ; VS))
(v[VI]: LI,crossing,T,T) OCs'}
if isEven(LI) /\
LI1 := (LI + 2) rem 8 /\
(lane[LI1]: (VI1 ; VS1))
(v[VI1]: LI1,VSt1,T11,T12) OCs1 := OCs /\
VSt1 = stopped /\ T < T12 /\
LI2 := (LI + 5) rem 8 /\
(lane[LI2]: (VI2 ; VS2))
(v[VI2]: LI2,VSt2,T21,T22) OCs2 := OCs /\
VSt2 = stopped /\ T < T22 /\
LI3 := (LI + 6) rem 8 /\
(lane[LI3]: (VI3 ; VS3))
(v[VI3]: LI3,VSt3,T31,T32) OCs3 := OCs /\
VSt3 = stopped /\ T < T32 /\
LI4 := (LI + 7) rem 8 /\
(lane[LI4]: (VI4 ; VS4))
(v[VI4]: LI4,VSt4,T41,T42) OCs4 := OCs /\
VSt4 = stopped /\ T < T42 /\
OCs' := letCross(VS,OCs) .
```

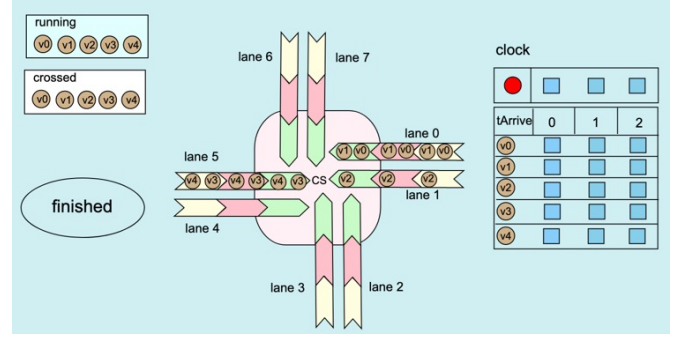


Figure 2: A state picture design for the LJPL protocol (1)

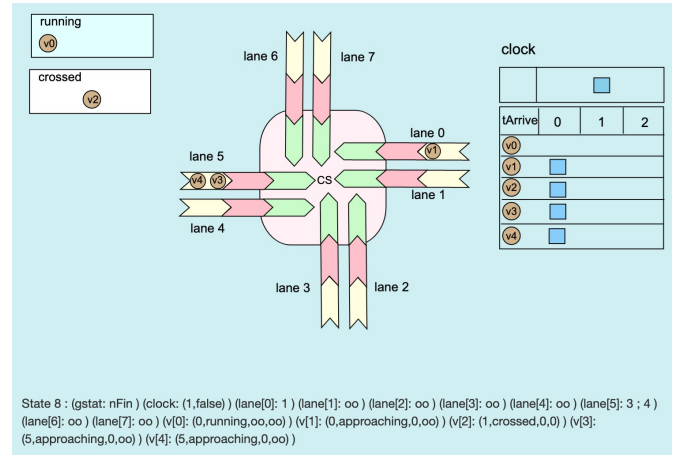


Figure 3: A state picture for the LJPL protocol (1)

where LIi for $i = 1, \dots, 4$ are Maude variables of natural numbers, VIi & Tj for $i = 1, \dots, 4$ & $j = 11, 12, \dots, 41, 42$ are Maude variables of natural numbers & ∞ , VS_i for $i = 1, \dots, 4$ are Maude variables of queues, VSt_i for $i = 1, \dots, 4$ are Maude variables of vehicle statuses, and OCs_i for $i = 1, \dots, 4$ are Maude variables of observable component soups. *isEven(LI)* holds if LI is even. The rewrite rule checks if all lead vehicles of the four conflict lanes (i.e., $LI1, LI2, LI3$, and $LI4$) are not crossing the intersection and the arrival time T of the vehicle VI is less than all arrival times of the lead vehicles on the conflict lanes. If the conditions are satisfied, the status of vehicle VI is changed to crossing from stopped and the statuses of all vehicles that follow VI and whose statuses are stopped also become crossing, which is done by *letCross(VS,OCs)*.

The rewrite rule *enter2* can be defined likewise. There are also two more rewrite rules *leave1* and *leave2* that are used to specify a set of transitions changing a vehicle *status* to crossed from crossing. All of them can be found from the webpage presented in Sect. 1.

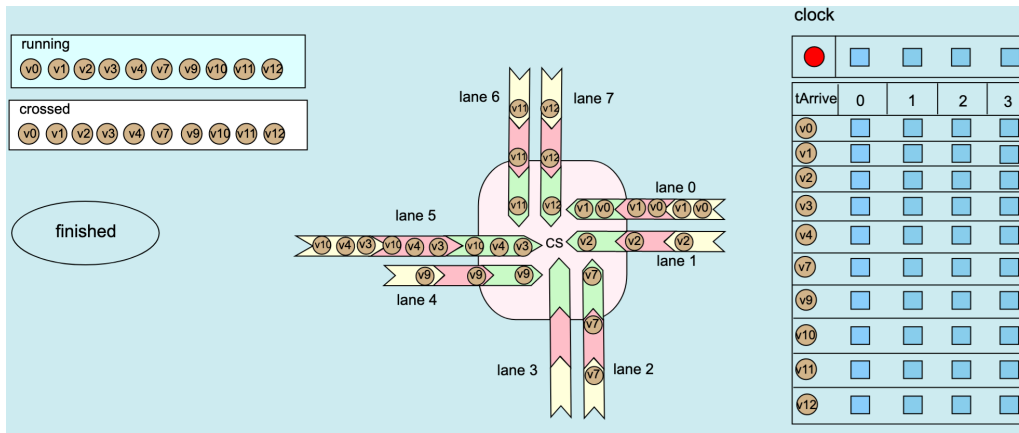


Figure 4: A state picture design for the LJPL protocol (2)

5. Graphical Animation of LJPL Protocol

5.1. Idea

At the beginning of the work, we needed to deal with a problem of how to design a good state picture so that it can display well a composite value of some observable component. At that time, the version of SMGA was only able to visualize observable components whose value is text or designated place. The tool, however, was not able to display specific components inside the composite values of the observable components. For example, considering the following observable component: $(time: (YY, MM, DD))$, SMGA was not able to display the value “YY”, “MM”, or “DD”. Therefore, we have modified the specification by adding some observable components that do not affect the behavior of the protocol. With the example above, three observable components are added: $(year: YY) (month: MM) (day: DD) (time: (YY, MM, DD))$. This is the key idea to makes the tool be able to produce good graphical animations for the LJPL protocol in particular and display observable components whose values are composite in general.

It is convenient for users if SMGA supports a functionality that can explicitly visualize specific component inside the composite values of the observable components without adding unneeded observable components. Therefore, we have revised the tool to support the functionality. The key idea is to add # followed by a natural number (start from 0) that represents a position inside the composite value of an observable component. For example, with the following composite value $(time: (YY, MM, DD(hh, mm, ss)))$, where its third component is also a composite value that consists of hh, mm and ss , we can extract the value mm by the notation $time\#2\#1$, where 2 denotes the third position of $time$'s value (i.e., $DD(hh, mm, ss)$), and 1 denotes the second position inside DD (i.e., mm). Therefore, users can display the component as a text or a designated place with the new functionality provided by SMGA.

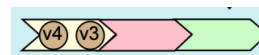
5.2. State Picture Design

In SMGA, designing a good state picture is an important task because it can help humans better perceive the characteristics of the protocol [5]. In our way to formalize the LJPL protocol, each state is expressed as a braced soup of observable components. Multiple similar observable components, such as v observable components, are used. v observable components contain values whose types are the same, such as the lane ID ($laneID$) and the $status$ ($vStat$) of a vehicle. By following the similarity principle of Gestalt [20, 19], they should be put together. Furthermore, the $laneID$ of a vehicle cannot be changed and hence we fix it as a constant text. We then come up with a state picture design for the initial state $init$ mentioned in the previous section (shown in Figure 2). A state picture generated from the state picture design is depicted in Figure 3.

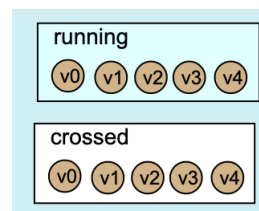
In Figure 2, there are eight arrow shapes representing eight lanes. A lane representation designed is as follows:



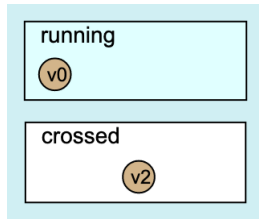
There are three colors: light green, pink, and light yellow that represent three statuses crossing, stopped, and approaching, respectively. For example, the status values of the fourth and fifth vehicles (i.e., $v3$ and $v4$) in the following figure are approaching:



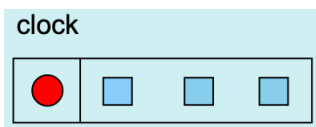
Two status values running and crossed representations used in Figure 2 are as follows:



A rectangle whose color is light cyan represents the status running. A rectangle whose color is white represents the status crossed. For example, the status values of the first and third vehicles (i.e., v0 and v2) in the following figure are running and crossed, respectively:



The design of the clock representation used in Figure 2 is as follows:



The value of the clock consists of two pieces of information: a natural number and a Boolean value (mentioned in Sect. 4). Three blue squares represent the natural number from 0 to 2. If the value of the natural number is 0, the first blue square is displayed. A red circle represents the Boolean value. If the value is *true*, a red circle is displayed, otherwise, nothing is displayed. For example, when the value of the natural number is 1, and the Boolean value is *false*, those values are displayed as follows:



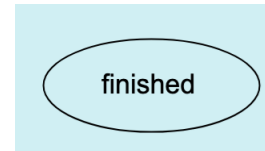
The design of the time arrivals of vehicles representation used in Figure 2 are as follows:

| tArrive | 0 | 1 | 2 |
|---------|---|---|---|
| v0 | □ | □ | □ |
| v1 | □ | □ | □ |
| v2 | □ | □ | □ |
| v3 | □ | □ | □ |
| v4 | □ | □ | □ |

In each horizontal line, three blue squares represent the value of the time arrival (from 0 to 2). If nothing is displayed, the value is ∞ . If the value is 0, the first blue square is displayed. For example, the figure below displays the case when the value of the first vehicle is ∞ , the values of the four other vehicles are 0:

| tArrive | 0 | 1 | 2 |
|---------|---|---|---|
| v0 | | | |
| v1 | □ | | |
| v2 | □ | | |
| v3 | □ | | |
| v4 | □ | | |

The design of the gstat representation used in Figure 2 is as follows:



If the value of gstat is *fin*, the circle and text is displayed, otherwise, nothing is displayed.

Figure 5 shows a state picture in which the initial state contains one vehicle in each lane1, lane2, lane4, lane6, and lane7, two vehicles in lane0, and three vehicles in lane5. It indicates that users need to redesign a new state picture since the initial state is changed. We design a flexible state picture such that it can be used when the number of vehicles participating in the protocol is small enough. Figure 7 displays the flexible state picture design, in which each lane can contain up to four vehicles, the value of the natural number of clock, and the value of the time arrival are up to 6.

5.3. Graphical Animation of LJPL Protocol

Figure 6 shows a state sequence for the LJPL protocol based on the state picture design depicted in Figure 5. Six pictures correspond to six consecutive states from State 13 to State 18 in one state sequence randomly generated by Maude. Those pictures follow the rewrite rules mentioned in 4, such as State 17 is the successor of State 16 by the rewrite rule *leave1*. Taking a look at the first picture (State 13) immediately makes us recognize that each of lane0 and lane5 contains two vehicles whose status values are stopped, each of lane6 and lane7 contains one vehicle whose status value is approaching, each of lane1 and lane2 contains one vehicle whose status value is stopped, the values of time arrival of those vehicles are equal to 0 except two vehicles whose status values are running have time arrival ∞ . Taking a look at State 13 and State 14 immediately makes us recognize that v12's status changes from approaching to stopped. Taking a look at State 15 to State 17 immediately makes us recognize that v2's status value changes from stopped to crossing and finally to crossed, and v4's status value changes from running to approaching.

Figure 8 shows another state sequence. Three pictures correspond to three consecutive states from State 27 to State 29. Taking a look at State 27 and State 28 makes us immediately recognize that three vehicles can change the status from stopped to crossing at the same time. It is interesting because crossing is regarded as the critical section such that

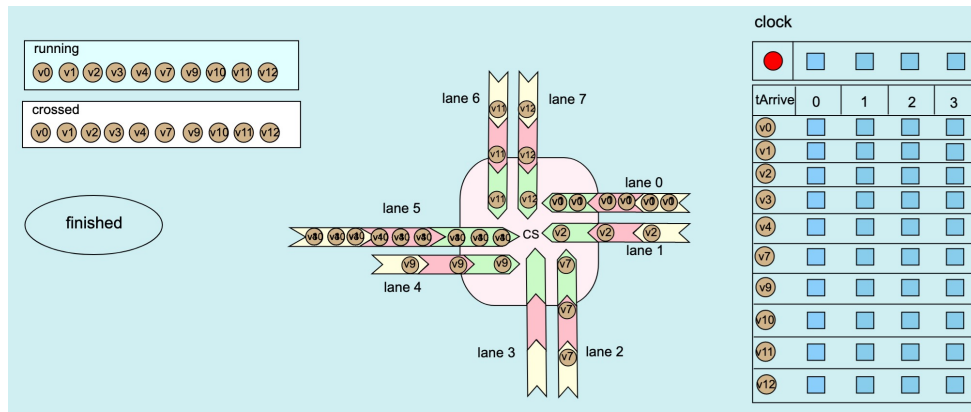


Figure 5: A state picture design for the LJPL protocol (3)

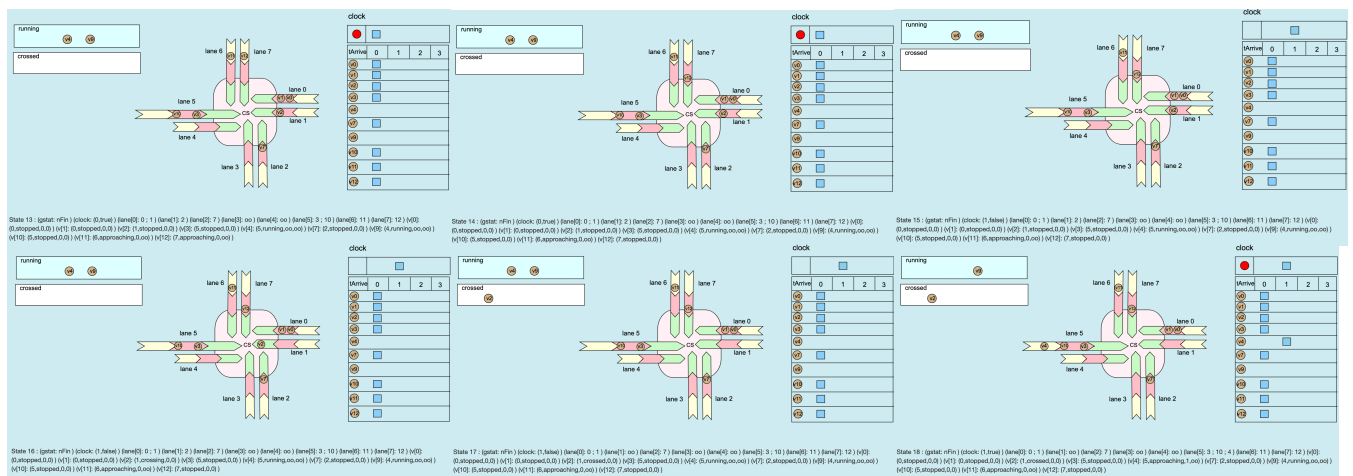


Figure 6: A state sequence for the LJPL protocol (1)

at most one vehicle should be located in the critical section at the same time. Taking a look at the State 28 makes us immediately recognize a case such that there exist two vehicles running on two different lanes, and their statuses are crossing. It can be explained that two vehicles running on two concurrent lanes (e.g., lane5 and lane2) are allowed to cross the intersection simultaneously.

5.4. Preservation of the Order of Vehicles on each Lane

In graphical animations of the LJPL protocol produced in our conference paper [14], the order of vehicles on each lane in each state picture instance may be different from the actual order of the vehicles on the lane. For example, in State 13 appearing in Figure 6, there are two v0 and v1 on lane0. v1 is the first vehicle and v0 is the second one on the state picture instance (State 13), while the lane[0] observable component has v0 ; v1 as its value, meaning that v0 is the first vehicle and v1 is the second one. This difference is not good because human users may mis-understand something about State 13 by looking at the state picture. This is because the

version of SMGA available when our conference paper [14] was written required us to fix the position for each vehicle on each lane when the vehicle status is one of the three statuses (approaching, stopped and crossing). Therefore, v1 is always in front of v0 in the state picture design used in the conference version even though v0 is actually in front of v1 when the statuses of both v0 and v1 are one of the three statuses.

One possible way to solve the situation is to use text display (see Sect. 2) to visualize queues of lane observable components. However, it is impossible to extract each vehicles' statuses from the queues because the queues only consists of vehicle IDs. For example, when the lane[0] observable component has v0 ; v1 as its value, the text v0 ; v1 is displayed. Because vehicles statuses are one piece of essential information of the protocol, this approach is not good enough.

We took a different approach. We have revised SMGA so that the tool can use mark display (see Sect. 2) to visualize queues of lane observable components so as to visualize vehicles statuses effectively. When a vehicle is in one of the three statuses, its ID is always displayed at a fixed

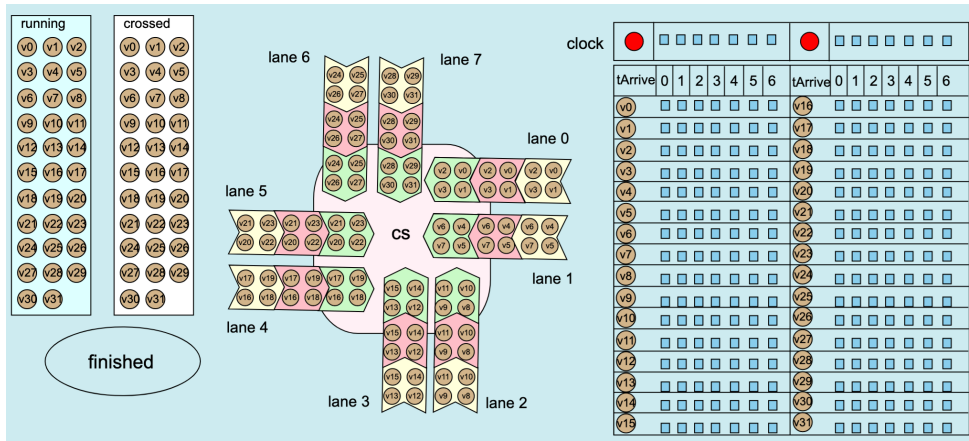


Figure 7: A flexible state picture design for the LJPL protocol (1)

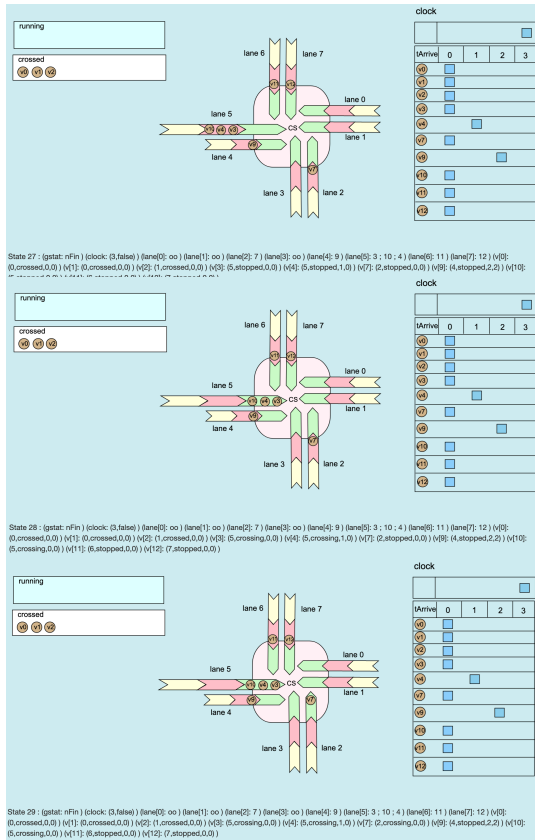
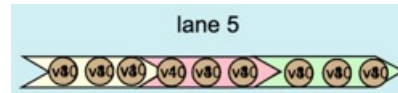


Figure 8: A state sequence for the LJPL protocol (2)

location of the arrow shape for the status in the previous approach. Thus, state pictures may not respect the order in which there are vehicles whose statuses are one of the three statuses (approaching, stopped and crossing) on a lane. On the other hand, in the current approach taken in the present paper, we allows the ID of such a vehicle to be displayed at any possible location of the arrow shape depending on the value (queue) of the lane observable component concerned.

For example, the design of the arrow shape for lane 5 is as follows:



We suppose that there are three vehicles v_3 , v_4 and v_{10} on lane 5. The design is made so that each of the three vehicles can be displayed at any of the three possible positions on each arrow sub-shape (note that there are three arrow sub-shapes on the lane).

When the value (queue) of the $lane[i]$ observable component is displayed, SMGA checks the status of each vehicle j in the queue by looking at the second component ($vstat$) of the $v[j]$ observable component. For example, when there are three vehicles v_3 , v_4 and v_{10} on lane 5 such that v_3 and v_4 are crossing and v_{10} is stopped, they are displayed as shown in Figure 9. The state picture respects the order of the three vehicles v_3 , v_4 and v_{10} on lane 5.

Figure 10 shows a sequence of state pictures produced by the latest version of SMGA that corresponds to the sequence of state pictures shown in Figure 8. Each state picture appearing in Figure 10 preserves the order of vehicles on each lane, while each state picture appearing in Figure 8 does not necessarily preserves the order.

5.5. Visualization of Conflict and Concurrent Lanes

For each of the eight lanes, there are four conflict and three concurrent lanes. In graphical animations of the LJPL protocol produced in our conference paper [14], any information on conflict and concurrent lanes is displayed. Information on conflict and concurrent lanes is not very static because each lane has different conflict and concurrent lanes. Furthermore, information on conflict and concurrent lanes is not stored in any observable components. Thus, such information cannot be handled as any other information, such as the status of each vehicle and the lane information. Be-

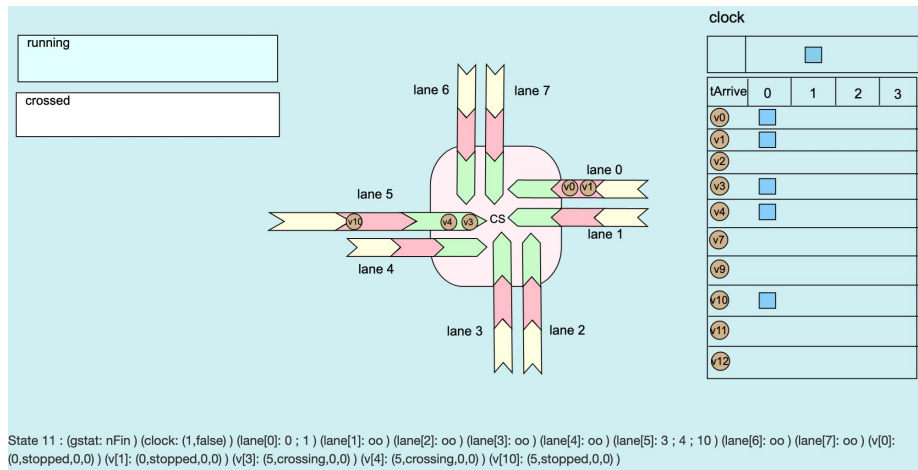


Figure 9: A state picture for the LJPL protocol (2)

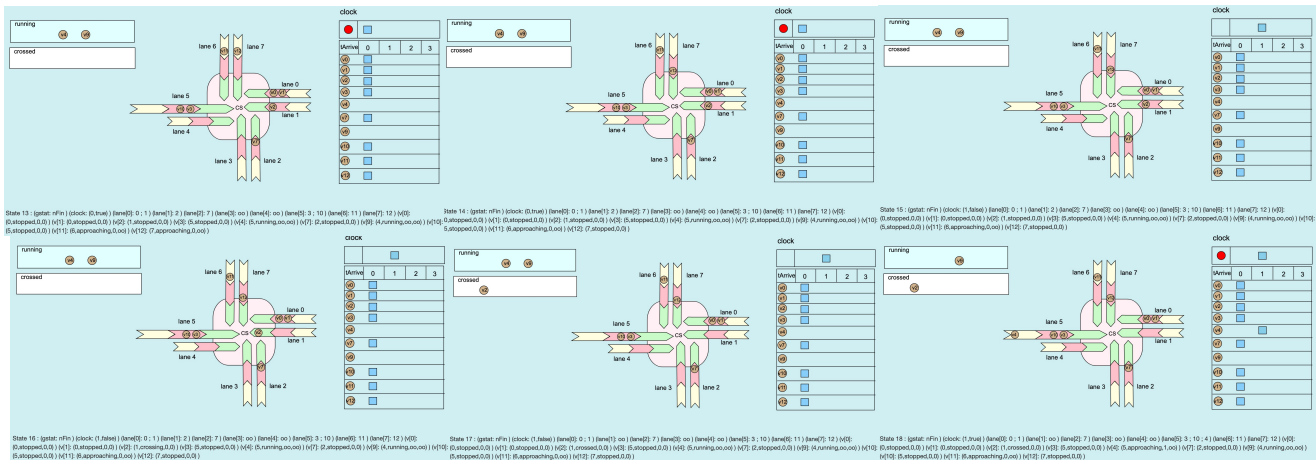


Figure 10: A state sequence for the LJPL protocol (3)

cause information on conflict and concurrent lanes is crucial for the LJPL protocol, it should be visualized.

Our approach to visualization of information on conflict and concurrent lanes is as follows. We make each text “lane i ” for $i = 0, 1, \dots, 7$ on each state picture clickable. Initially, each text “lane i ” is displayed in black. When “lane i ” is clicked, its four conflict lane texts become red and its three concurrent lane texts become blue, while “lane i ” is kept black. For example, when “lane 5” is clicked, each “lane j ” for $j = 0, 3, 6, 7$ becomes red and each “lane k ” for $k = 1, 2, 4$ becomes blue. Even while playing a graphical animation, each “lane i ” can be clicked. Figure 11 shows four state pictures in which “lane 5”, “lane 0”, “lane 2” and “lane 6”, respectively. The functionality that visualizes the conflict and concurrent lanes of “lane i ” by clicking “lane i ” is called the conflict/concurrent lane interaction functionality.

Let us note that there are four state pictures in Figure 11 but they represent one state (State 15). State 15 is an example in which a deadlock situation occurs, when no vehicle will

cross the intersection. It is reported by Moe et al. [2] that the original LJPL protocol does not enjoy the deadlock freedom property. When the lead arrival times of the two top vehicles on two conflict lanes are exactly the same, the original LJPL protocol cannot select one of the two vehicles. Moe et al. [2] propose that when that is the case, one vehicle whose lane ID is less than the other vehicle’s lane ID is selected. We use the revised protocol by Moe et al. [2] in the following sections.

6. Confirmation of Guessed Characteristics with Maude

We first guess four characteristics by observing graphical animations of the LJPL protocol and confirm them with the Maude search command. We then describe one seeming characteristic by observing graphical animations of the LJPL protocol. The characteristic is refuted by the Maude search command. We revise the characteristic and confirm it by the Maude search command. Let $init$ be the initial state

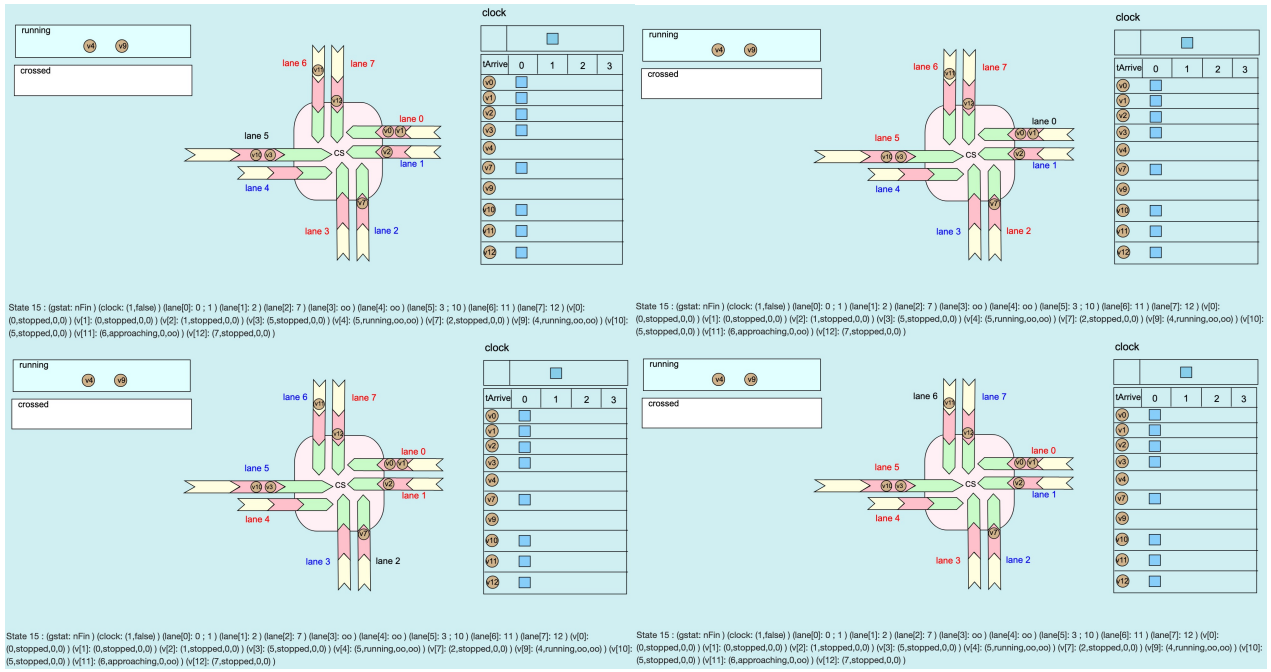


Figure 11: Four state pictures in which lanes 5, 0, 2, 6 are clicked, respectively

defined in Sect. 4.

6.1. Four Characteristics Gussed and Confirmed

Observing some graphical animations, we first see that the status of a vehicle sometimes does not change when the value of clock changes (shown at State 15 in Figure 6). Carefully focusing on the value of clock, we guess that when the Boolean value of clock is false, the arrival time of any vehicle cannot be greater than the first value of clock. The characteristic can be confirmed by the Maude search command as follows:

```
search [1] in RIMUTEX : init =>*
{(clock: T,false) (v[VI]: LI,VS,T1,T2) OCs}
such that
T1 >= T and T1 /= oo .
```

The search command above tries to find a reachable state in which the arrival time T_1 (that is not ∞) of a vehicle VI is greater than or equal to the first value T of clock. Maude does not find any reachable state that satisfies the condition from the state *init*. Therefore, the guessed characteristic is confirmed with the initial state *init*.

Observing some graphical animations, we guess that if the first value of clock is equal to the arrival time of a vehicle, the status of the vehicle is not running. The characteristics can be confirmed by Maude search command as follows:

```
search [1] in RIMUTEX : init =>*
{(clock: T,B) (v[VI]: LI,VS,T1,T2) OCs}
such that
T1 == T and VS == running .
```

The search command above tries to find a reachable state in which the arrival time T_1 of a vehicle VI is equal to the first value T of clock and the status of the vehicle is running. Maude does not find any reachable state that satisfies the condition from the state *init*. Consequently, the guessed characteristic is confirmed with the initial state *init*.

Observing some graphical animations with the conflict/concurrent lane interaction functionality, we recognize that if a vehicle is the top of lane LI_1 and its status is crossing, then another vehicle that is the top of lane LI_2 that is a conflict one of lane LI_1 is never crossing. The charismatic can be confirmed by the Maude search command as follows:

```
search [1] in RIMUTEX : init =>*
{(lane[LI1]: VI1 ; Q1) (v[VI1]: LI1,crossing,T11,T12)
(lane[LI2]: VI2 ; Q2) (v[VI2]: LI2,crossing,T12,T22) OCs}
such that
areConflict(LI1,LI2) = true /\ LI1 /= LI2 .
```

The search command above tries to find a reachable state in which two lanes LI_1 and LI_2 are conflict, two vehicles VI_1 and VI_2 are the top of the two lanes, respectively, and the two vehicles' statuses are crossing. Maude does not find any reachable state that satisfies the condition from *init*. Therefore, the guessed characteristic is confirmed with the initial state *init*.

Observing some graphical animations, we carefully focus on concurrent lanes. Let us note that there are three concurrent lanes for each lane. We recognize that there are at most two concurrent lanes on which vehicles are crossing simultaneously. The charismatic can be confirmed by Maude search command as follows:

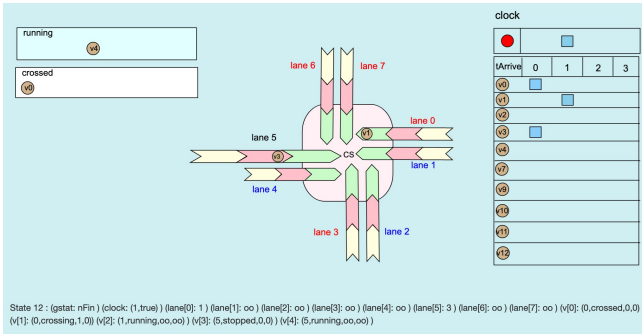


Figure 12: A state picture in which the seeming characteristic is broken

```
search [1] in RIMUTEX : init =>*
{(lane[L1]: v1 ; Q1) (lane[L2]: v2 ; Q2)
 (lane[L3]: v3 ; Q2)
 (v[V1]: L1,crossing,T11,T12)
 (v[V2]: L2,crossing,T21,T22)
 (v[V3]: L3,crossing,T31,T32) OCs}
such that
areConcur(L1,L2) /\ areConcur(L1,L3) /\
areConcur(L2,L3) /\ (L1 /= L2) /\ (L1 /= L3)
/\ (L2 /= L3) .
```

The search command above tries to find a reachable state in which three different lanes $L1$, $L2$ and $L3$ are concurrent, three vehicles $v1$, $v2$ and $v3$ are the top of the three lanes, respectively, and the three vehicles are crossing. Maude does not find any reachable state that satisfies the condition from `init`. Therefore, the guessed characteristic is confirmed with the initial state `init`.

6.2. One Seeming Characteristic and its Revision

By using one tip called CCT-2 (By concentrating on two different-kind observable components, we may find a relation between them, from which we may conjecture some characteristics.) [6], we carefully focus on the two top vehicles $VI1$ and $VI2$ of two conflict lanes $LI1$ and $LI2$, the statuses of $VI1$ and $VI2$ and the arrival times $T11$ and $T21$ of $VI1$ and $VI2$. We then guess that if $VI1$ is crossing, then $T11$ is less than or equal to $T21$. We tried to confirm the characteristic with the following Maude search command:

```
search [1] in RIMUTEX : init =>*
{(lane[LI1]: VI1 ; Q1) (v[VI1]: LI1,crossing,T11,T12)
 (lane[LI2]: VI2 ; Q2) (v[VI2]: LI2,VS2,T21,T22) OCs}
such that
areConflict(LI1,LI2) = true /\
LI1 /= LI2 /\ VI1 /= VI2 /\ T11 > T21 .
```

Maude found a counterexample of the guessed characteristic, although it is seemingly correct. Figure 12 shows a state picture of the counterexample. Let us take a look at the vehicles $v1$ on lane 0 and $v5$ on lane 5. The two lanes are conflict and the two vehicles are the top of the lanes as shown in Figure 12. $v1$ is crossing and $v5$ is stopped, while the $v1$'s arrival

time is 1 and the $v5$'s arrival time is 0. $v1$ is the top of lane 0 and seems to be the lead vehicle of lane 0, but $v1$ is not the lead vehicle of lane 0. $v0$ that is crossed was the lead vehicle of lane 0 because both its arrival time and lead arrival time (the third and fourth components of $v[v0]$ observable component) are 0 and $v1$ has been following $v0$ when crossing the intersection. This is because the lead arrival time of $v1$ is 0 that is the same as the arrival time of $v0$.

To guess the characteristic, we should have considered the lead arrival time instead of the arrival time of each vehicle concerned. We revise the characteristic as follows: if there are the top vehicles $VI1$ and $VI2$ of two conflict lanes $LI1$ and $LI2$ such that $VI1$ is crossing, the lead arrival time $T12$ of $VI1$ is less than or equal to the lead arrival time $T22$ of $VI2$. The revised characteristic can be confirmed by the following Maude search command:

```
search [1] in RIMUTEX : init =>*
{(lane[LI1]: VI1 ; Q1) (v[VI1]: LI1,crossing,T11,T12)
 (lane[LI2]: VI2 ; Q2) (v[VI2]: LI2,VS2,T21,T22) OCs}
such that
areConflict(LI1,LI2) = true /\
LI1 /= LI2 /\ VI1 /= VI2 /\ T12 > T22 .
```

Maude does not find any counterexamples.

7. Lessons Learned

Through the case study with the LJPL protocol, we obtain several lessons on how to design a good state picture so that we can conjecture some non-trivial characteristics, especially when there are some observable components with composite values. The lessons learned can be summarized as follows:

- When an observable component has a composite value, which consists of more than one component value inside, we need to carefully select which component values to visualize. For example, the second and the third component values (i.e., the *status* and the time arrival) of the vehicle observable component are selected while the fourth component value (i.e., the time arrival of the lead) of the vehicle observable component is not used in our design.
- If a value of an observable component does not change, it should be expressed at a fixed label, such as `laneID` of each vehicle observable component.
- If there exist observable components that have values whose types are the same, we should design and display the values together in one designated place.
- If there exist observable components that have a natural number as their values and the values are small enough, the values should be visually expressed nearby together so that we can see them simultaneously and compare them instantaneously. For example, the first value of `clock` (i.e., a natural number) and the time arrival of each vehicle have been visualized in our design.

8. Related Work

Bui and Ogata [4] have graphically animated a mutual exclusion distributed protocol called Suzuki-Kasami with SMGA. The protocol contains the network component used to exchange messages. They have realized the messages that have been put into the network and deleted from the network are crucial information so that they have revised SMGA to be able to display those messages. Their solution is to prepare two places for those messages. Some guessed characteristics are confirmed as invariant properties of the protocol with Maude. The authors have summed up their experiences as tips to help human users design a state picture for distributed protocol. This research and ours can share the working flow, but we cannot apply the technical content or tips to our work. One such reason is that the behavior of distributed protocol and autonomous vehicle intersection control protocol cannot share each other.

A mutual exclusion protocol called Qlock has been conducted by May Thu Aung et al. [3]. Some properties of the protocol have been conjectured by observing graphical animation. The properties also have model checked with Maude and theorem proved with CafeOBJ [8]. One piece of our future work is to theorem prove the characteristics guessed in this paper with CafeOBJ.

María Alpuente et al. [13] have proposed a methodology to check whether a Maude program is correct or not via logical assertions based on rewriting logic theories. They also have developed a prototype tool that is an implementation of that methodology. One functionality of the tool is to visualize the possible trace slides (as state sequences in our paper) to help users identify the cause of the error. Human users can observe the specific states corresponding to their rewriting rules by selecting them from a given initial state. In case that many possible rewriting rules may appear, the visualization is looked like a graph or a tree in which the states (displayed as text) are nodes. This visualization approach can be applied to our work, although its purpose is different than ours. One piece of our work is to compare those approaches together.

SMGA can be regarded as an integration of formal methods and visualization. We introduce two recent studies on an integration of formal methods and visualization. One [9] is a study on visualization of what is done inside by Vampire [11], an automated first-order logic theorem prover, and the other is a study on visualization of the structural operational semantics of a simple imperative programming language [17]. Although automated theorem provers are attractive because they may automatically prove theorems, they cannot truly fully automatically prove all possible theorems. Proof attempts may fail. If that is the case, human users need to comprehend why the proof attempts fail and need to change the format of input logical formulas and/or some internal proof strategies. It is very difficult for non-expert users and at least non-trivial for expert users to really comprehend why the proof attempts fail because it is necessary to understand what is done inside by an automated theorem prover, such as Vampire. Gleiss, et al. have then developed SATVIS,

a tool to visualize what is done inside by Vampire. Students and even programmers should learn semantics of programming languages so as to understand programming languages better, which may make it possible for them to write better programs. However, it is hard for students to learn semantics of programming languages. Perhác and Zuzana Bilanová have then developed an interactive tool for visualization of the structural operational semantics of a simple imperative programming language. SMGA partially shares the motivation of the first study [9]. This is because the main purpose of SMGA is to help human users conjecture lemmas needed for interactive theorem proving through graphical animations of state machines concerned. Nothing special is directly shared by SMGA and the second study [17] except an integration of formal methods and visualization. However, several formal semantics of programming languages have been described in the \mathbb{K} framework [18], where \mathbb{K} has been implemented in Maude. SMGA basically graphical animates state machines specified in Maude. Therefore, it would be possible to integrate SMGA and the \mathbb{K} framework so that formal semantics of programming languages can be visualized.

9. Conclusion

We have described graphical animations of the Lim-Jeong-Park-Lee autonomous intersection control protocol with SMGA in which the composite data are explicitly visually displayed. Two state picture designs that deal with initial states in two different ways have been created, and a flexible state picture design has been proposed so that all initial states can be handled provided that the number of vehicles is less than or equal to a given number. Some characteristics are guessed by observing graphical animations based on our design to demonstrate that graphical animation could help humans visually perceive the characteristics of the protocol. Those characteristics are confirmed by model checking. We have summarized our experiences as some tips on how to design a good state picture for autonomous vehicle intersection control protocol. One future direction is to apply our work to other self-driving vehicle protocols, such as a merging protocol [1]. Another future work is to integrate SMGA into Maude so that the tool can use some functionalities of Maude, such as pattern matching and generating a state sequence of state on the fly.

Acknowledgment

The authors would like to thank the anonymous reviewers who carefully read an earlier version of the paper and gave them valuable comments without which they were not able to complete the present paper.

References

- [1] Aoki, S., Rajkumar, R., 2017. A merging protocol for self-driving vehicles, in: ICCPS 2017, pp. 219–228. doi:10.1145/3055004.3055028.
- [2] Aung, M.N., Phyo, Y., Ogata, K., 2019. Formal specification and model checking of the Lim-Jeong-Park-Lee autonomous vehicle in-

- tersection control protocol, in: SEKE 2019, pp. 159–208. doi:[10.18293/SEKE2019-021](https://doi.org/10.18293/SEKE2019-021).
- [3] Aung, M.T., Nguyen, T.T.T., Ogata, K., 2018. Guessing, model checking and theorem proving of state machine properties – a case study on Qlock. *IJSECS* 4, 1–18. doi:[10.15282/ijsecs.4.2.2018.1.0045](https://doi.org/10.15282/ijsecs.4.2.2018.1.0045).
- [4] Bui, D.D., Ogata, K., 2019. Graphical animations of the Suzuki-Kasami distributed mutual exclusion protocol. *JVLC* 2019, 105–115. doi:[10.1007/978-3-319-90104-6_1](https://doi.org/10.1007/978-3-319-90104-6_1).
- [5] Bui, D.D., Ogata, K., 2020. Better state pictures facilitating state machine characteristic conjecture, in: *DMSVIVA 2020*, pp. 7–12. doi:[10.18293/DMSVIVA20-007](https://doi.org/10.18293/DMSVIVA20-007).
- [6] Bui, D.D., Ogata, K., 2021. Better state pictures facilitating state machine characteristic conjecture. *Multimedia Tools and Applications* doi:[10.1007/s11042-021-10992-z](https://doi.org/10.1007/s11042-021-10992-z).
- [7] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C. (Eds.), 2007. *All About Maude*. volume 4350 of *LNCS*. Springer. doi:[10.1007/978-3-540-71999-1](https://doi.org/10.1007/978-3-540-71999-1).
- [8] Diaconescu, R., Futatsugi, K., 1998. *CafeOBJ Report*. World Scientific.
- [9] Gleiss, B., Kovács, L., Schnedlitz, L., 2019. Interactive visualization of saturation attempts in Vampire, in: *IFM 2019*, Springer. pp. 504–513. doi:[10.1007/978-3-030-34968-4_28](https://doi.org/10.1007/978-3-030-34968-4_28).
- [10] K. W. Brodlić, et al. (Ed.), 1992. *Scientific Visualization: Techniques and Applications*. Springer. doi:[10.1007/978-3-642-76942-9](https://doi.org/10.1007/978-3-642-76942-9).
- [11] Kovács, L., Voronkov, A., 2013. First-order theorem proving and Vampire, in: *CAV 2013*, Springer. pp. 1–35. doi:[10.1007/978-3-642-39799-8_1](https://doi.org/10.1007/978-3-642-39799-8_1).
- [12] Lim, J., Jeong, Y., Park, D., Lee, H., 2018. An efficient distributed mutual exclusion algorithm for intersection traffic control. *J. Supercomput.* 74, 1090–1107. doi:[10.1007/s11227-016-1799-3](https://doi.org/10.1007/s11227-016-1799-3).
- [13] M. Alpuente, et al., 2016. Debugging Maude programs via runtime assertion checking and trace slicing. *J. Log. Algebraic Methods Program.* 85, 707–736. doi:[10.1016/j.jlamp.2016.03.001](https://doi.org/10.1016/j.jlamp.2016.03.001).
- [14] Myint, W.H.H., Bui, D.D., Tran, D.D., Ogata, K., 2021. Graphical animations of the Lim-Jeong-Park-Lee autonomous vehicle intersection control protocol, in: *DMSVIVA 2021*, pp. 22–28. doi:[10.18293/DMSVIVA2021-004](https://doi.org/10.18293/DMSVIVA2021-004).
- [15] Nguyen, T.T.T., Ogata, K., 2017a. Graphical animations of state machines, in: *15th DASC*, pp. 604–611. doi:[10.1109/DASC-PICom-DataCom-CyberSciTec.2017.107](https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2017.107).
- [16] Nguyen, T.T.T., Ogata, K., 2017b. Graphically perceiving characteristics of the MCS lock and model checking them, in: *7th SOFL+MSVL*, pp. 3–23. doi:[10.1007/978-3-319-90104-6_1](https://doi.org/10.1007/978-3-319-90104-6_1).
- [17] Perhác, J., Bilanová, Z., 2020. Another tool for structural operational semantics visualization of simple imperative language, in: *ICETA 2020*, IEEE. pp. 513–518. doi:[10.1109/ICETA51985.2020.9379205](https://doi.org/10.1109/ICETA51985.2020.9379205).
- [18] Rosu, G., 2017. \mathbb{K} : A semantic framework for programming languages and formal analysis tools, in: *Dependable Software Systems Engineering*. IOS Press. volume 50, pp. 186–206. doi:[10.3233/978-1-61499-810-5-186](https://doi.org/10.3233/978-1-61499-810-5-186).
- [19] Todorovic, D., 2008. Gestalt principles. *Scholarpedia* 3, 5345. doi:[10.4249/scholarpedia.5345](https://doi.org/10.4249/scholarpedia.5345).
- [20] Ware, C., 2004. *Information Visualization: Perception for Design*. MKP Inc.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

Precise Embodying Tree Structures onto Their Latent Feature Vectors

Tiansi Dong

University of Bonn, Germany

ARTICLE INFO

Article History:

Submitted 1.20.2022

Revised 1.26.2022

Accepted 1.30.2022

Keywords:

Deep Learning

Symbolic Structure

Geometric Unification

ABSTRACT

The limitivism philosophy holds that an accurate connectionist account can only approximate good symbolic descriptions within certain limit. Grounding symbolic structure onto the vector space has been researched in the literature but precise solution has not yet emerged. Here, we present a geometric method that embodies symbolic tree structures precisely onto learned vector representation. This method turns vector embedding of symbols into nested sets of n -spheres (spheres in a higher dimensional space), with two desirable properties: (1) each vector embedding is well preserved by the central point of an n -sphere; (2) symbolic tree structures are precisely encoded by inclusion relations among n -spheres. This unified representation bridges the gap between Deep Learning and symbolic structural knowledge. Significant experiment results are obtained by embodying a large hypernym trees word-sense tree onto GloVe word embeddings of tree nodes. Our geometric method shows a new way to completely resolve the antagonism between connectionism and symbolicism.

© 2022 KSI Research

1. Introduction


A concept can be understood from two perspectives, one from the inside – its content, in terms of a set of features, the other from outside – its connections with other concepts, in terms of a symbolic structure. In the battle between the two perspectives, both sides believe they explain the same phenomena [26]. If we imagine the two perspectives as different eyes of the monster *Artificial Intelligence*, how can this monster construct the external world in its mind using the inputs from the two heterogeneous eyes? Precisely, How, if possible at all, can discrete symbolic structures be (precisely) unified with their own feature vectors?

The two perspectives belong to the two paradigms in Artificial Intelligence, namely the *symbolic paradigm* and the *connectionist paradigm*. The symbolic paradigm is concerned with structural knowledge and rules for inference and decision making. A typical symbolic system consists of three components [9]: (i) symbols, either primitive or constructed; (ii) the meaning of constructed symbols, interpreted via the meaning of primitive symbols and the way of construction; (iii) reasoning via symbolic manipulations. The connectionist paradigm is inspired by the physiology of the brain and

models cognitive capabilities in terms of networks of simple computational nodes. Problem solving in this paradigm is to design and to train networks using exemplary data, in which knowledge is implicitly represented by weights of connections between nodes.

Approaches in the two AI paradigms are based on complementary mechanisms and target different levels of cognitive analysis [45, 50]. Symbolic approaches excel at reasoning but can hardly learn and are vulnerable to noisy inputs. Connectionist approaches, in particular Deep Learning [32], are robust to noisy or unforeseen inputs and capable of learning from data. However, they lack explainability, and are limited to approximated reasoning [25, 10], can be deliberately fooled by adversarial inputs [44, 28], and requiring much more training data than human learners would need [29]. Nevertheless, connectionist approaches can make sense of data via *similarity judgments* [25, 52] and thus simulate one of three judgment methods under uncertainty [53].

An open challenge remains with respect to the question of how connectionist approaches can reach symbolic levels of reasoning [3, 49] or achieve cognitive modeling [39]. Researches in hybrid neuro-symbolic approaches aim at realising robust connectionist learning and sound symbolic reasoning [1, 2, 21, 16, 5]. Most of the approaches utilise interface neural networks to approximately bridge vector embed-

 dongt@bit.uni-bonn.de (T. Dong)

 <https://tiansidr.github.io/> (T. Dong)

ORCID(s):

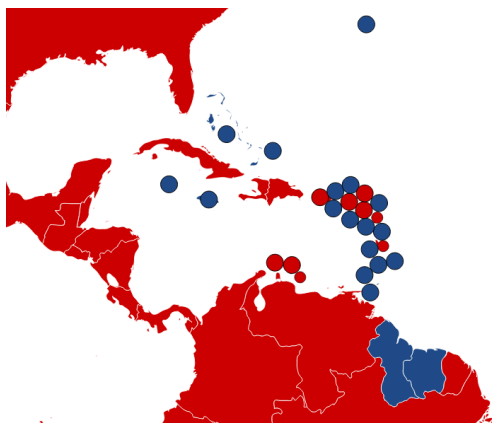


Figure 1: Blue areas represent islands with Left Hand Traffic System; red areas represent islands with Right Hand Traffic System

dings and symbols [17], in a way to roughly ground abstract symbols into a vector space [23, 24]. Geometrical structure is advocated as a potential cognitive representation apart from symbols or connectionist networks [18, 19].

Here, we geometrically construct a unified representation of a symbolic tree structure and its node embeddings: (1) vectorial node embeddings are promoted onto spheres in higher dimensional space; (2) the symbolic tree is spatialised onto these spheres such that inclusion relations among spheres precisely embody symbolic tree structures. The existence of these spheres shows the possibility to create a geometric continuum between symbolic and connectionist models, to completely resolve the antagonism between connectionism and symbolism, and to realise the hope that to design artificial cognitive systems (the mind of the AI monster) that combine the two complementary paradigms (inputs from its two eyes) and solve problems from both perspectives [38, 9].

2. The Structuralist Eye Cannot Be Replaced by the Connectionist Eye

The *eliminativism* claims that connectionist approaches can sweep away (*eliminate*) symbolic approaches, and was refuted by observing that neural-networks were computed by symbolic computational devices (Turing machine) [7]. In this section, we give more evidences to refute the *eliminativism* perspective, starting from a thought experiment.

2.1. You won't trust neural navigators in way-finding

On your birthday party, your wonderful birthday gift is placed in the middle of a large maze. You are given a route instruction to find it at the end of the route. You are extremely excited, take the route instruction, and dive into the maze, and forget to ask how to get out of the maze. You call your friends outside the maze for help. They tell you the gift is a neural navigator. Given the current route instruction, it produces the next route instruction. Your friends try to convince you that a sequence of route instructions can be under-

stood as a long sentence, and the neural navigator is exactly trained by the route you have, and routes of all other mazes around the world, with the same training mechanism as that for word-embeddings, e.g., [36]. Will you trust such kinds of neural navigator? If you know symbolic approaches to get out of the maze, such as the wall-follower method, or simply reversing the current route instruction, will you choose this neural navigator?

2.2. Connectionist eyes cannot see anything that structuralist eyes see

Figure 1 illustrates places where different traffic systems are used. The blue areas use the left-hand traffic system, the red areas use the right-hand traffic system. Only being fed with sufficient traffic scenarios, can autonomous driving cars learn there are two traffic systems by themselves? If they switch among right-hand traffic and left-hand traffic places, they cannot learn whether there are two traffic systems at all. It is hard to imagine that deep-learning systems can be intelligent enough to generate concepts of left and right, if they are only fed with traffic images without labeling which traffic systems. Because the concept of being left or right is not originated from images of street scenes. The original meaning of the left hand refers to the hand that is close to the heart of the body. Even for humans, the term of being left/right may not exist. For example, Guugu Yimithirr people only have absolute orientation coordinates, such as north, south in their spatial conceptual system. A Guugu Yimithirr speaker would say something like “I left some tobacco on the southern edge of the western table in the house” [33, 41]. Connectionist eyes cannot see anything that structuralist eyes see. To teach connectionist eyes see objects, we have to teach them by correctly imposing object names with object images [4, 36, 46, 14, 57, 22, 47, 32]. External knowledge must be imposed onto the connectionist networks. This is not new to connectionists. In image recognition, they shall first precisely label object names to each image in the training set. If each cat image is labelled as ‘dog’, the well-trained networks will recognize each cat image as ‘dog’.

2.3. Structures can exist without data

The existence of laws lies in the fact that violation exists in the reality. That *stealing is not allowed* as a law is due to the fact that *stealing* behaviors exist in the society. Even *stealing* behaviors does not exist, it still holds that *stealing is not allowed* – A phrase may be denoting, and yet not denote anything [43, p.41]. Only fed with data of stealing behaviors, connectionist networks would be more likely to mimic stealing behavior, rather than to be enlightened that stealing is not allowed. Excluding all stealing data from the training set, connectionist networks may not learn the concept of *stealing* at all.

2.4. Mental Representation of Partial Tree Structures

Structural knowledge is often modeled in terms of relations between or among entities. Figure 2(a) illustrates a

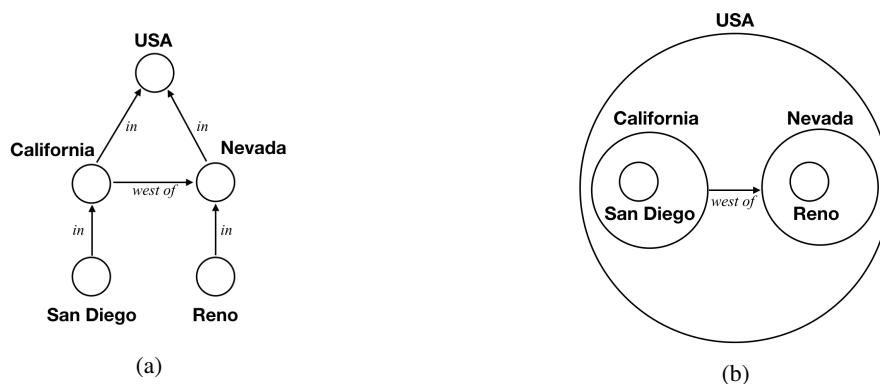


Figure 2: (a) A simple semantic network of spatial knowledge among California, Nevada, San Diego, and Reno; (b) A partially region-based hierarchical structure among California, Nevada, San Diego, and Reno

spatial structure among the cities San Diego and Reno, the state of California and Nevada and the USA. Most people mistakenly judge that San Diego (CA) is further west than Reno (NV). The reason is that tree nodes are represented as regions in mind and that only relations between locations inside the same spatial region are explicitly stored [48], as illustrated in Figure 2(b). This region-based representation can explain systematical errors that people made in reasoning with spatial knowledge [54, 35].

The real challenge for connectionists is not to defeat symbolic theorists, but rather to come to terms with the ongoing relevance of the symbolic level of analysis [3, p.16]. Here, we examine the possibility to promote vectors of entities into regions such that inclusion relation shall precisely represent the child-parent relation in the tree, as illustrated in Figure 2(b).

3. The Statement of the Problem, and the Challenges

The problem addressed here can be stated as follows: Given a symbolic tree structure, and vector representations of its nodes, can we embody each tree node into a sphere such that (1) each child-parent relation in the tree structure (seen from the structuralist eye) is precisely encoded as inclusion relations among spheres; (2) the vector representation of a tree node (seen from the connectionist eye) is very well preserved by the sphere of the tree node.

The first criteria can be re-formulated within the connectionists' community as the criteria of reaching global loss zero. In the literature of connectionism, the termination condition of training processes only needs to be a local minimum. As we need to precisely encode all symbolic relations into inclusion relations among regions, we require global loss zero. A small scaled experiment in [11] shows that it is not possible to reach global minimum zero only by utilising the back-propagation method.

The second criteria may suggest us to create a sphere for a tree node by taking its vector as the central point of the sphere. This turns out to be not realistic. Take a hypernym tree as the example. In many cases, words, such as

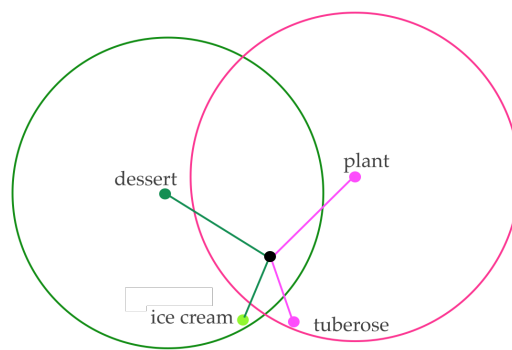


Figure 3: dessert sphere partially overlaps with plant sphere, although they should be disconnected from each other

ice_ cream, tuberose, and their superordinate words, such as dessert, plant, seldom occur in the same context, their vector embeddings differ to such a degree that the cosine value is less than zero. For example, using GloVE embedding [40], the cos value of ice_ cream and dessert is -0.1998 , the cos value of tuberose and plant is -0.2191 . This follows that the dessert sphere by taking the vector embedding as the central point, will contain the origin point of the embedding space, if it contains ice_ cream sphere. The plant sphere by taking the vector embedding as the central point will contain the origin point of the embedding space, if it contains tuberose sphere. Then, dessert sphere overlaps with plant sphere, as illustrated in Figure 3. Such overlapping is not allowed, as there is no entity which are both dessert and plant.

4. Constructing Spheres in Higher Dimensional Spaces

Promoting vectors into spheres appears deceptively simple, as it seems that we only need to add two new elements for each vector: one representing the length of the central point vector, the other representing the radius. Could the back propagation method be successful for this task? Experiments show that it cannot guarantee to achieve the target configuration precisely [12]. We abandon back propagation method, and use geometric construction and illustrate the

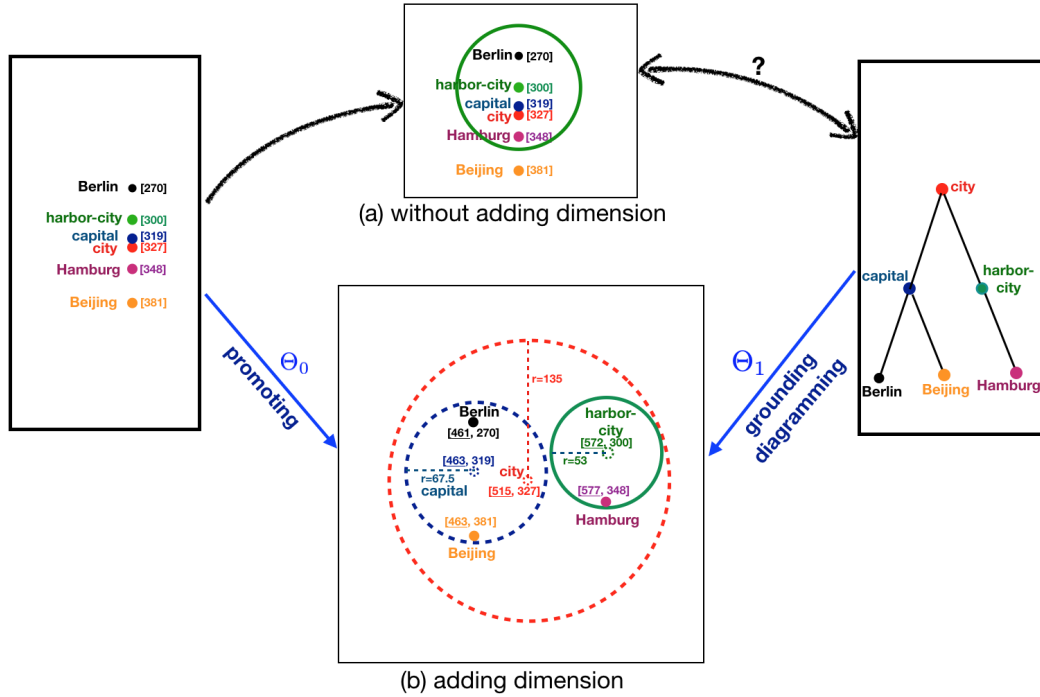


Figure 4: Spatializing symbolic structures onto vector space. Connectionist network represents a word as a one-element vector. Biased by training sentence, Hamburg vector [348] is closer to city vector [327] and captial vector [319] than to harbor-city vector [300]. To precisely encode symbolic tree structures, we promote them into circles, for example, harbor-city vector to a circle with central point [572, 300] with radius=53, with the target that inclusion relations among circles encode child-parent relations in the tree structure

method with the example as follows: Let the sentence “it was proposed to construct a maglev train between Berlin, capital of Germany, and harbor-city Hamburg” be in a training set, connectionist networks can capture co-occurrence relations among words in terms of vectors (word-embeddings). Figure 4 illustrates one dimensional word embeddings, e.g., the word embedding of Berlin is [270]. Suppose that Hamburg is a harbor-city, as a piece of truth-knowledge, be not clearly stated in the training text. As a result, Hamburg vector is closer to city vector and capital vector than to harbor-city vector, which violates the symbolic structure that Hamburg is-a harbor-city and Hamburg is-not-a capital. Such bias may lead connectionists and statisticians to stay at the level of approximation in reasoning and to believe the limitivism philosophy that *an accurate connectionist account can only approximate good symbolic descriptions within certain limit*. Our novel method is to promote a one dimensional word-embedding [y] into two dimensional circles with central point [x, y] and radius r. Then, we gear all xs and rs, so that inclusion relations among circles precisely encode the symbolic tree structure, as illustrated in Figure 4. Adding dimension is due to the fact that the space to embed semantic relations of words may not be the same as the space to embed their co-occurrence relations. Here, a symbol is embodied as a sphere in high dimensional space with the restriction that a part of the center vector is the vector embedding from connectionist networks. That is, symbols are only partially landed onto the vector embedding space. Through

such a symbol spatialising process, we embody a symbolic tree structure into a continuous space, so that the brittleness problem of symbolic approaches is removed. We describe the symbol spatializing process for a tree structure as follows.

4.1. Method

A tree \mathfrak{T} is a relational structure that can be described as a relational structure (T, S) [6] in which

1. T is the set of nodes $\{t_1, t_2, \dots, t_n\}$
2. S is the set of node pairs $\{(t_i, t_j) | t_i, t_j \in T\}$
3. \mathfrak{T} has a unique root node r that for any other node t there is a unique chain $[r = t'_1, \dots, t'_i, \dots, t = t'_w]$ under the condition that neighborhood nodes, t'_j and t'_{j+1} , are node pairs in S.
4. For every non-root node u , there is a unique node pair (v, u) in S.
5. For any node x , (x, x) does not exist in S

We geometrically represent each tree node t as a sphere \odot_t with central point O_t and radius r_t , and node pair as the cover relation COV as follows: (t_i, t_j) is in S, if and only if \odot_{t_i} covers \odot_{t_j} , written as $\text{COV}(\odot_{t_i}, \odot_{t_j})$. Geometrically, we define that \odot_{t_i} covers \odot_{t_j} , if and only if radius r_{t_i} is *greater than* the sum of r_{t_j} and the distance between their central points $\|O_{t_i} - O_{t_j}\|$, that is, $r_{t_i} > r_{t_j} + \|O_{t_i} - O_{t_j}\|$. This *greater than* relation excludes the case that a sphere covers

| sphere | contained by spheres |
|---|--|
| $\odot(\text{beijing.n.01})$ $\odot(\text{berlin.n.01})$ | $\odot(\text{city.n.01}) \subset \odot(\text{municipality.n.01}) \subset \odot(\text{region.n.03})$ $\subset \odot(\text{location.n.01})$ |
| $\odot(\text{berlin.n.02})$ | $\odot(\text{songwriter.n.01}) \subset \odot(\text{composer.n.01}) \subset \odot(\text{musician.n.02})$ $\subset \odot(\text{artist.n.01})$ |
| $\odot(\text{hamburg.n.01})$ | $\odot(\text{port.n.01}) \subset \odot(\text{point.n.02}) \subset \odot(\text{location.n.01}) \subset \odot(\text{object.n.01})$ |

Table 1

Child-parent relations are encoded by continuous inclusion relations among spheres. The direct hypernym of a word-sense w is the word-sense x whose sphere is the smallest sphere that covers w 's sphere. $\odot(w)$ represents the sphere of w , $A \subset B$ represents $\text{COV}(B, A)$

| word-sense 1 | word-sense 2 | word |
|--------------|--|---|
| beijing.n.01 | london.n.01, atlanta.n.01, washington.n.01, paris.n.0, potomac.n.02, boston.n.01 | china, taiwan, seoul, taipei, chinese, shanghai, korea, mainland, hong, wen, kong, japan, hu, guangzhou, chen, visit, here, tokyo, vietnam |
| berlin.n.01 | madrid.n.01, toronto.n.01, rome.n.01, columbia.n.03, sydney.n.01, dallas.n.01 | vienna, warsaw, munich, prague, germany, moscow, hamburg, bonn, copenhagen, cologne, dresden, leipzig, budapest, stockholm, paris, frankfurt, amsterdam, german, stuttgart, brussels |
| berlin.n.02 | simon.n.02, williams.n.01, foster.n.01, dylan.n.01, mccartney.n.01, lennon.n.01 | petersburg, rome, austria, bucharest, düsseldorf, zurich, kiev, austrian, heidelberg, london |
| hamburg.n.01 | glasgow.n.01, bristol.n.01, oslo.n.01, santos.n.01, colon.n.04, hull.n.05 | munich, stuttgart, bundesliga, frankfurt, freiburg, bayern, borussia, vfb, fc., germany, werder, bremen, eintracht, berlin |

Table 2

Top-6 sphere nearest neighbors compared with top-N GloVe nearest neighbors. Neighbours of a sphere are strictly constrained by hypernym structures. In our tree structure, 'hamburg.n.01' is the neighbor of other ports. In contrast, GloVe neighbours are biased training corpus, mixing with or neglecting other word-senses. Glove neighbors of 'hamburg' are severely biased to football-related training sentences, neighbors 'beijing' mixes with names of countries and persons, the word-sense of family names of 'berlin' is totally neglected in its neighborhood

itself (condition 5). The *cover* relation is transitive, that is, if \odot_{t_i} covers \odot_{t_j} , and \odot_{t_j} covers \odot_{t_k} , then \odot_{t_i} covers \odot_{t_k} . This follows that the root sphere covers all other spheres.

We adopt depth-first recursive process to traverse the nodes in a tree structure. A parent sphere will be constructed after all its child spheres are constructed, as illustrated in Figure 5(a). Geometric construction is carried out as a sequence of operations selected from three geometric transformations as follows.

1. A Homothetic operation on sphere \odot with the ratio $k(k > 0)$, written as $H(\odot, k)$, zooms out the length of the vector of its central point and the radius with the same ratio k , $H(\odot(O, r), k) = \odot(kO, kr)$, as illustrated in Figure 5(g).
2. A Shifting operation on sphere \odot with vector \vec{v} , written as $S(\odot, \vec{v})$, shifts this sphere with vector \vec{v} , $S(\odot, \vec{v}) = \odot(O + \vec{v}, r)$, as illustrated in Figure 5(h).
3. A Rotation operation rotates sphere $\odot(O, r)$ with unit vector $\vec{\beta}$ in the subspace spanned by the i -th and the j -th basis, written as $R(\odot(O, r), \vec{\beta}, i, j) = \odot(O', r)$, in which $O_k = O'_k$ for all $k \neq i, j$, $O'_i = O_i \cos \beta + O_j \sin \beta$, $O'_j = -O_i \sin \beta + O_j \cos \beta$, as illustrated in Figure 5(i).

4.2. Experiments

GloVe word-embeddings [40] are used as the pre-trained word-embedding, hypernym trees among word-senses are extracted from WordNet 3.0 [37], totaling 291 hypernym trees and 54,310 spheres [12], each representing a word-sense in a hypernym tree. These sphere embeddings have 32,503 word-stems. Precise spatialisation has been achieved, pre-trained GloVe word-embeddings have been very well-preserved. Only a tiny portion (1.3%) of pre-trained word-embeddings indicates a small variation ($\text{std} \in (0.1, 0.7666)$). Symbolic tree structure is precisely embedded onto the continuous space, which leads to precise encoding of category information, as illustrated in Table 1, and precise separation of word-senses in nearest neighborhood experiments, as illustrated in Table 2. In order to prevent the deterioration of already constructed relations, we will apply the same transformation for all its child spheres, if we apply a geometric transformation for a sphere. The recursive geometric construction process will generate a sequence of transformations for the construction of the sphere of a tree node that transform a sphere from its initial status to the final status. This dynamic information can be likened as a route instruction that tells a baby's home address starting from the hospital address where it is born. If we already have the route instruction of its siblings, we can use it to send the new born baby home. Using this idea, we have conducted experiments

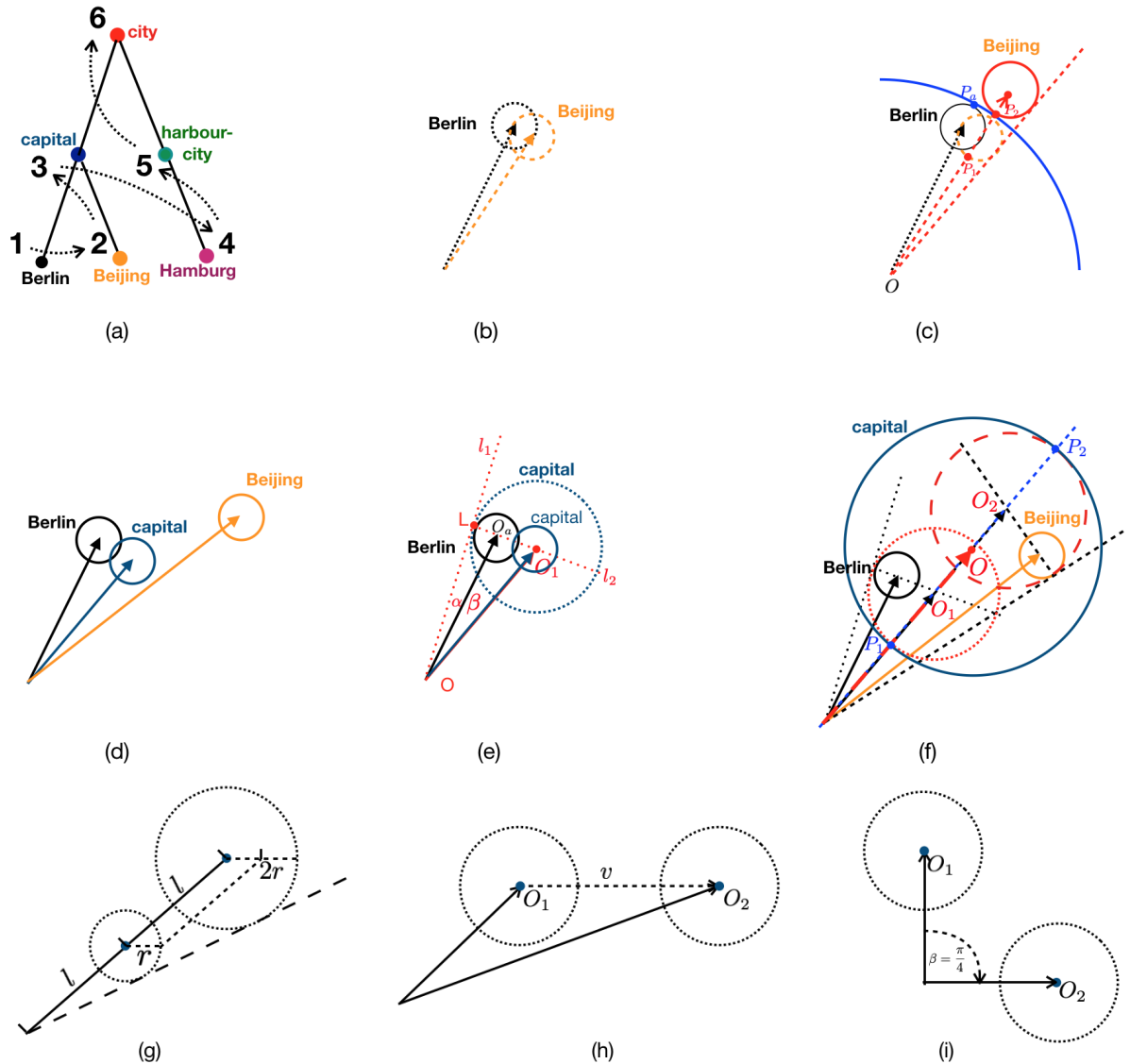


Figure 5: (a) Depth-first sequence to update spheres; (b-c) homothetic transformation will be applied for overlapped sibling spheres; (d-e) construct one *capital* sphere that covers the Berlin sphere; (f) construct the final *capital* sphere that covers all existing *capital* spheres; (g) homothetic transformation with $k = 2$, (h) shifting transformation with \vec{v} , (i) rotation transformation with $\beta = \frac{\pi}{4}$

to validate the category of an unknown word that appears in corpus. For example, when we read a text *Solingen has long been renowned for the manufacturing of fine swords, knives, scissors and razors . . .*, we wonder whether *Solingen* is a *city* or a *person*? Supposing it is a *city*, we initialize its sphere by using the type information of *city*. One of its sibling is *Berlin* that we have its route instruction that guides it into the sphere of *city*. So, we use this route to guide *Solingen*. If it is finally located inside sphere of *city*, we will predict that *Solingen* is a *city*, otherwise not. We have experimented this method to predict the type of unknown entities in knowledge graph [13]. Our experiments show that this geometric approach greatly outperforms traditional embedding approaches, especially when the route is long.

5. Conclusion

The relation between neural networks and symbolic structures remains an open debate. This debate is nothing new and dates back to the antagonism between Connectionism and Symbolicism, e.g., [45]. The difficulty is that precise encoding of symbolic relations cannot be achieved by the back-propagation algorithm – the fundamental algorithm of Connectionism. Here, sphere embeddings are created using geometric construction, and by abandoning back propagation method. Our methodology does not belong to the connectionism paradigm. The created sphere embeddings only exist in a space whose dimension is higher the vector space produced by connectionist networks. This refutes

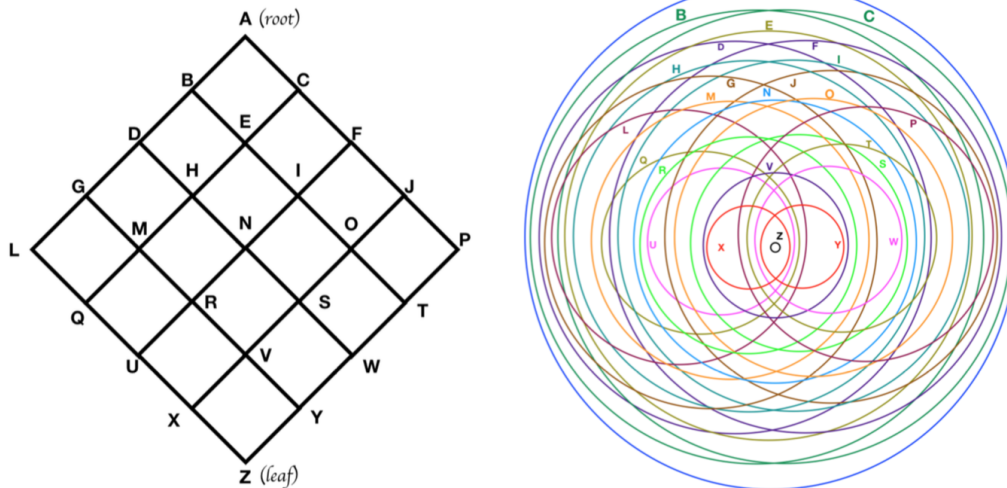


Figure 6: Spatializing a complex lattice

eliminativism, *implementationalism*, and *revisionism*¹, and also refutes the analogy between the symbolic-subsymbolic approaches and the relation between macro–micro physical theories [45]. The sphere embeddings should appeal psychologists, for discrete symbolic tree structures have been precisely transformed into sphere configurations in a continuous space [45, 8]. They also appeal cognitive linguists [30], for now bounded regions and paths can be implemented by geometric transformations.

The nested structure of sphere configurations would solve the *elaboration tolerance* problem² in connectionist networks [34]. The sphere configuration favors both *limitivism* and *hybridism*³, and serves as a big geometric *patchwork* that bridges symbolic and network components, and promises to merge the two complementary methods and theories in cognitive science [39]. For example, the grounding model of metaphor [31, 20], the embodiment of language and thoughts [41, 15], and spatial thinking [55]. It also favors the way of *killing two birds with one stone*⁴[56], in the sense that it is able to precisely reproduce the two birds: subsymbolic vectors, and symbolic structure – It kills the two birds without losing any information about them. The success of the geometric approach largely depends on being able to precisely spatialise more complex symbolic structures onto vector space. Figure 6 shows an on-going work to spatialize a

¹*eliminativism* philosophy holds that connectionist approach can achieve all symbolic approach can do; *implementationalism* philosophy holds that neural-network is the “hardware” of symbolic system; *revisionism* philosophy holds that a symbolic account can be generated by connectionist networks

²*elaboration tolerance* challenges whether connectionist network can be elaborated with additional phenomenon.

³*limitivism* philosophy holds that accurate connectionist account can approximate good symbolic descriptions within certain limit; *hybridism* philosophy holds that a *patchwork* can be created for the gap between symbolic and neural-network components

⁴*killing two birds with one stone* refers to a two-system hypothesis of the mind that the same neural event that is capable of simultaneously manipulate conceptual-level symbols and perform subsymbolic operations

complex grid onto vector space.

Connectionism is not a complete theory for learning [27]. Learning through huge amount of data using back-propagation is *bottom-up* [51] and inefficient, and only establishes a harmony between input and output. An important learning style in schools and universities is learning under instruction (*top-down* style of learning [51]). It is easy to translate “white as snow” into German (“weiß wie Schnee”), French (“blanc comme neige”), and many other languages. How shall we translate it into the Natemba language? People who speak Natemba live in Benin, a country near to the equator, where the temperature is around 20°C in the winter, so no snowing in the winter, as a consequence, no word for snow in the Natemba language. To describe something very white there, people would say “white as pelican” (pelican is a kind of white bird. This is an example of *elaboration tolerance* in translation, see footnote 2). We would feel this translation is interesting and reasonable, after having been informed about the right background knowledge. Methods should be developed to precisely inform (or impose) external knowledge to pure data-driven machine learning systems [42].

Acknowledgement

I am indebted to S.K. Chang for the invitation and his long-term interest in my work, and to Ron Sun for the comments on an early version of this paper. This research is funded by the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R under grant number 01/S18038C.

References

- [1] Bader, S., Hitzler, P., 2004. Logic programs, iterated function systems, and recurrent radial basis function networks. Journal of Applied Logic, Special Issue on Neural-Symbolic Systems 2, 273–300.
- [2] Bader, S., Hitzler, P., 2005. Dimensions of neural-symbolic integration — a structured survey, in: Artemov, S., Barringer, H., Garcez,

- A.S.d., Lamb, L.C., Woods, J. (Eds.), *We Will Show Them: Essays in Honour of Dov Gabbay*. King’s College Publications. volume 1, pp. 167–194.
- [3] Bechtel, W., Abrahamsen, A., 2002. Connectionism and the mind: Parallel processing, dynamics, and evolution in networks. Graphicraft Ltd, Hong Kong.
- [4] Bengio, Y., Ducharme, R., Vincent, P., Janvin, C., 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3, 1137–1155. URL: <http://dl.acm.org/citation.cfm?id=944919.944966>.
- [5] Besold, T.R., d’Avila Garcez, A.S., Bader, S., Bowman, H., Domingos, P.M., Hitzler, P., Kühnberger, K., Lamb, L.C., Lowd, D., Lima, P.M.V., de Penning, L., Pinkas, G., Poon, H., Zaverucha, G., 2017. Neural-symbolic learning and reasoning: A survey and interpretation. CoRR abs/1711.03902. URL: <http://arxiv.org/abs/1711.03902>, arXiv:1711.03902.
- [6] Blackburn, P., 2000. Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto. *Logic Journal of the IGPL* 8, 339–625.
- [7] Chalmers, D.J., 1992. Subsymbolic Computation and the Chinese Room, in: *The Symbolic and Connectionist Paradigms: Closing the Gap*, Erlbaum. pp. 25–48.
- [8] Dellarosa, D., 1988. The psychological appeal of connectionism. *Behavioral and Brain Sciences* 1, 28–29.
- [9] Dinsmore, J., 1992. Thunder in the Gap, in: *The Symbolic and Connectionist Paradigms: Closing the Gap*, Erlbaum. pp. 1–23.
- [10] Dong, H., Mao, J., Lin, T., Wang, C., Li, L., Zhou, D., 2019a. Neural Logic Machines, in: *ICLR-19*, New Orleans, USA.
- [11] Dong, T., 2021. A Geometric Approach to the Unification of Symbolic Structures and Neural Networks. volume 910 of *Studies in Computational Intelligence*. Springer-Nature.
- [12] Dong, T., Bauckhage, C., Jin, H., Li, J., Cremers, O.H., Speicher, D., Cremers, A.B., Zimmermann, J., 2019b. Imposing Category Trees Onto Word-Embeddings Using A Geometric Construction, in: *ICLR-19*, New Orleans, USA. May 6-9.
- [13] Dong, T., Wang, Z., Li, J., Bauckhage, C., Cremers, A.B., 2019c. Triple Classification Using Regions and Fine-Grained Entity Typing, in: *AAAI*.
- [14] Faruqui, M., Dodge, J., Jauhar, S.K., Dyer, C., Hovy, E., Smith, N.A., 2015. Retrofitting word vectors to semantic lexicons, in: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, ACL. pp. 1606–1615. URL: <http://www.aclweb.org/anthology/N15-1184>, doi:10.3115/v1/N15-1184.
- [15] Feldman, J., 2006. *From Molecule to Metaphor: A Neural Theory of Language*. The MIT Press, Cambridge, Massachusetts.
- [16] Feldman, J., 2013. The neural binding problem(s). *Cogn Neurodyn* 7, 1–11.
- [17] d’Avila Garcez, A.S., Lamb, L.C., Gabbay, D.M., 2007. Connectionist modal logic: Representing modalities in neural networks. *Theor. Comput. Sci.* 371, 34–53.
- [18] Gärdenfors, P., 2000. *Conceptual Spaces – The Geometry of Thought*. MIT Press, Cambridge, Massachusetts, USA.
- [19] Gärdenfors, P., 2017. *The Geometry of Meaning*. MIT Press, Cambridge, Massachusetts, USA.
- [20] Grady, J., 1997. *Foundations of Meaning: Primary Metaphors and Primary Scenes*. University Microfilms.
- [21] Hammer, B., Hitzler, P. (Eds.), 2007. *Perspectives of Neural-Symbolic Integration*. Springer.
- [22] Han, X., Liu, Z., Sun, M., 2016. Joint representation learning of text and knowledge for knowledge graph completion. CoRR abs/1611.04125. arXiv:1611.04125.
- [23] Harnad, S., 1990. The symbol grounding problem. *Phys. D* 42, 335–346.
- [24] Harnad, S., 2003. The symbol grounding problem, in: *Encyclopedia of Cognitive Science*, Macmillan.
- [25] Hinton, G.E., 1981. Implementing semantic networks in parallel hardware, in: Hinton, G.E., Anderson, J.A. (Eds.), *Parallel Models of Associative Memory*. Erlbaum, Hillsdale, NJ, pp. 161–187.
- [26] Hinton, G.E., 1986. Learning distributed representations of concepts, in: *Proceedings of the eighth annual conference of the cognitive science society*, Amherst, MA. p. 12.
- [27] Hunter, L.E., 1988. Some memory, but no mind. *Behavioral and Brain Sciences* 1, 37–38.
- [28] Jia, R., Liang, P., 2017. Adversarial examples for evaluating reading comprehension systems, in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, Copenhagen, Denmark, September 9-11, 2017, pp. 2021–2031.
- [29] Lake, B., Salakhutdinov, R., Tenenbaum, J., 2015. Human-level concept learning through probabilistic program induction. *Science* 350, 1332–1338.
- [30] Lakoff, G., 1988. Smolensky, semantics, and the sensorimotor system. *Behavioral and Brain Sciences* 1, 39–40.
- [31] Lakoff, G., Johnson, M., 1980. *Metaphors We Live By*. The University of Chicago Press, Chicago. Citation is based on the reprinted in 2003.
- [32] LeCun, Y., Bengio, Y., Hinton, G.E., 2015. Deep learning. *Nature* 521.
- [33] Levinson, S.C., 1997. Language and Cognition: The Cognitive Consequences of Spatial Description in Guugu Yimithirr. *Journal of Linguistic Anthropology* 7, 98–131.
- [34] McCarthy, J., 1988. Epistemological challenges for connectionism. *Behavioral and Brain Sciences* 1, 44.
- [35] McNamara, T.P., 1991. Memory’s View of Space. *The Psychology of Learning and Motivation* 27, 147–186.
- [36] Mikolov, T., 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis. Brno University of Technology. Brno, CZ.
- [37] Miller, G.A., 1995. Wordnet: A lexical database for english. *Commun. ACM* 38, 39–41.
- [38] Newell, A., 1990. *Unified theories of cognition*. Harvard University Press, Cambridge, MA.
- [39] Núñez, R., Allen, M., Gao, R., Rigoli, C.M., Relaford-Doyle, J., 2019. What happened to cognitive science? *Nature Human Behaviour* 3, 782–791.
- [40] Pennington, J., Socher, R., Manning, C.D., 2014. GloVe: Global Vectors for Word Representation, in: *EMNLP’14*, pp. 1532–1543.
- [41] Regier, T., 1997. *The Human Semantic Potential: Spatial Language and Constrained Connectionism*. The MIT Press, Cambridge, Massachusetts.
- [42] von Rüden, L., Mayer, S., Garcke, J., Bauckhage, C., Schücker, J., 2019. Informed Machine Learning - Towards a Taxonomy of Explicit Integration of Knowledge into Machine Learning. CoRR abs/1903.12394. URL: <http://arxiv.org/abs/1903.12394>.
- [43] Russell, B., 1956. *Logic and Knowledge*. Routledge, an imprint of Taylor & Francis Books Ltd.
- [44] Seo, M.J., Kembhavi, A., Farhadi, A., Hajishirzi, H., 2016. Bidirectional attention flow for machine comprehension. CoRR abs/1611.01603. URL: <http://arxiv.org/abs/1611.01603>, arXiv:1611.01603.
- [45] Smolensky, P., 1988. On the proper treatment of connectionism. *Behavioral and Brain Sciences* .
- [46] Socher, R., Chen, D., Manning, C.D., Ng, A., 2013. Reasoning with neural tensor networks for knowledge base completion, in: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (Eds.), *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., pp. 926–934.
- [47] Speer, R., Chin, J., Havasi, C., 2017. Conceptnet 5.5: An open multilingual graph of general knowledge, in: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, February 4-9, 2017, San Francisco, California, USA., pp. 4444–4451.
- [48] Stevens, A., Coupe, P., 1978. Distance estimation from cognitive maps. *Cognitive Psychology* 13, 526–550.
- [49] Sun, R., 2002. Hybrid connectionist symbolic systems, in: Arbib, M. (Ed.), *Handbook of Brain Theories and Neural Networks (2nd Edition)*. MIT Press, Cambridge, MA., pp. 543–547.
- [50] Sun, R., 2015. Artificial Intelligence: Connectionist and Symbolic Approaches, in: James, D.W. (Ed.), *International Encyclopedia of*

the Social and Behavioral Sciences (2nd edition). Pergamon/Elsevier, Oxford, pp. 35 – 40.

- [51] Sun, R., 2016. Implicit and explicit processes: Their relation, interaction, and competition, in: Macchi, L., Bagassi, M., Viale, R. (Eds.), *Cognitive Unconscious and Human Rationality*. MIT Press, Cambridge, MA., pp. 257–27.
- [52] Tversky, A., 1977. Features of similarity. *Psychological Review* 84, 327–353.
- [53] Tversky, A., Kahneman, D., 1974. Judgment under uncertainty: Heuristics and biases. *Science* 185, 1124–1131.
- [54] Tversky, B., 1981. Distortions in Memory for Maps. *Cognitive Psychology* 13, 407–433.
- [55] Tversky, B., 2019. *Mind in Motion*. Basic Books, New York, USA.
- [56] Woodfield, A., Morton, A., 1988. The reality of the symbolic and subsymbolic systems. *Behavioral and Brain Sciences* 1, 58.
- [57] Xiao, H., Huang, M., Zhu, X., 2016. From one point to a manifold: Knowledge graph embedding for precise link prediction, in: *IJCAI*.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jv1c

A Comprehensive Analysis and Design of Land Cover Usage from Satellite Images using Machine Learning Algorithms

Rajeswara Rao Duvvada¹, Shaik Ayesha Fathima², Shaik Noorjahan³

1,2,3 Department of Computer Science & Engineering, VR Siddhartha Engineering College, India
1rajeshpitam@gmail.com, 2shaikfathima579@gmail.com, 3shaiknoorjahan2001@gmail.com

ARTICLE INFO

Article History.

Submitted: 2.9.2022

Revised: 5.9.2022

Accepted: 5.23.2022

Keywords:

Land Cover

Classification

DeepLabv3Plus

Deep Globe Land

Cover Classification

UNet

FPN

ABSTRACT

Real-time satellite images provide a true representation of Earth and its environment. Satellite images provide important remote sensing data that can be used in a variety of applications such as image fusion, change detection, land cover classification, agriculture, mining, disaster mitigation planning, and monitoring climate change. This project performs land cover classification, that is classification of satellite images into multiple predefined classes by means of several algorithms. In this study, we compared the accuracies of different segmentation models such as Unet, FPN, deeplabv3Plus with different encoders such as Resnet101, Resnet50, Resnet34 with different activation functions, such as softmax and sigmoid functions. Following the classification of satellite images, we will calculate the area of each land cover class. The dataset used here is the Deep Globe Land Cover Classification dataset. It contains seven pre-defined classes. The approach followed for each algorithm involves gathering and processing satellite imagery. The preliminary images are then preprocessed with data preprocessing techniques and the preprocessed images are fed into the appropriate algorithm. The obtained result will be then analyzed. After measuring the accuracy of each algorithm, the algorithm that gives the best results is identified based on the comparison of the accuracy obtained from each algorithm. Among the three segmentation models, DeepLabV3Plus segmentation model with resnet101 encoder gave more accuracy i.e, 90.17% when compared two remaining UNet, FPN segmentation models

1. Introduction

Many visual applications now use segmented images to get a deeper understanding of a scene. Segments or objects are created by segmenting video frames or images and, consequently, take on an important role in real-life applications. Examples include optical remote sensing, facial segmentation, autonomous driving, and computational photography. Literature has reviewed various techniques for image segmentation, including region-growing, thresholding, watersheds, Otsu, and K-means clustering. Also discussed are graph cuts and Markov random fields. Most of these older techniques utilize low-level features and cues to segment objects.

There are several neural network-based methods for detecting and classifying objects which have achieved notable success in recent years, showing improvement in terms of performance accuracy and performance time. A growing number of deep learning-based image segmentation models are being developed recently. These models can be used to segment a number of objects including humans, cars, horses, trees and the sky. In deep learning-based segmentation, most models recognize and characterize objects based on many images during the learning process. These models are developed primarily for identifying objects in frontal views.

In our previous study [1] we used Deeplabv3Plus with resnet50 encoder as our model to perform land cover

classification in satellite imagery and now in this paper we are doing comparative analysis using different deep learning segmentation models, DeepLabV3, Unet, FCN for land cover classification. The proposed methodology will classify the images into predefined classes such as barren land, range land, forest land, agriculture land, water, urban land with different RGB values. Then we calculate the area of each predefined class based on the pixel count obtained from `count_nonzero()` method of numpy module. Deep Globe Land Cover Classification Dataset is the dataset used for this study as shown in figure 1. We calculate the accuracy of each segmentation model and identify the model that gives best accuracy.

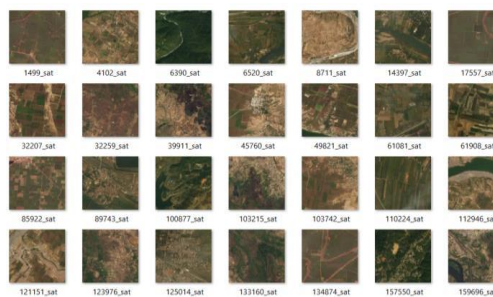


Figure 1: Sample image of the dataset [16]

1.1 DeepLabV3Plus Segmentation model

Assigning semantic labels to each pixel in a digital image is the goal of DeepLabv3+, a state-of-the-art deep learning model for semantic segmentation. It has been improved several times since it was first open sourced by Google in 2016; the latest version is DeepLabv3+ which has an encoding phase and decoding phase. Encoding an image employs convolutional neural networks in order to collect the essential information, whereas decoding reconstructs the output of desired dimensions using the acquired information. The DeepLab decoder modules support better segmentation along object boundaries using the Xception network backbone for training our DeepLab model. As an example, Google's Pixel smartphone is supposed to perform various image segmentation tasks using the trained model from DeepLab. DeepLab supports the following network backbones: MobileNetv2, Xception, ResNet, PNASNet, Auto-DeepLab.

1.2 Unet Segmentation Model

U-Nets are architectures that allow for semantic segmentation. They are constructed from a contracting path (sometimes called an encoder) and an expanding path (sometimes called a decoder). Typical convolutional network architecture is followed by the contracting path. A 2x2 max pooling step with stride 2 for downsampling is applied as well as two 3x3 convolutions (unpadded convolutions) repeatedly. By downsampling each feature channel twice, the number of channels is doubled. In the expanding path, upsampling is performed at every step, followed by a 2x2 convolution to halve the number of features channels, concatenation with a cropped map of all the features from the contracting path, and two 3x3 convolutions, each followed by a ReLU. In order to prevent the loss of border pixels, you must crop all of the convolutions. The final layer of the algorithm uses a 1x1 convolution to map each 64-component feature vector to the desired number of classes. 23 convolution layers make up the network.

1.3 FPN Segmentation Model

With the help of fully-convolutional algorithms, the Feature Pyramid Network, or FPN, extracts features from any image at any scale. This approach, which creates feature pyramids without using the backbone convolutional architectures, could be used for object detection despite not being convolutional. Using this general solution, feature pyramids can be built in deep convolutional networks. Pyramids are constructed in two ways: top-down pathways and bottom-up pathways.

In bottom-up approaches, a feature hierarchy is derived by combining features at two scale levels using backbone ConvNets. For each stage of the feature hierarchy, feature pyramid levels are defined. The last layer of each stage's feature pyramid constitutes a reference set of feature maps. Feature activation maps generated by the final residual block of each stage are used for ResNets.

By upsampling more coarsely detailed, yet more semantically rich, feature maps at higher pyramid levels, and then combining them with features from the bottom-up pathway along lateral axes, the top-down pathway enhances lower resolution features. In contrast to the top-down pathway, which affects more regions, the bottom-up pathway was sampled less frequently, so its activations are more accurately located but the top-down pathway affects more regions. From the bottom-up pathway and from the top-down pathway, each lateral connection merges feature maps with the same spatial size.

1.4 Resnet

Residual neural networks (ResNet) consist of artificial neural networks that use shortcuts or skip connections, allowing them to jump over certain layers. In a typical ResNet model, there are two or three layers of skips with nonlinearities (ReLU) and batch normalization between them. A model known as HighwayNet uses a weight matrix in addition to skip weights in order to learn the skip weights. DenseNets are network models that have multiple parallel skips. New networks are non-residual networks that do not have parallel skips. In a suitably deep model, adding more layers can lead to a higher training error if it is subject to Degradation (accuracy saturation). The latter can be mitigated by adding skip connections. With training, the weights augment the previously-skipped layer by muting the upstream layer. Resnet 34, Resnet 50, Resnet 101 are the encoders used in this work.

1.5 Sigmoid Function

In mathematics, the sigmoid function, which is similar to a logistic function, a hyperbolic tangent, and an arctangent, is a function with a typical S-shaped curve. Logistic functions, also known as logistic sigmoid functions, are used in machine learning. Logistic functions are defined as follows:

$$S(x) = \frac{1}{1+e^{-x}}$$

The logistic function takes value between zero and one for both input and output.

1.6 Softmax Function

In mathematics, a Softmax is a function that converts numbers/logits into probabilities. Its output is a vector (we'll call it v) with probabilities of everything that can possibly happen. These probabilities are added for all possible outcomes or classes. Mathematically, Softmax is defined as,

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^n \exp(y_j)}$$

2. Literature Survey

Rahul Neware [2] used three categories of algorithms: unsupervised, supervised and object-based classification. Supervised classification is a machine learning technique where inputs are given, and then classification is made

according to the inputs. Unsupervised classification largely relies on cluster analysis method. To perform cluster analysis, pixels are segmented into numerous classes and grouped according to their common characteristics. Object-oriented classification is specifically applied to high resolution imagery. Objects are mainly segmented from satellite images in object-oriented classification. In supervised classification of high resolution multispectral LISS-IV images, two methods are employed: Maximum Likelihood (ML) and Support Vector Machine (SVM). The obtained optical image of the study area will be subjected to supervised classification methods of various algorithms. Obtain the optical image of the study area based on the LISS-IV measurement. Obtain the optical image of the study area based on the LISS-IV measurement. Subsequently, rectify the obtained optical image of the study area. Create a graphical representation of each band based on the classified image as output. This paper concludes by providing a step-by-step procedure for optical data classification, concluding that SVM classification was superior for optical data classification compared to ML classification. It proposes a step-by-step procedure for optical image classification. Efficient comparative analysis is made between SVM classification and ML classification. SVM classification gives a better result than ML classification technique with 77.5% accuracy.

Keerti Kulkarni, P. A. Vijaya [3] performed a comparative analysis of land classification is performed utilizing several land classification algorithms which are available in remote sensing methods, such as Maximum Likelihood, Multi-Label Classification (MLC) and Minimum Distance. Minimum distance classification is done by defining two classes and then calculating the mean vector for each class. As new objects (pixels) are classified, they are compared to the nearest mean vector calculated from those of known classes. The closest mean vector is then used to classify the new objects. Euclidian distance, Normalized Euclidian distance, Mahalanobis distance these distances are often used in this procedure. Having the minimum distance equal to the maximum similarity as the index of similarity. Maximal likelihood classification's uses a probability based decision rule. The maximum likelihood procedure assumes that training classes in each band have a normal distribution (Gaussian distribution). An assignment of pixels to the class in which they have the highest probability is made in this paper. Calculating the probability assigned to each pixel assigns the pixels to the class with the highest probability. In traditional image classification, mixed pixels can cause problems because they belong to multiple classes but can only be assigned to one class at a time. Therefore, Multilabel Classification makes use of Spectral Mixture Analysis in order to estimate the likely composition of each pixel of the image. The methods used are Problem Transformations and Algorithm Adaptation Methods. It is concluded that multi-label method classifier would produce better results for hyperspectral remote sensing images and would be more appropriate for them. It is highly effective in overcoming the underlying dependency between labels. Multilabel classification gives a more natural representation, reduces complexity, and speeds up processing time.

Vineetha P., Smitha Asok Vijayamma ., Sandra Yesudasan Miranda [4] assessed Both the unsupervised image

classification accuracy and the supervised classification-based predicted land use accuracy are evaluated with Kappa coefficients. Classification of images are undertaken in order to determine land use analysis for the selected area. LISS-IV 2015 image was categorized into six types of unsupervised classification classes: built up, paddy, water body, vegetation, plantation and scrub. Both the supervised and unsupervised classification methods utilized Maximum Likelihood Classification. The classes were resolved using the training sets from Google Earth that are selections of pixels which display a similar pattern. The supervised classification-based images were then recoded to produce the desired classes. For the training set identification stage, LISS-IV 2015 imagery was classified unsupervised using ArcGIS.

The resulting vectors were then transformed into maps. ArcGIS was used to convert this vector form into KML format and Google Earth was used to view the training sets for different classes, then the supervised classification was performed. The results of the study showed a moderate degree of accuracy in the classification of different land use categories based on area-based details. This study analyzed both supervised and unsupervised methods of remote sensing data classification to determine land use in sacred groves in Thiruvananthapuram district. The overall accuracy obtained was 69.09 %. A database of the types of land uses in these areas was also created as part of the study.

K. S. Ravichandran ; R. Sivagami; R. Krishankumar; [5] aimed at analysing the performance of different supervised learning algorithms to label pixels in images using the semantically labelled datasets obtained from (ISPRS). The workflow for every algorithm involves taking input data and obtaining the respective groundtruth image, generating the feature vector, and then obtaining a labeled thematic output using a classifier and finally assessing accuracy. For this comparative study, SVM based classification was used with Gaussian kernels with different kernel scale values which has been used in linearly non-separable data classification. With this approach, training data with associated class labels is used along with the attribute selection measure, in order to construct a top-down recursive divide and conquer approach to decision tree classification. This study uses the Gini index as an attribute selection measure among Information Gain, Gain ratio, and Gini index. Among the three attributes selection measures, the Gini index is chosen. KNN classification involves selecting the training and testing data and then selecting the distance metric. The best k value is detected and the knn model accuracy is evaluated after it has been built. Comparison of the results from the three state of the art techniques led to a conclusion that fine Gaussian support vector machines outperformed them all, with a classification accuracy of approximately 75.1448%. Workflow of every algorithm is same and easy to perform. Comparative analysis is made between SVM Classification, KNN Classification and Decision Tree Classification.

3. Proposed System

3.1 Dataset

Deep globe land cover classification dataset:

- The dataset we used for this work is Deep Globe Land cover classification dataset which consists of nearly 1700 satellite images including masks shown in Figure 2.
- Resolution of each satellite image is 2448 x 2448 and training data consists 803 satellite images with pixel resolution of 50cm by DigitalGlobe satellite.
- The dataset consists of validation images(171) and test images(172) in total excluding mask images.

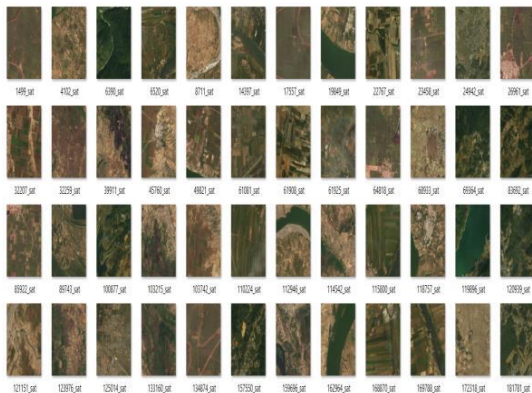


Figure 2: Deep globe land cover classification dataset [16]

3.2 Segmentation Models

In this paper we are using three segmentation models. They are Unet, FPN, DeepLabV3Plus.

Unet Segmentation Model:

U-Net is based on the idea of FCN segmentation. The U-Net architecture could be described as an encoder-decoder unit, divided into three parts. As shown in Figure 3, the down-sampling path is composed mainly of seven stages. The first stage consists of eight stages primarily applying three 3 x 3 convolutions with batch norms, followed by two 2 x 2 max-poolings. This horizontal bottleneck consists of two layers 3 x 3 convolution, followed by two layers 2 x 2 Up-Convolution, as illustrated in the Figure 3. Similarly, the upsampling path consists of four stages, each involving halving the features maps. There are two convolutional layers 3 x 3 representing a decoder, followed by two layers 2 x 2.

Figure.3 shows how the model skips paths between upsampling and downsampling paths in order to provide local and global information during upsampling. Last but not least, the convolutional layer provides segments at output 1 x 1 where the number of feature maps corresponds to the number of segments to be created. In the creation of the Unet segmentation model, we used Resnet34 as an encoder, softmax as an activation function, and sigmoid as an activation function.

FPN Segmentation Model:

Figure 4 shows a diagram of FPN. The system is composed of a top-down and bottom-up approach. We use Pretrained ResNet50 as a feature encoder for the bottom-up pathway of

convolutional network extraction. Each module contains many convolution layers (i=1 to 5) and is composed of many convolution modules. During the top-down pathway, each convolutional module's output is labeled as C_i and then used in the top-down pathway. The spatial dimension reduces by 1/2 with each step up (in other words, it expands by double).

With each layer removed, the semantic value increases. With each step of the bottom-up pathway, the spatial resolution will decrease. In the top-down pathway, we apply a 1x1 convolution filter to reduce C_5 channel depth to 256 so that we can create M_5 . Using two successive layers of 3x3 convolution, we create P_5 ; this is the first feature map layer that is needed to segment objects. P_5 has the same spatial resolution as conv5 and is a 128-channel layer. Then, we apply successively three times 3x3 convolution to output the next layer of features. This reduces the aliasing of the upsampling process. To reduce aliasing of the upsampling process we apply successively three 3x3 convolutions. This reduces the effect of the upsampling process.

By the end of the process, all P_i modules with 128 channels and 1/4 of the input image resolution have been concatenated. The next step is to apply 512 3x3 convolutional filters, batch normalization, and ReLU activation. Our output feature map consists of seven classes, so our output channel map consists of seven classes. Using convolutions again, we reduced the number of channels to acquire seven output channels. A spatial dropout operation is then performed and then the output image is upscaled by bilinear interpolation. to the original image size.

DeepDeepLabV3Plus Segmentation Model:

As shown in Figure.5, this model primarily relies on an encoder-decoder architecture. When encoding, features are encoded by the CNN model, then decoded by the decoder. The encoder extracts all the essential information from each channel separately and combines the information with point-wise convolutions to reconstruct the output, as shown in Figure 5. By using the spatial pyramid pooling (SPP) method, DeepLab divides feature maps separated by convolutional layers into spatial bins of a fixed number of bins. DeepLabV3 uses an array of convolutions with SPP to process images with multiple scales and to add context without requiring as many parameters as possible. This increases the field of view as opposed to the previous models that used max-pooling and batch normalization. The new model presented by uses more layers of Xception backbone with depth-wise dilated separable convolutions instead of using the previous models. FIGURE 5 illustrates the encoder module, which encodes multi-scale contextual information with the aid of atrous convolution, and FIGURE 6 shows the decoder module, which refines the segmentation results by finely tuning the boundary conditions around each object. With our atrial convolution model, we can acquire multi-scale information by adjusting filters. With our atrial convolution model, the features are resolved by using atrial convolution.

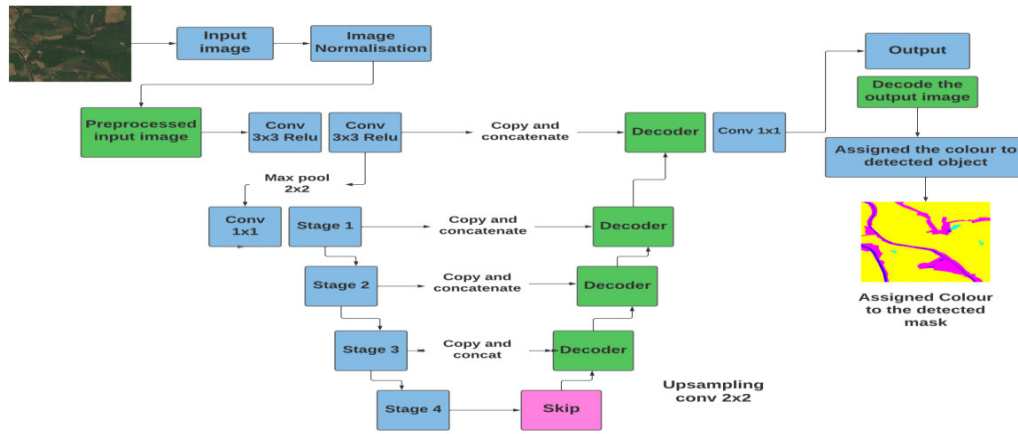


Figure 3: Unet Segmentation Model Architecture



Figure 4: FPN Segmentation Model Architecture

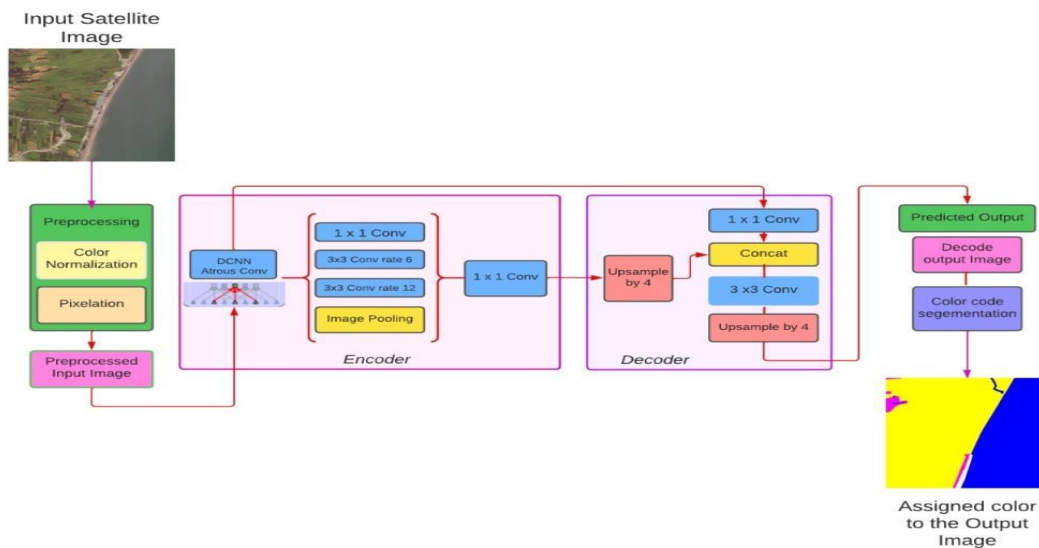


Figure 5: DeeplabV3Plus Segmentation Model Architecture

3.3 Proposed Methodology:

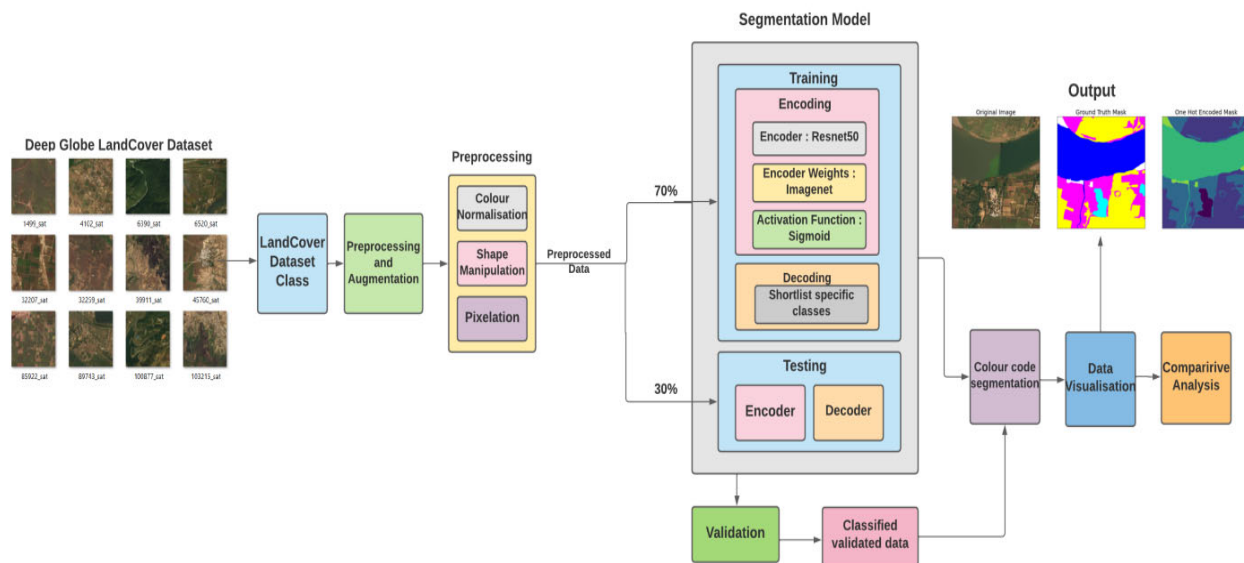


Figure 6: Flow Diagram for our approach

- Firstly we define LandCoverDataset class in which we read satellite images and then we apply augmentation on this images to create new copies of existing data using center_crop() and Random_crop() methods to increase the size of the dataset.
- We perform data preprocessing on our satellite images using Color Normalization, Shape Manipulation, pixelation.
- Color normalization is used to normalize the rgb values in satellite images on the basis of pixel values, Shape Manipulation is to resize the satellite images in our dataset to unified dimensions and apply Pixelation to not exceed the original or actual dimensions of our satellite images .

Segmentation Model:

We divide our dataset in the ratio 7:3 for training and testing 70% of the data is used to train our segmentation models and 30% of the data is used for testing.

Create Segmentation model with pretrained encoder:

Create Unet Segmentation model

- ‘Resnet34’ is used as encoder, ‘imagenet’ as encoder- weights and activation functions as sigmoid and softmax.
- From ‘segmentation_models_pytorch’ module we used Unet() method.
- Perform training and testing of the model.

Create FPN Segmentation model

- ‘Resnet50’ is used as encoder, ‘imagenet’ as encoder- weights and activation functions as sigmoid and softmax.
- From ‘segmentation_models_pytorch’ module use FPN() method.
- Perform training and testing of the model.

Comparison of Unet, FPN and DeepLabV3Plus Segmentation Models:

- Compare the accuracies of segmentation models with different encoders and different activation functions.
- Determine the best segmentation model.

Decoding:

- In decoding we Shortlist specific classes to segments using ‘tolist()’ and ‘values.tolist()’ methods to get the classes and values.

Testing the model:

- In order to test the model test dataloader is created for each segmentation model.
- We display the results of testing using Data visualization method.

Color code Segmentation:

- We perform color code segmentation by coloring the classes with their corresponding rgb values which are mapped to their class keys.

Data Visualization:

- In data visualization we display our original satellite images, classified image, converted image.
- Classified image is obtained from color_code_segmentation() method and converted image is obtained from reverse_one_hot() method.
- In order to plot the results we use matplotlib module.

Area Calculation:

- We perform area calculation of each class by obtaining the pixel count of each predefined class in that satellite image using ‘count_nonzero()’ method of numpy module.

4. Visualization Results Of Segmentation Models For Satellite Images

For each segmentation model we give original image, classified image and converted image. We classify the satellite images into predefined multiple classes such as Urban Land, Agriculture Land, Range Land, Forest Land, Water, Barren Land, Unknown with rgb values 0,255,255; 255,255,0; 255,0,255; 0,255,0; 0,0,255; 255,255,255; 0,0,0 respectively and with colors cyan/aqua, yellow, pink, green, blue, white, black respectively.

Unet Model with Resnet 34 encoder and sigmoid activation function:

Figure 7 depicts the data visualization output of Unet Model. The Encoder used is Resnet34 and the activation function is sigmoid activation function.

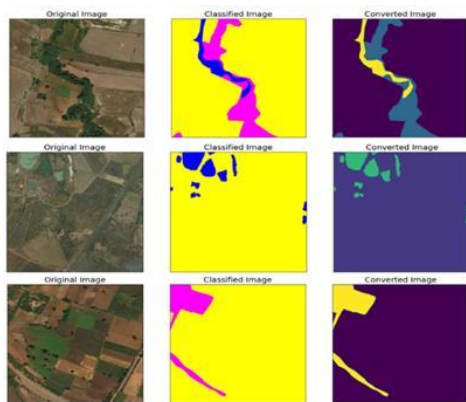


Figure 7: Unet Model with Resnet34 encoder and sigmoid activation function

Unet Model with Resnet 34 encoder and softmax activation function:

Figure 8 depicts the data visualization output of Unet Model. The Encoder used is Resnet34 and the activation function is softmax activation function.

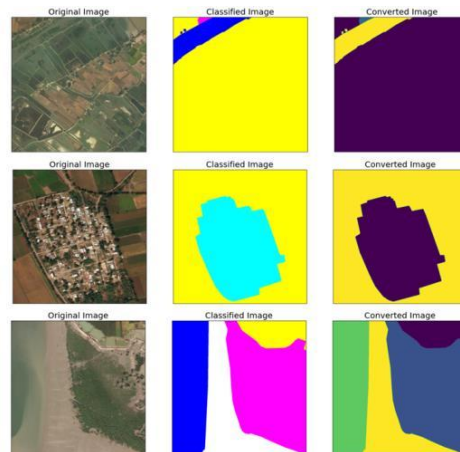


Figure 8: Unet Model with Resnet 34 encoder and softmax activation function

FPN Model with Resnet 50 encoder and softmax activation function:

Figure 9 depicts the data visualization output of FPN Model. The Encoder used is Resnet50 and the activation function is softmax activation function.

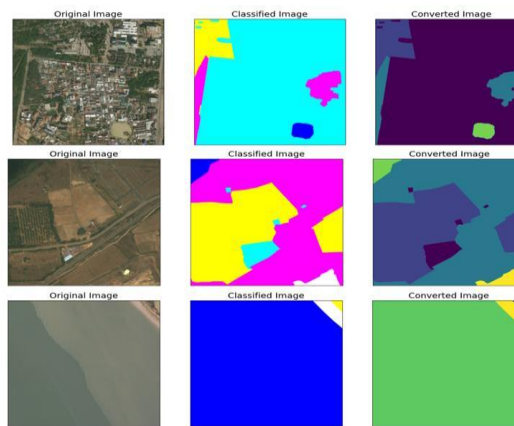


Figure 9: FPN Model with Resnet 50 encoder and softmax activation function

FPN Model with Resnet 50 encoder and sigmoid activation function:

Figure 10 depicts the data visualization output of FPN Model. The Encoder used is Resnet50 and the activation function is sigmoid activation function.



Figure 10: FPN Model with Resnet 50 encoder and sigmoid activation function

DeepLabV3Plus with resnet101 encoder:

Figure 11 depicts the data visualization output of DeepLabV3Plus Model. The Encoder used is Resnet101 and the activation function is sigmoid activation function.

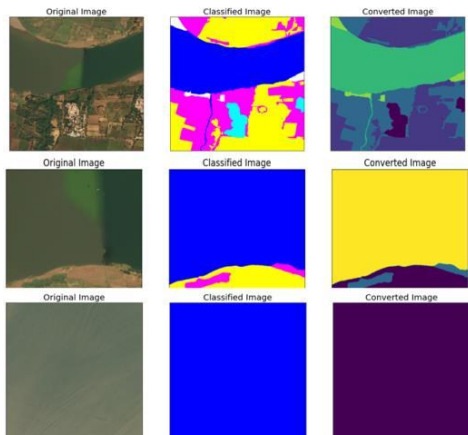


Figure 11: DeepLabV3Plus with resnet101 encoder

Performance Evaluation

IOU Score:

A method's ability to predict bounding boxes based on a set of input data is measured by this statistic. Methods for generating anticipated bounding boxes can be evaluated using this calculator. The difference between the expected and ground truth bounding boxes is represented by this percentage between 0 and 1.

$$IOU = \frac{\text{Area of overlap}}{\text{Area of Union}}$$

Unet Model:

IOU Scores of Unet segmentation model with sigmoid and softmax activation functions for different epochs are plotted in Figure 12.

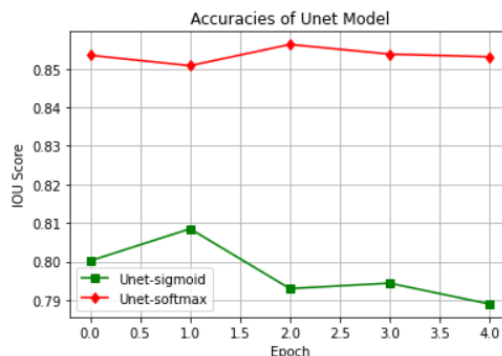


Figure 12: Accuracies of Unet Model

FPN Model:

IOU Scores of FPN segmentation model with sigmoid and softmax activation functions for different epochs are plotted in Figure 13.

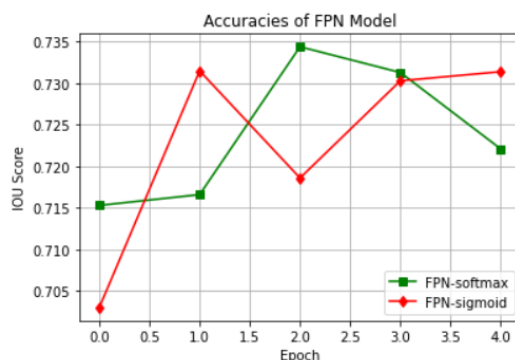


Figure 13: Accuracies of FPN Model

IOU Scores of DeepLabV3Plus segmentation model with resnet50 and resnet101 encoders for different epochs are plotted in Figure 14.

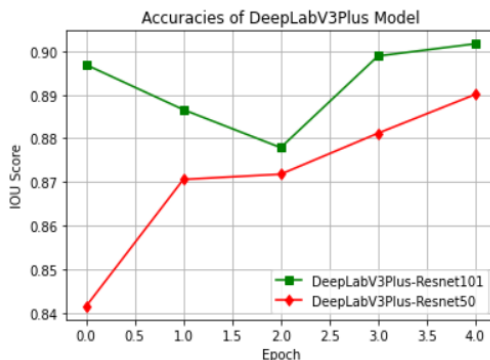


Figure 14 : Accuracies of DeepLabV3Plus Model

IOU scores of the 3 different segmentation models are as follows,

| S. No | Segmentation Model | Encoder | Activation Function | M-IOU Score |
|-------|--------------------|-----------|---------------------|-------------|
| 1 | Unet | Resnet34 | Sigmoid | 0.7890 |
| 2 | Unet | Resnet34 | Softmax | 0.8531 |
| 3 | FPN | Resnet50 | Sigmoid | 0.7314 |
| 4 | FPN | Resnet50 | Softmax | 0.7221 |
| 5 | DeepLabV3Plus | Resnet50 | Sigmoid | 0.8713 |
| 6 | DeepLabV3Plus | Resnet101 | Sigmoid | 0.9017 |

Table 2 : Mean IOU Scores of Segmentation Models

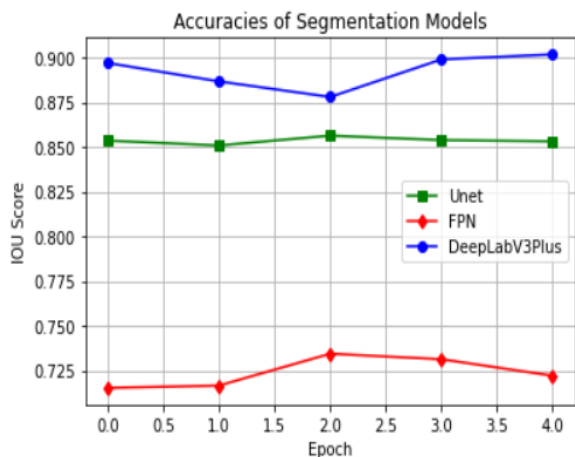


Figure 15: Accuracies of Segmentation Models

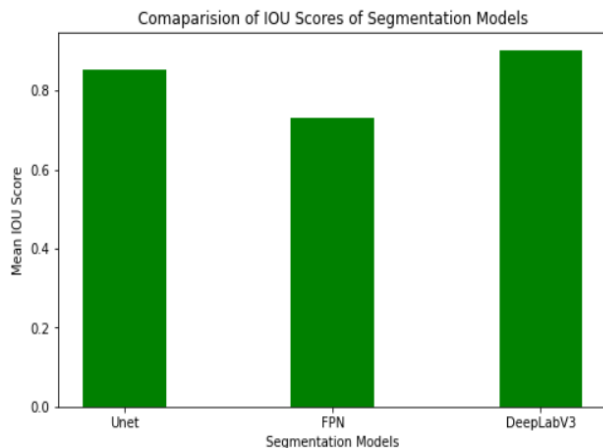


Figure 16: Comparison of IOU Scores of Segmentation Models

Among the three segmentation models, DeepLabV3Plus gives more iou score with resnet101 encoder. Unet gives more iou score with softmax function than with sigmoid function. Similarly, FPN also gives more iou score with softmax function than with sigmoid function. In case of Unet, resnet34 encoder is used whereas for FPN resnet50 encoder is used. DeepV3Plus gives more iou score with resnet101 encoder than with resnet50 encoder [1].

Dice Loss:

In the 1940s, Srensen-Dice created a statistic to measure how similar two samples are. It is named dice loss after the Srensen-Dice coefficient. A 3D segmentation technique for medical images was introduced by Milttari et al. in 2016 for computer vision research.

$$D = \frac{2 \sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

Pi and gi, as seen above, are Dice coefficients which represent pixels that are both associated with prediction and ground truth, respectively. The boundaries are detected by either displaying pi and gi values of 0 or 1, which indicate whether the pixel is a boundary (value 0) or not. If pi and gi are the same, when the sum advances, the numerator is the

sum of successfully predicted boundaries, while the denominator is the sum of ground truth boundaries (both 1 in value).

Unet Model:

Dice Loss of Unet segmentation model with sigmoid and softmax activation functions for different epochs are plotted in Figure 17.

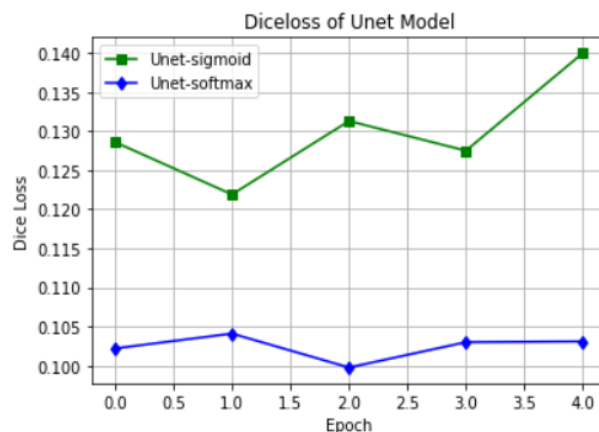


Figure 17: Dice Loss of Unet Model

FPN Model:

Dice Loss of FPN segmentation model with sigmoid and softmax activation functions for different epochs are plotted in Figure 18.

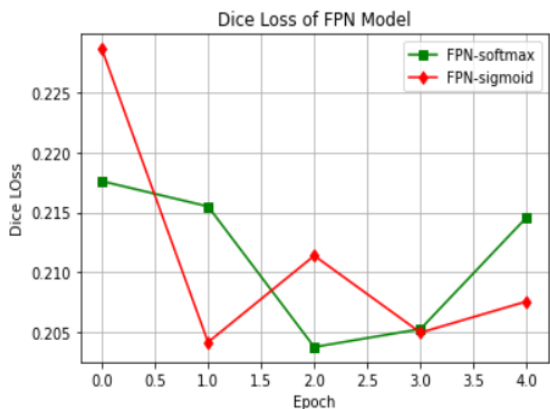


Figure 18 : Dice Loss of FPN Model

Dice Loss of DeepLabV3Plus segmentation model with resnet50 and resnet101 encoders for different epochs are plotted in Figure 19.

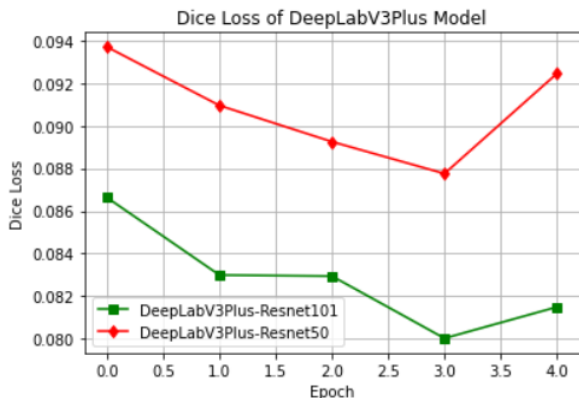


Figure 19 : DiceLoss of DeepLabV3Plus Model

Dice Loss of the 3 different segementation models are as follows,

| S.NO | Segmentation Model | Encoder | Activation Function | Mean Dice Loss |
|------|--------------------|-----------|---------------------|----------------|
| 1 | Unet | Resnet34 | Sigmoid | 0.1400 |
| 2 | Unet | Resnet34 | Softmax | 0.1031 |
| 3 | FPN | Resnet50 | Sigmoid | 0.2076 |
| 4 | FPN | Resnet50 | Softmax | 0.2146 |
| 5 | DeepLabV3Plus | Resnet50 | Sigmoid | 0.0908 |
| 6 | DeepLabV3Plus | Resnet101 | Sigmoid | 0.0815 |

Table 3 : Mean Dice Loss of Segmentation Models

Among the three segmentation models, DeepLabV3Plus gives more less dice loss with resnet101 encoder. Unet gives more dice loss with sigmoid function than with softmax function. FPN gives more dice loss with softmax function than with sigmoid function. In case of Unet, resnet34 encoder is used whereas for FPN resnet50 encoder is used. DeepV3Plus gives less dice loss with resnet101 encoder than with resnet50 encoder.

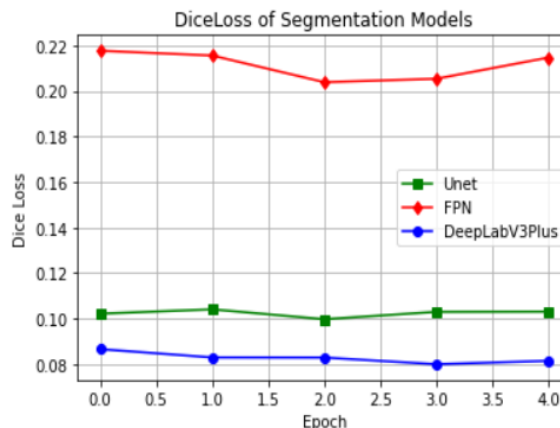


Figure 20 : Dice Loss of Segmentation Models

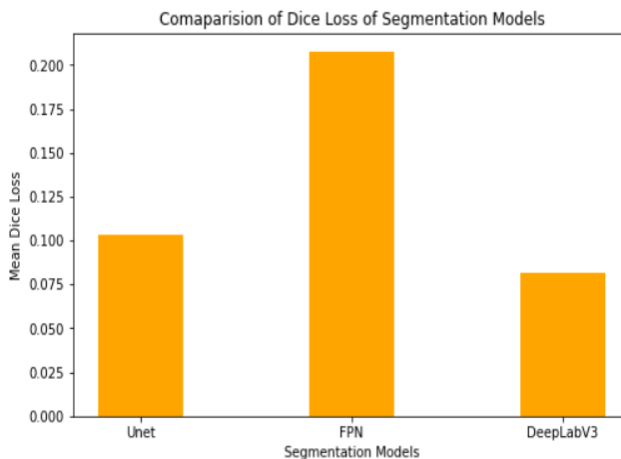


Figure 21: Comparison of Dice Loss of Segmentation Models

Accuracies:

Accuracies of the 3 different segementation models are as definded above.

Among the three segmentation models, DeepLabV3Plus gives more accuracy with resnet101 encoder. Unet gives more accuracy with softmax function than with sigmoid function. Similarly, FPN also gives more accuracy with softmax function than with sigmoid function. In case of Unet, resnet34 encoder is used whereas for FPN resnet50 encoder is used. DeepV3Plus gives more accuracy with resnet101 encoder than with resnet50 encoder.

| S.N O | Segmentation Model | Encoder | Activati on Function | Accuraci es |
|-------|--------------------|-----------|----------------------|-------------|
| 1 | Unet | Resnet34 | Sigmoid | 78.90% |
| 2 | Unet | Resnet34 | Softmax | 85.31% |
| 3 | FPN | Resnet50 | Sigmoid | 73.14% |
| 4 | FPN | Resnet50 | Softmax | 72.21% |
| 5 | DeepLabV3Plus | Resnet50 | Sigmoid | 87.13% |
| 6 | DeepLabV3Plus | Resnet101 | Sigmoid | 90.17% |

Table 4 : Accuracies of Segmentation Models

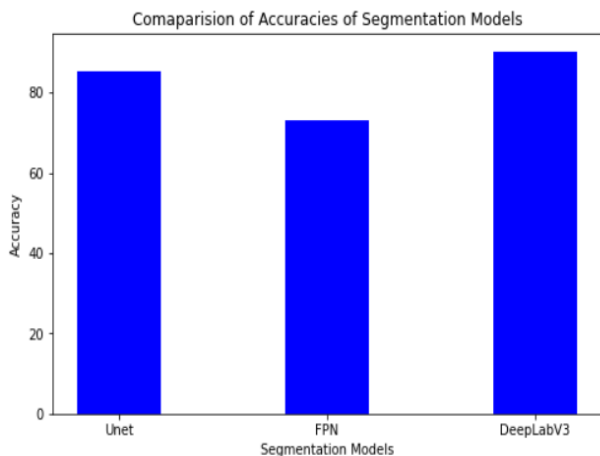


Figure 22: Comparison of Accuracies of Segmentation Models

5. Conclusion

Our work explores three semantic segmentation models, FPN, U-Net, and DeepLabV3Plus, for the classification of Land cover. Even though the satellite images show considerable variation, the pre-trained deep learning-based segmentation models still offer encouraging results. Further testing of all three models with a valid set of satellite images has further enhanced their accuracy compared to FPN. Overall, DeeLab3Plus and U-Net are more accurate than FPN. As a result, the deep learning-based segmentation models achieve Mean IoU of 90%, 85% and 73%, respectively. These models outperform conventional methods significantly by a large margin. We may extend the work to include other deep learning-based segmentation models using different datasets in the future.

6. References

- [1] D. R. Rao, S. Noorjahan and S. A. Fathima, "Classification of Land Cover Usage from Satellite Images using Deep Learning Algorithms," in ICEARS, 2022.
- [2] Rahul Neware " Comparative Analysis of Land Cover Classification Using ML and SVM Classifier for LISS-iv Data", 2019.
- [3] Keerti Kulkarni; P. A. Vijaya " A comparitive study of land classification using remotely sensed data" in IEEE, 2017.
- [4] Sandra Yesudasan Miranda, Vineetha P., Smitha Asok Vijayamma, "A Comparative Study of Land use Classification using Remote Sensing Techniques, in and around Selected Sacred Groves of Thiruvananthapuram District "in 2016.
- [5] R. Sivagami; R. Krishankumar; K. S. Ravichandran, " A Comparative Analysis of Supervised Learning Techniques for Pixel Classification in Remote Sensing Images" in IEEE, 2018.
- [6] Selim S. Seferbekov, Vladimir I. Iglovikov, Alexander V. Buslaev, Alexey A. Shvets, "Feature Pyramid Network for Multi-Class Land Segmentation" in 2018.
- [7] Python, <https://www.python.org/doc/essays/blurb/>
- [8] Pandas, <https://pandas.pydata.org/docs/pandas.pdf>
- [9] UML diagrams includes use case, activity diagrams, sequential diagrams, <https://modeling-languages.com/>
- [10] RajibMall, Fundamentals of Software Engineering. 2ed, PHI.
- [11] Use case, activity diagrams, sequential diagrams, <https://modeling-languages.com/>
- [12] Understanding of the LSTM Networks, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [13] TensorFlow, <https://www.tensorflow.org/about>.
- [14] Connor Shorten, Taghi M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning" in 2019.
- [15] "Segmentation-models-pytorch 0.2.1" in 2021.
- [16] Deep Globe Land Cover Classification Dataset in Kaggle, 2018.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

An Innovative Methodology of the Algorithmic Composition Knowledge Base for the Chinese Calligraphy Painting Interaction: from Sonification to Musification

Chih-Fang Huang^{a,*} and Jenny Ren^b

^a Department of Health and Marketing, Kainan University, Taiwan

^b Chunghwa Telecom Laboratories, Taiwan

ARTICLE INFO

Article History:

Submitted 11.8.2021

Revised 3.17.2022

Second Revision 5.20.2022

Accepted 5.30.2022

Keywords:

Chinese Calligraphy Painting, sonification scheme, two-dimensional spatial image information, temporal music acoustics domain, visually impaired, mind and imagery

ABSTRACT

Chinese characters are remarkable for the form of art, as the Chinese Calligraphy Painting. However, it is difficult for the visually impaired and people who are unfamiliar with Chinese to experience the beauty of the Chinese characters. In this study, the Sonification scheme, Im2Ms, is proposed to extract the melody between the lines, i.e., the lines in strokes. In Im2Ms, the two-dimensional spatial image information is transformed into the temporal music acoustics domain based on artistic conception and human perception among space, color and sound interaction. Therefore, the Sonification of Chinese Calligraphy Painting not only provide a free access for the visually impaired and people who are unfamiliar with Chinese to appreciation but also enrich the state of mind and imagery in the delivery process. Thus, an immersive appreciation environment of Chinese Calligraphy Painting can be further developed.

© 2022 KSI Research

1. Introduction

The birth of this study is driven by the idea: “Is there any mechanism for assisting the visually impaired, and people who are unfamiliar with Chinese in experiencing the art in the form of Chinese Characters (i.e., calligraphy) through an alternative modality, hearing?” Fortunately, we found that both image and sound can evoke emotional responses. Since each modality has its certain strengths and each combination of modalities may produce different synergistic results, sound can provide an additional and complementary perceptual channel. Besides, sound can be used to augment the visualization by permitting a user to visually concentrate on one field, while listening to the other. Consequently, the aim of this study is to explore and utilize the auditory display to strengthen the synesthesia and to supplement the visual interpretation of data based on the artistic interrelationship. In other words, the digital data in two different kinds of medium (images and sounds) are being manipulated. (Figure 1) depicts the

interaction of different display modalities of cross-disciplinary arts.

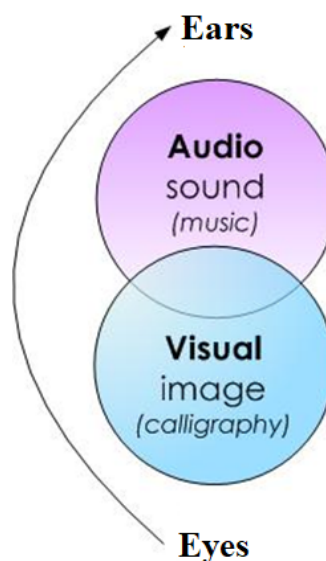


Figure 1: Three display modalities of cross-disciplinary arts in this study.

1.1 Scenarios and Contributions

The following is the scenario in this study, which maps from image onto sound, or space to time: appreciating the Chinese Calligraphy Painting, taking the example of the “Cursive Script”. Although there has been research on mapping from image to sound, none of them is dedicated to Chinese Characters. The contributions of this study are as follows: Firstly, image is analytically transformed from graphic data view into abstract sonic space. Secondly, data is mapped to sound in a musical way. The visual data representation is algorithmically compiled into audio data representation with philosophical and aesthetic interrelation through compositional mind and process rather than arbitrarily or directly converted — a step forward from Sonification to Musification. Thirdly, an immersive learning environment with audio-visual aids is built since it supports concentration, provides engagement, increases perceived quality, and enhances learning creativity during the appreciation process.

1.2 Sonification

The word “Sonification” comprises the two Latin syllabus “sonus”, meaning sound, and the ending “fication”, forming nouns from verbs which are ending with ‘-fy’. Therefore, to “sonify” means to convey the information via sound. A Geiger detector can be seen as the very basic scientific example for Sonification, which conveys (i.e., sonifies) information about the level of radiation. A clock is even more basically an example for Sonification, which conveys the current time. The word “Sonification” has already been defined in a majority of researches. “Sonification is to communicate information through nonspeech sounds” [1]; “Sonification is the use of data to control a sound generator for the purpose of monitoring and analysis of the data” [2]; “Sonification is the transformation of data relations into perceived relations in an acoustic signal for the purpose of facilitating communication or interpretation” [3]; “Sonification is a mapping of numerically represented relations in some domain under study to relations in an acoustic domain for the purpose of interpreting, understanding, or communicating relations in the domain under study” [4]. (Figure 2) illustrates the existing Sonification techniques, which are already categorized into three types according to the mapping approach adopted: syntactic, semantic or lexical mapping [5], [6].

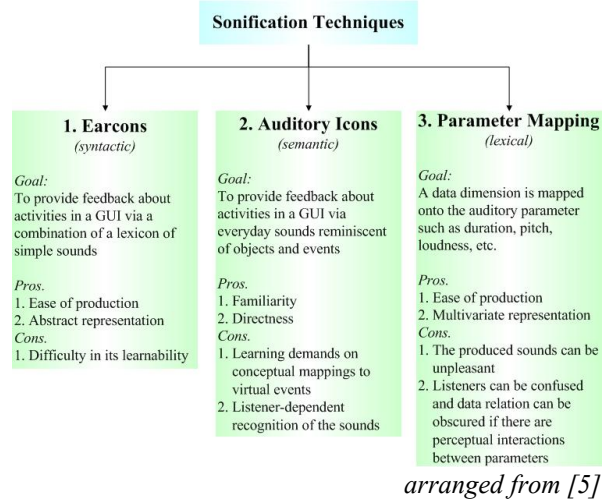


Figure 2: Three types of existing sonification techniques.

Besides, the fundamental elements of a Sonification are suggested in (Figure 3) from both data-centric and human-centric points of view [7], including the functionality to be identified, tasks to be performed, and several related disciplines to be worked with.

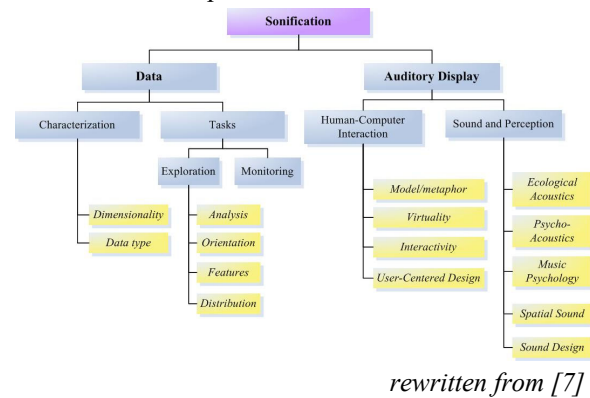


Figure 3: Elements hierarchy of a sonification display.

Moreover, (Figure 4) illustrates the fundamental procedure about how to design a Sonification system, where the Communicative Medium is the core of Sonification [8].

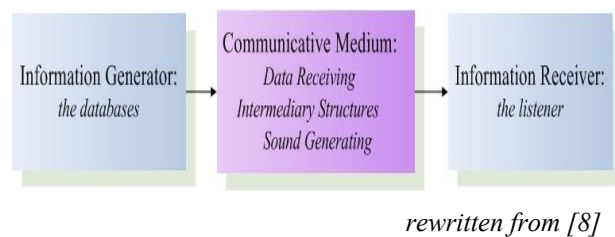


Figure 4: Schematic of an Auditory Display System.

However, the essential goal of Sonification is to yield an auditory display that will be orderly and intuitively maximal in meaning (i.e., coherence) to the observer. Inevitably, the effectiveness is what most counts in designing a Sonification software or system as shown

in (Figure 5). For functionality, the goal-oriented function of the system must be clearly defined. For utility, if the sound is ugly, people won't use it. Consequently, the craft of composition is important to auditory display design (i.e., a composer's skill can contribute to making auditory displays more pleasant and sonically integrated and so contribute significantly to the acceptance of such displays). For expectancy, evaluation (e.g., questionnaire) is needed.

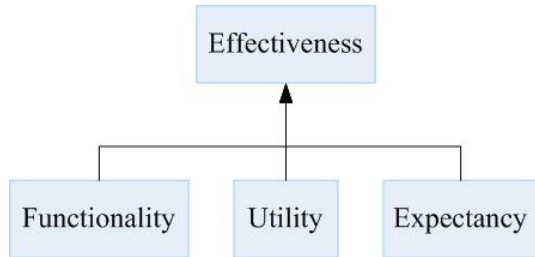


Figure 5: How to make an effective sonification?

1.3 Musification

Since data can be visualized by means of graphics and sonified by means of sounds, data can be musified by means of music as well. “Musification is the musical representation of data” [9]. However, the difference between Sonification and *Musification* lies in the fact — Music is “organized sounds”, coined by French composer, Edgard Varèse [10]. Specifically speaking, data is no longer directly mapped onto audio signal level, but algorithmically compiled onto musical structure level, which means, to follow some musical grammars or based on musical acoustics.

1.4 Current Research Trends in Sonification

This section reviews relevant research trends in Sonification from two aspects: a diversity of usages and image-to-sound.

1.4.1 Sonification in a Diversity of Usages

Sonification has been put into practice in a variety of areas, inclusive of medical usages, assistive technologies, or even data mining and information visualization. The idea of using Sonification in medical usage is to use sounds to diagnose illness; the idea of carrying out Sonification in assistive technologies is to make maps, diagrams and texts more accessible to the visually impaired through multimedia computer programs; the idea of applying a direct playback technique, called “*Audification*”, in data mining and information visualization is to assist in overlooking large data sets, event recognition, signal detection, model matching and education [11]. Besides, the method for rendering the complex scientific data into sounds via additive sound synthesis and further visualizing the sounds in Virtual-Reality environment has been proposed in [12], which is

aimed to help scientists explore and analyze huge data sets in scientific computing.

1.4.2 Image-to-Sound Sonification





Kandinsky produced many paintings, which borrows motifs from traditional European music, based on the correspondence between the timbres of musical instruments and colors of visual image [13]. Contrary to Kandinsky's attempt to “see the music”, there are researchers and artists who have been trying to “hear the image”. Iannis Xenakis' UPIC (Unité Polyagogique Informatique du CEMAMu) system may be one of the first digital graphics-to-sound schemes. Composers are allowed to draw lines, curves, and points as a time-frequency score on a large-size and high-resolution graphics tablet for input [14]. Later on, many of the ideas that drive image-to-sound software are inspired from Xenakis' research. Unlike UPIC as a graphical metaphor of score, Coagula is an image synthesizer which uses pixel-based conversion, where x and y coordinates of an image are regarded as time and frequency axis, with a particular set of color-to-sound mappings. Red and Green control stereo panning, while Blue smears the sounds to noise content [15]. The vOICE (read the capitalized letters aloud individually to get “Oh, I see!”), or, “Seeing with Sounds”, is a system that makes inverted spectrograms in order to translate visual images into sounds, where the two-dimensional spatial brightness map of a visual image is 1-to-1 scanned and transformed into a two-dimensional map of oscillation amplitude as a function of frequency and time [16]. The mapping translates, for each pixel, vertical position into frequency, horizontal position into time-after-click, and brightness into oscillation amplitude — the more elevated position the pixel, the higher frequency the associated oscillator; the brighter the pixel, the louder the associated oscillator. The oscillator signals for a single column are then superimposed. In Wang's research [17], the image is converted from RGB to HSI system and then be mapped from Hue (0-360 degree) to pitch (MIDI: 0-127), from Intensity (0-1) to playback tempo (0-255), respectively. In the research of Osmanovic, the image is mapped from its electromagnetic spectrum to tone frequency and from intensity to volume based on color properties and sound properties. The frequency of the tone is redoubled 40 times to compute the frequency of the color: $\text{tone} \times 2^{40} = \text{color}$ [18].

1.4.3 Chinese Calligraphy Painting

Chinese Calligraphy Painting is highly ranked as an important art form in East Asia, referring to the beautiful handwriting of Chinese Characters. Seal Script, Clerical Script, Cursive Script and Regular Script are the primary styles in the evolution of Chinese Calligraphy. (Table 1) displays the same Chinese Character, which means “thousand” in Chinese, in four different styles. Among all, Cursive Script is the most expressive and individual style, which draws the musical rhythm and speed in two dimensional space on

the Shuan Paper.

Table 1: Evolution of four primary styles of Chinese calligraphy.

| Image | In Chinese | In English |
|---|------------|-----------------|
|  | chuan-shu | Seal Script |
|  | li-shu | Clerical Script |
|  | tsao-shu | Cursive Script |
|  | kai-shu | Regular Script |

This artistic creation of Chinese Characters is rich in pictorial splendor, and deep in implicit imagery. The thickness, length, strength, speed and shape of the characters with the transition (stop and change) of the brush strokes convey the disposition of the calligrapher and enchantment of the calligraphy itself.

2. Methodology

“The mapping problem” has been regarded as the essential issue of Sonification. In Chinese Calligraphy Painting, we found that the calligraphers signify their personality through the strokes and modeling of a character. In this study, the Sonification mechanisms of transforming Chinese Calligraphy Painting (image) into music is explored. First of all, the aesthetic features are extracted from Chinese Calligraphy Painting. Afterwards, the rules are applied to Im2Ms (image-to-music mapping of Chinese Calligraphy Painting) parameter-mapping mechanism. The conversion of image-to-music mapping is based on a relationship that exists among space, color and sound in human perception. (Figure 6) illustrates a general paradigm of our Sonification in this study. A limited number of features and corresponding sonic attributes are taken into account so as to keep the resultant sounds as simple as possible and easy to decode because the listeners always wish to hear what the data is doing. However, the sound will be still rich in itself.

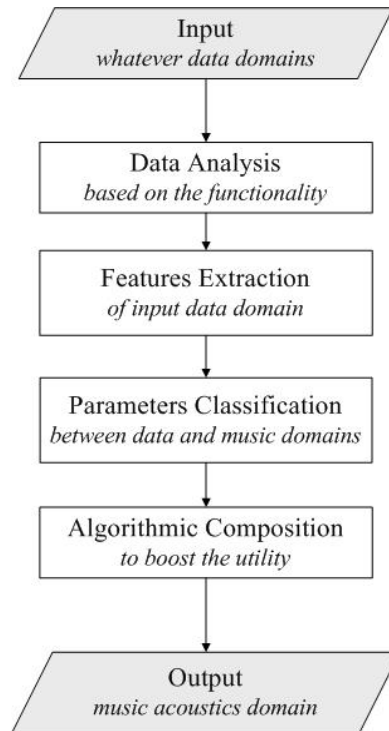


Figure 6: General paradigm of sonification.

Throughout this section, we provide a mechanism of mapping calligraphy data onto appropriate sound features along with an overview of the system architecture.

2.1 Mapping Recipe of Im2Ms

In most practical applications, a raw data image is hardly observed any useful information. However, the preprocessing of an image contributes to features extraction in image analysis, as shown in (Figure 7).

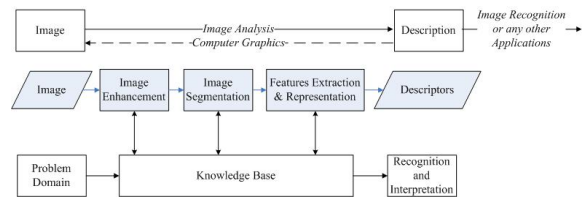


Figure 7: Typical process in image analysis.

Since a Chinese Character is regarded as an image in this study, we are supposed to explore the abstract elements which comprise a picture from a Chinese Character. Every image has its external frame and internal content, where the former means the explicit shape characteristics (i.e., contour information, gesture of lines) and the later means the implicit structural features (i.e., skeleton information, end points). Basically we convert a Chinese Character to sound according to its shaping frame of pixels. Furthermore, the sound is characterized by its structural component of content implied in the Chinese Character, as shown in (Figure 8).

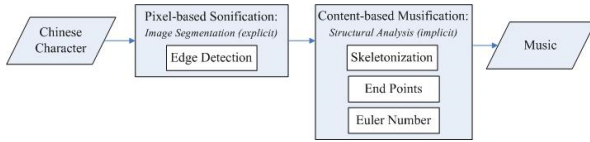


Figure 8: Image analysis of a Chinese character.

From the micro perspective, the smallest unit of an image in computer vision has information about its RGB (true-color), intensity (gray-level), position (X-Y), etc. According to the traditional spectrograph display of sounds, the two-dimensional axes are one for frequency and the other for time. Besides, concerning the human behavioral habits of writing a Chinese character (where most strokes are basically from top left towards bottom right corner), two-way scanning methods (i.e., from left to right & from top to bottom) are both adopted. Moreover, from the top-to-down perspective, each image as a whole can be analyzed by its contour, shape, etc. (Table 2) shows the mapping between image and music parameters.

Table 2: Mappings from image information to musical parameters.

| Parameters Mapping | | | Music | | | |
|--------------------|--------------|--------------|-------|----------|-------|--------|
| Image Edge | Position | Right, Top | Pitch | Dynamics | Tempo | Timbre |
| | | Left, Bottom | High | Low | NULL | |
| Image Structure | Intensity | Dark | NULL | Loud | NULL | |
| | | Bright | NULL | Soft | NULL | |
| Image Structure | End Point | More | NULL | Slower | NULL | |
| | | Less | NULL | Faster | NULL | |
| Image Structure | Euler Number | Non Positive | NULL | | | Smooth |
| | | Positive | NULL | | | Sharp |

2.2 Preliminaries of Im2Ms

2.2.1 Segmentation

Human vision is very good at edge detection so that edge detection is one of the most essential tasks in image analysis. Edge detection is extensively used in image segmentation since edges characterize object boundaries. Representing an image by its edges has the advantage that the amount of original image data is significantly reduced and useless information is filtered out, while preserving most of the important structural properties in an image. In typical image, edges are places with strong intensity contrast. An edge is a jump in intensity from one pixel to the next, where drastic change occurs in gray level over a small spatial distance (e.g. surface color or illumination discontinuity). Hence, edges correspond to high spatial frequency components in the image signal. The majority of different methods to perform edge detection can be grouped into two categories, gradient and Laplacian. The gradient-based method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian-based method finds the edges by searching for zero crossings in the second derivative of the image. For Gradient-based edge detection methods, the gradient vector represents: (1) the direction in the n-D space along which the function increases most rapidly, and (2) the rate of the increment. Here we only consider 2D field:

$$\nabla \cong \frac{\partial}{\partial x} \vec{i} + \frac{\partial}{\partial y} \vec{j} \quad (1)$$

where \vec{i} and \vec{j} are unit vectors in the x and y directions respectively. The generalization of a 2-D function $f(x, y)$ is the gradient:

$$\vec{g}(x, y) \cong \nabla f(x, y) = \left(\frac{\partial}{\partial x} \vec{i} + \frac{\partial}{\partial y} \vec{j} \right) f(x, y) = f_x \vec{i} + f_y \vec{j} \quad (2)$$

The magnitude of $g(x, y)$ is first computed, and is then compared to a threshold to find candidate edge points.

$$|\nabla f(x, y)| = \sqrt{\left(\frac{\partial f(x, y)}{\partial x} \right)^2 + \left(\frac{\partial f(x, y)}{\partial y} \right)^2} \quad (3)$$

For Laplacian-based edge detection methods, the Laplace operator is defined as the dot product (inner product) of two gradient vector operators:

$$\Delta \cong \nabla^2 \cong \nabla \cdot \nabla = \left(\frac{\partial}{\partial x} \vec{i} + \frac{\partial}{\partial y} \vec{j} \right) \cdot \left(\frac{\partial}{\partial x} \vec{i} + \frac{\partial}{\partial y} \vec{j} \right) = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad (4)$$

The generalization of a 2-D function $f(x, y)$ is the gradient:

$$\Delta f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (5)$$

2.2.2 Features Extraction

Based on the topologically and morphologically structural attributes of a Chinese Character, two descriptors, as shown in (Table 3), are utilized to produce the mapping rules of implicit structural content along with the mapping criteria of explicit frame of pixels. One is the morphological attribute — End Point, where each End Point $EP(x, y)$ must satisfy the following two conditions within the eight-connected chain code (Figure 9):

$$EP(x, y) = 1 \quad (6)$$

$$EP(x+1, y) + EP(x+1, y+1) + EP(x, y+1) + EP(x-1, y+1) + EP(x-1, y) + EP(x-1, y-1) + EP(x, y-1) + EP(x+1, y-1) = 1 \quad (7)$$

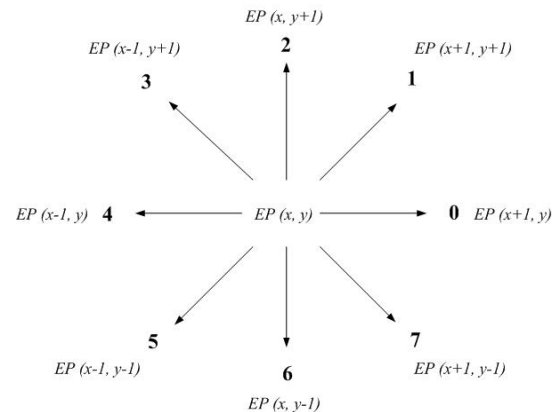






Figure 9: End Point detection by examining the 8-connected chain code elements.

The other is the topological attribute — Euler Number, which means the total number of objects in the image (i.e., C) minus the total number of holes in those objects (i.e., H), defined as:

$$E = C - H \tag{8}$$

Table 3: Two structural descriptors of a Chinese character.

| <i>Structural Features</i> | <i>Illustration</i> | <i>Metaphor</i> |
|------------------------------|---|---|
| <i>End Point</i> |  End Point = 7 | Since the number of End Points implies the number of strokes of a Chinese Character, the more the end points, the more complicated and higher fragmentation degree a Chinese Character; the less the end points, the smoother a Chinese Character. |
| <i>Loop and Euler Number</i> |  Loop = 3 Euler Number = -2  Loop = 2 Euler Number = -1  Loop = 1 Euler Number | Since the number of Euler Number implies the number of closed regions in a Chinese Character, the more the Euler Number, the higher closed degree a Chinese Character. Moreover, the more End Points plus the Euler Number, the more changes of breaths during the writing and thus the slower tempo to accomplish a Chinese Character. |

| | | |
|--|-----|--|
| | = 0 | |
|--|-----|--|

2.2.3 System Architecture

(Figure 10) and (Figure 11) illustrate the system flow chart and the system architecture of Im2Ms, namely the image-to-music conversion of Chinese Calligraphy Painting, where Fig. 11 takes a Chinese Cursive Script, which means “thousand”, for example.

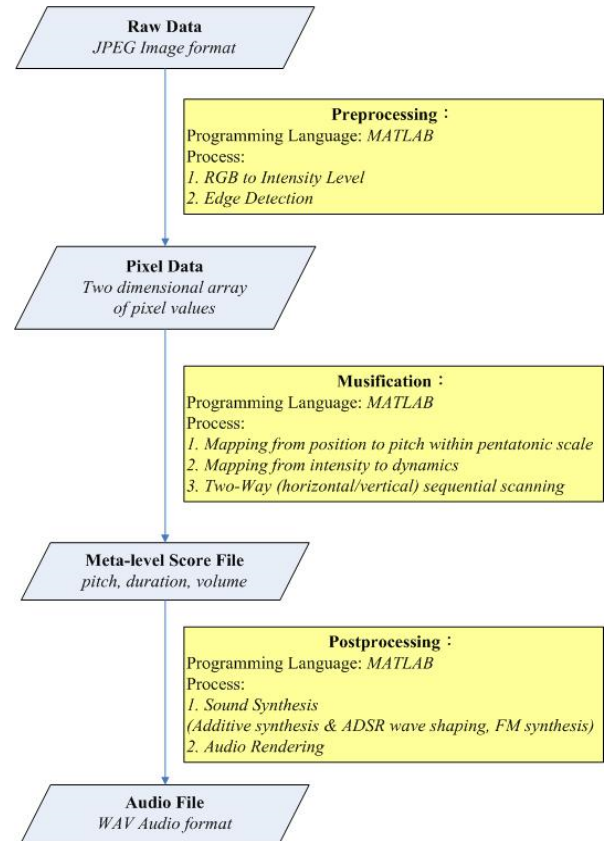


Figure 10: System flow chart of Im2Ms.

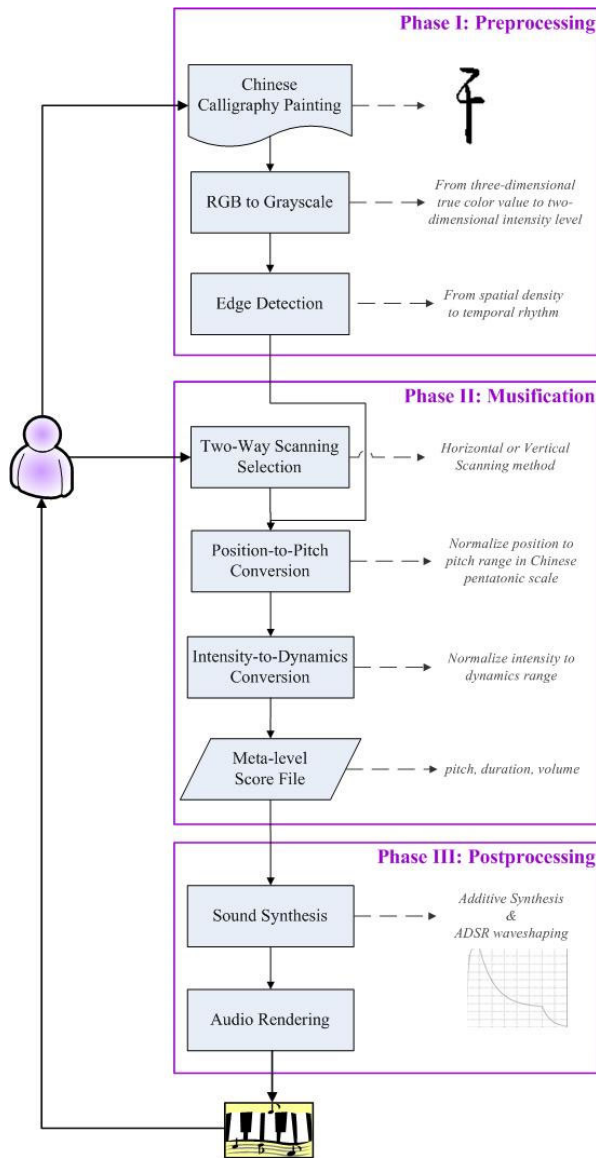


Figure 11: System architecture of Im2Ms.

3. Experiment Results

The Sonification scheme is implemented in MATLAB for rapid prototyping and exploration, where MATLAB is an environment with powerful mathematical computing especially for digital signal and image processing. (Figure 12) and (Figure 13) illustrate the output of Im2Ms in both waveform and spectrogram displays of the exemplified cursive script, as shown in (Figure 11), by horizontal and vertical scanning method respectively.

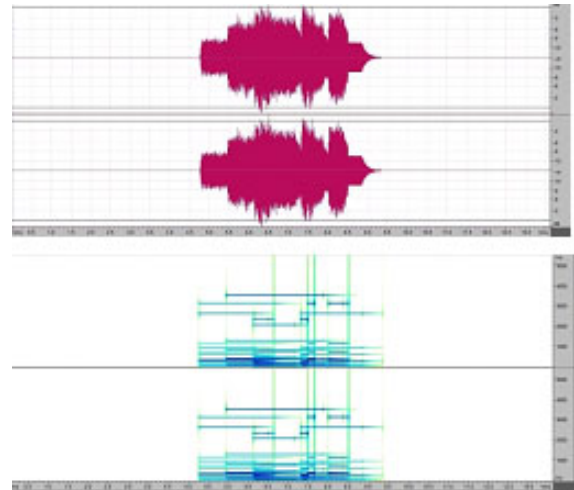


Figure 12: Waveform and spectrogram of output in Fig. 11 by horizontal scanning.

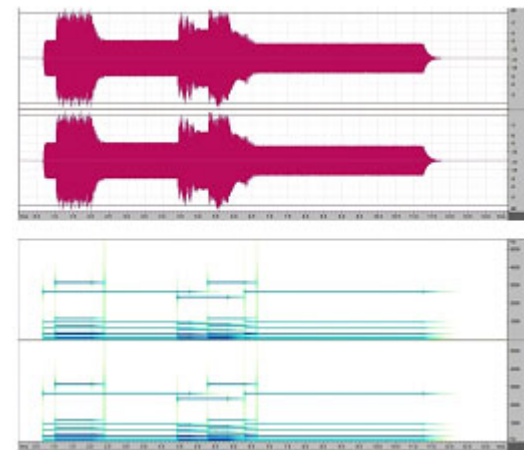


Figure 13: Waveform and spectrogram of output in Fig. 11 by vertical scanning.

As shown in (Figure 14) and (Figure 15), Im2Ms achieve its responsiveness during the Musification process by observing pitch distributions from two-way scanning mechanism. It does meet the goal in expectancy of an effective Sonification as the mentioned.

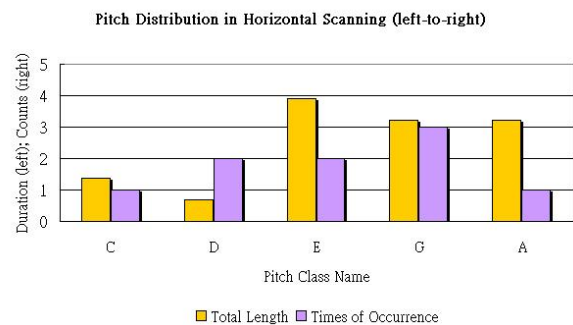


Figure 14: Pitch Distribution of output in Fig. 11 by horizontal scanning.

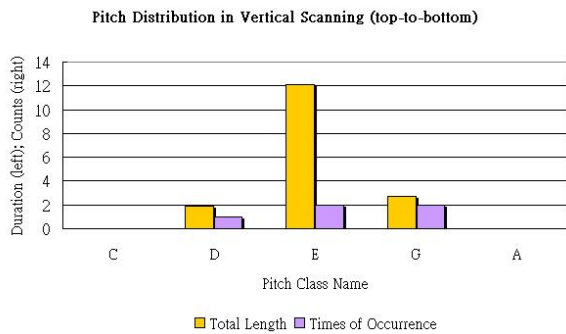


Figure 15: Pitch Distribution of output in Fig. 11 by vertical scanning.

4. Conclusion and Future Work

Currently, the adaptive Musification prototype designed for Chinese Calligraphy Painting is proposed. General conclusion is that the sounds produced convey the information about the imagery state of mind and the qualitative nature of the data. For Image-to-Music conversion, on the one hand, the position-to-pitch mapping is more intuitively responsive to original visual data and easy for gestalt formation than color-to-pitch applied in the two related works. However, color could be mapped into timbre instead. On the other hand, notwithstanding the two parameters are taken into account (i.e., position and intensity), the two-way scanning results in an extra musical effect — the sonority. To sum up, the texture of the image in both horizontally and vertically sequential scanning reflects on the sonority of the music with interaction. Many interesting applications could be realized based on this study, such as an Audible Digital Album (let the photos sing!). Nevertheless, the actual resolution obtainable with human perception of these sound representations remains to be evaluated, and the algorithmic composition throughout the Musification process need more improvements. The involvement of expertise in image processing (Chinese Character Recognition), music composition, and even psychology is critical for its success. Although this study has systematically investigated the logical and reasonable mappings from the degrees of freedom in the data to the parameters controlling the algorithmic composition or sound synthesis process, such as the FM (Frequency Modulation) Synthesis for the image sonification [19] and [20], there are still few limitations of this study. The most obvious one is the lack of strokes sequential information in the Im2Ms. The sequential strokes of a Chinese character play a significant role in this kind of specific image as an important feature itself. Consequently, there might be an alternative demand for a real-time and interactive Musification for Chinese Calligraphy Painting. Mouse, write pad, or other related input devices could be used to obtain more image

information, such as the sequence of the character, instead of simply horizontal and vertical scanning. Take the following idea for example. Since the writing sequence is based on the “arrow”, the writing segments are then retrieved for sections of music, with “rest” based on the timing between the end of the last segment and the beginning of the next segment. Simply speaking, the scanning sequence is no longer the pure left-to-right or top-to-bottom, but the real-time writing strokes recorded sequentially. In this way, the image content could be mapped into music, where the vertical axis variance determines the pitch in Pentatonic Scale up or down and the horizontal axis variance determines the timbre, as shown in (Figure 16).

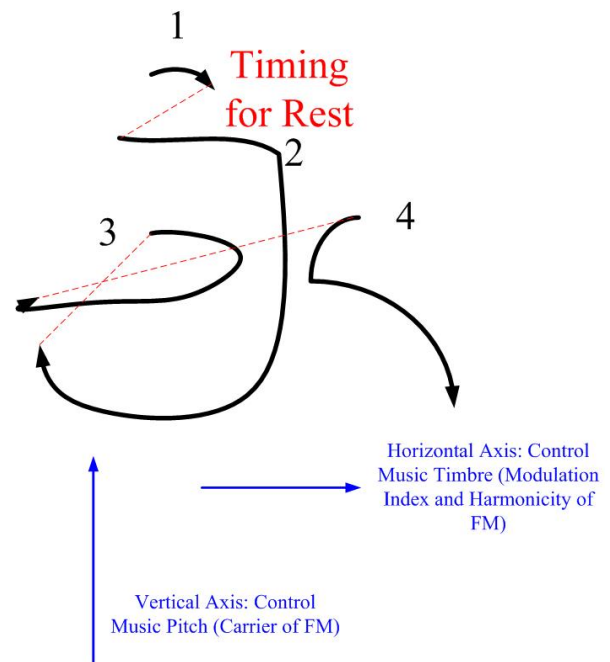


Figure 16: An exemplified Calligraphy Musification algorithm with FM.

Acknowledgement

The authors are appreciative of the support from the Ministry of Science and Technology project of Taiwan: MOST 108-2511-H-424 -001 -MY3.

References

- [1] Kramer, G. (2000). “Auditory display: sonification, audification and auditory interfaces”. Addison-Wesley Longman Publishing Co., Inc..
- [2] Kramer, G. (1994). “Some organizing principles for representing data with sound”. Auditory Display-Sonification, Audification, and Auditory Interfaces, 185-221.
- [3] Kramer, G. et al. “Sonification Report: Status of the Field and Research Agenda”. Report prepared for the National Science Foundation by members of the International Community for Auditory Display. <http://www.icad.org/websiteV2.0/References/nsf.html> (accessed May 9, 2020).

- [4] Kramer, G. (2000). "Auditory display: sonification, audification and auditory interfaces". Addison-Wesley Longman Publishing Co., Inc..
- [5] Barrass, S., Kramer, G. (1999). "Using sonification. Multimedia systems", 7(1), pp. 23-31.
- [6] Blattner, M. M. (1994). "Sonic enhancement of two-dimensional graphics displays". Auditory display-Sonification, audification and auditory interfaces, 447-470.
- [7] Saue, S. (2000). "A model for interaction in exploratory sonification displays". Georgia Institute of Technology.
- [8] Kramer, G. (1994). "An introduction to auditory display. Auditory display-Sonification, audification and auditory interfaces", pp. 1-77.
- [9] Visi, F., et al. (2014). "Unfolding| Clusters: A Music and Visual Media Model of ALS Pathophysiology". Proceedings of SoniHED Conference: Sonification of Health and Environmental Data. York, United Kingdom.
- [10] Richardson, R. (2005). "Edgard Varèse and the Visual Avant-Garde: A Comparative Study of Intégrales and Works of Art by Marcel Duchamp". Doctoral dissertation, University of Cincinnati, USA.
- [11] Barrass, S., Kramer, G. (1999). "Using sonification. Multimedia systems". 7(1), pp. 23-31.
- [12] Kaper, H. G., Wiebel, E., Tipei, S. (1999). "Data sonification and sound visualization". Computing in science & engineering, 1(4), pp. 48-58.
- [13] Kandinsky, W. (2012). "Concerning the spiritual in art. Courier Corporation". Dover Publications, USA.
- [14] Marino, G., Serra, M. H., Racizinski, J. M. (1993). "The UPIC system: Origins and innovations". Perspectives of New Music, pp. 258-269.
- [15] Ekman, R. (2003). "Coagula-Industrial Strength Colour Note Organ-Light Version 1.666".
- [16] Meijer, P. B. (1992). "An experimental system for auditory image representations". IEEE transactions on biomedical engineering, 39(2), pp. 112-121.
- [17] Wang, W. C. (2005) "A Study of Computer Composition on Using Image Analysis". Master thesis, National Taipei University of the Arts, Taiwan.
- [18] Osmanovic, N., Hrustemovic, N., Myler, H. R. (2003). "A testbed for auralization of graphic art". In Annual Technical Conference IEEE Region 5, pp. 45-49, IEEE.
- [19] Yeo, W. S., Berger, J. (2006). "Application of raster scanning method to image sonification, sound visualization, sound analysis and synthesis". In Proc of the 9th Int. Conference on Digital Audio Effects, pp. 18-20.
- [20] Kew, H., Stables, R. (2020). "Sonification of Spectroscopic analysis of food data using FM Synthesis". In Audio Engineering Society Convention 149. Audio Engineering Society.

Journal of Visual Language and Computing

Volume 2022, Number 1