
This is the Author Accepted Manuscript.

There may be differences between this version and the published version. You are advised to consult the publisher’s version if you wish to cite from it.

http://eprints.gla.ac.uk/278456/

Deposited on: 05 September 2022
Motion Embedding for On-road Motion Object Detection for Intelligent Vehicle Systems

Rongzihan Song¹, L.L Chamara Kasun¹, Qi Cao², Xinggan Peng¹, Guang-bin Huang³ and Zhiping Lin¹

Abstract—Accurate motion object detection (MOD) using in-vehicle cameras in driving vehicles is a challenging task. Several deep learning based motion segmentation approaches have been reported based on the interpretable optical flow feature. However, the interpretable optical flow feature has not been explored by object-level MOD approaches. In this paper, we propose a motion embedding pipeline (MEP) architecture that utilizes interpretable optical flow and deep learning to solve object-level MOD problems. The MEP is a three-stage pipeline that consists of an object detector, a novel feature extraction algorithm to capture relative motion between objects and the background, as well as a motion predictor for representation learning with stacked autoencoder to determine motions. A new dataset, Singapore motion object detection (SG-MOD) dataset is constructed in this work with much larger variations in urban environments. Experimental results show that the proposed MEP outperforms other pipeline-based architecture and deep learning based approaches on the SG-MOD and KITTI-MOD datasets in most metrics.

I. INTRODUCTION

Intelligent vehicle systems (IVS) provide real-time assistance to drivers by monitoring surrounding environment and states using artificial intelligent (AI) enabled in-vehicle cameras. One of the typical applications of the IVS assistance could be in the form of autonomous driving systems (ADS) [1] or warning systems [2] for motion object detection (MOD) which identifies moving and static objects along driving paths of the moving vehicle using image data. It is an interesting research area for MOD algorithms to achieve good detection accuracy in various environments. Real-time detected object motion information can be directed to ADS to avoid potential collisions or to a warning system to alert drivers in advance. Besides, MOD techniques can be adopted in active illegal road-side parking detection [3] using cameras installed on moving vehicles.

MOD algorithms reported in the literature can be categorised as end-to-end based architecture, e.g., [4], [5], [6], [7], [8] and pipeline based architecture, e.g., [9], [10]. End-to-end based architecture consists of multiple modules, each of which module performs a specific task such as learning motion information or learning object boundaries.

This architecture usually works for pixel-level MOD. An example is an end-to-end deep learning approach MODNet [5], that learns the object motion by segmentation. In contrast, pipeline based architecture consists of multiple algorithms in stages, where the previous stage outputs are used as inputs to the current stage algorithm. This architecture is usually adopted to object-level MOD tasks. Context-aware motion descriptor (CMD) [9] is a pipeline-based architecture that determines object motions by modeling the object and background motion using histograms. But the performance of CMD is constrained by the shallow network architecture object detector. Furthermore, no motion shape information in the optical flow is utilized.

For object detection tasks, YOLOx algorithm achieves state-of-the-art (SOTA) results with fast processing speeds [11]. In contrast to previous YOLO architectures [12], [13], [14] which use anchors to determine the object bounding box, YOLOx is an anchor-free algorithm that predicts the object bounding box directly. Hence, YOLOx requires less predictions and mitigates the effect of overfitting. Furthermore, deep learning based YOLOx is suitable for MOD tasks under occlusion, lighting conditions, and illumination conditions in real-world on-road environments.

Image optical flow data has often been used in motion-segmentation algorithms but has seldom been investigated in object-level MOD. Only parts of the optical-flow [9] or the motion boundary histogram (MBH) [15], [16] features are utilized in object-level MOD. The MBH consists of histogram of gradient (HOG) features to describe the shape of the motion information. However, the MBH is not good at capturing the changes of motion information by dividing the motion into x and y-axis directions. Motion changes are important for MOD tasks since they indicate the relative motion between objects and background.

In order to address these challenges, a motion embedding pipeline (MEP) algorithm is proposed in this paper by adopting the deep learning-based detector and interpretable image optical flow for object-level MOD problems. The proposed MEP consists of three stages: (1) object detection using deep learning-based YOLOx algorithm; (2) a novel motion gradient of flow (MGF) feature on interpretable image optical flow which captures the change of motions efficiently; (3) learning a suitable representation using the Stacked Autoencoder (SAE) [17], [18] to classify motions accordingly.

Existing datasets for MOD have relatively small variations in urban environments. In this work, a new dataset named SG-MOD has been constructed using in-vehicle cameras by driving around the central business district in Singapore.

¹Rongzihan Song, L.L Chamara Kasun, Xinggan Peng and Zhiping Lin are with School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Ave, Singapore 639798. Email: song01599e.ntu.edu.sg, chamarakasun@ntu.edu.sg, XINGGAN0001@e.ntu.edu.sg, ezplin@ntu.edu.sg
²Qi Cao is with the School of Computing Science, University of Glasgow, Singapore campus, Singapore 567739. Email: qi.cao@glasgow.ac.uk
³Guang-bin Huang is with the Mind PointEye, Singapore. Email: egbhuang@ntu.edu.sg
Fig. 1: MEP architecture

at different time slots to capture the complex surrounding vehicle motion. These image data contain large variations under urban scenarios.

The main contributions of this paper are:

1) Propose a new MGF feature that captures relative motion between the object and background to determine the motion of an object efficiently.

2) A three-stage MEP with deep learning based detector is designed by learning the representation of the proposed MGF features to better solve MOD tasks.

3) The motion detection dataset, SG-MOD is created with the largest variations under urban scenarios with similar numbers of moving and static objects. Experiments results on the SG-MOD dataset and KITTI-MOD benchmark dataset show that the MEP outperforms other SOTA algorithms in most metrics and with a lower computational cost.

The rest of this paper is organized as follows: In Section II, we describe the proposed MEP architecture and the MGF feature, and also give detailed analysis to them. Experimental results comparing the MEP with SOTA methods are presented in Section III. Extensive experiments and discussions are reported in this section. Section IV concludes this paper.

II. METHODOLOGY

The proposed MEP consists of three stages: (1) detector using YOLOx algorithm; (2) MGF features extraction of the objects identified in stage one; (3) motion classification by learning representations. SAE is a representation learning algorithm that learns a low-dimensional manifold of input data using encoder-decoder architecture by minimizing the reconstruction cost. In traditional motion detection algorithms, the encoder-decoder architecture is used for motion segmentation but not for learning representations. We show that learning representations are useful in MOD.

The MEP architecture is presented in Fig. 1. In the training phase we use the motion detection training data to fine-tune the YOLOx detector. This process is followed by the MGF feature calculation for ground truth (GT) bounding boxes on training data and training motion classification. In the testing phase, YOLOx bounding boxes are calculated on the testing data. This process is then followed by MGF feature extraction for detected YOLOx bounding boxes and motion classification. In this section, we first discuss the three stages in the training phase, then present the proposed MEP in the testing phase.

A. Training: YOLOx Detector

The object detection is performed by the YOLOx algorithm which returns the detected objects using bounding box positions and identity information. A transfer learning training strategy is adopted for the YOLOx detector using the training data. Given an input image at frame $t$, $x = x(t)$, the bounding box $d_v = y_v(t, k) \in \mathbb{R}^4$ is the $k$th object detected at frame $t$. While class $y_c$ is denoted by $y_c = y_c(t, k) \in \mathbb{R}$, and confidence $y_s$ is denoted by $y_s = y_s(t, k) \in \mathbb{R}$ of the $k$th object at frame $t$. YOLOx consists of two parts: one is a backbone network made of a convolutional neural network with residual connections; the other is a feature pyramid network with a decoupled head for object detection and object classification. We train the YOLOx algorithm by fine-tuning the bounding box prediction and object type classification with the MOD data as shown in Eq. (1):

$$J_{vc} = E_x[\|d_v - y_v\|^2_2 - E_x[d_v \log(y_c)]$$

where $d_v = d_v(t, k)$ is the GT bounding box and $d_v = d_v(t, k)$ is the GT object class of the $k$th object at frame $t$, respectively. $E_x$ refers to the expectation. The fine-tuning process intents to reduce the $J_{vc}$, thus errors of the predicted
bouding box position and classification will be reduced accordingly.

**B. Training: MGF Feature**

The MGF feature models the motion feature of objects and background by capturing the gradient changes between their motions. This property enables MGF to ignore the noises introduced by image optical flow and to focus on motion changes only. MGF is efficient since it provides better descriptions to motion changes compared to MBH as described in Section III-C, but is only half in feature size. It benefits the MEP algorithm in improving the processing speed as described in Section III-C.

As illustrated in Fig. 2, the MGF feature extraction consists of four steps: (1) optical flow generation; (2) generating a region of interest (ROI) from the bounding box GT $d_v$; (3) reshaping the ROI; (4) applying HOG feature extraction on the reshaped ROI. The optical flow is generated by PWCNet [19] and the output is converted to color representation as given in Eq. (2):

$$x_o(t) = \text{OF}(x(t), x(t - 1))$$  \hspace{1cm} (2)

where $\text{OF}$ denotes the optical flow function and $x_o(t)$ is the generated optical flow image of frame $t$. The GT bounding box $d_v = d_v(t, k)$ is used to generate the ROI in Eq. (3):

$$x_v = \text{ROI}(x_o(t), d_v, \alpha)$$  \hspace{1cm} (3)

The ROI function scales the width and height of the GT bounding box $d_v$ with a ratio $\alpha$, thus covers the nearby background for motion comparison. The ROI is then reshaped to size $64 \times 64$ to extract HOG features as given in Eq. (4):

$$x_h = \text{HOG}(x_v)$$  \hspace{1cm} (4)

where $x_h = x_h(t, k)$ denotes the MGF feature of the $k$th object at frame $t$.

**C. Training: Motion Classification**

Representation learning aims to find the low dimensional manifold of complex data using encoder-decoder architecture. The encoder learns a representation $z = z(t, k)$ of the $k$th object at frame $t$. The decoder reconstructs the input data $\hat{x}_h = \hat{x}_h(t, k)$ of the $k$th object at frame $t$. Given a input $x_h$, the encoder output $z(t, k)$ is computed in Eq. (5):

$$z = f(W_1x_h + b_1)$$  \hspace{1cm} (5)

where $W_1$ and $b_1$ are trainable weights and biases of encoder respectively. The decoder output is derived in Eq. (6):

$$\hat{x}_h = f(W_2z + b_2)$$  \hspace{1cm} (6)

where $W_2$ and $b_2$ are trainable weights and biases of decoder respectively. The autoencoder minimizes the reconstruction cost as shown in Eq. (7):

$$J_r = E_{x_h} \left[ ||x_h - \hat{x}_h||^2 \right]$$  \hspace{1cm} (7)

The SAE’s parameters are learned by greedily training single layer encoder-decoder architecture to minimize the reconstruction cost penalty with the previous layer’s representation as input data and stacking. Finally, SAES is fine-tuned by minimizing the reconstruction cost, followed by removing the decoder layers and training with the softmax layer for classification. Considering a single hidden layer SAE, the object motion classification is calculated in Eq. (8):

$$y_m = \text{softmax}(W_3z + b_3)$$  \hspace{1cm} (8)

where $y_m = y_m(t, k)$ is the motion of the $k$th object at frame $t$; $W_3$ and $b_3$ are trainable weights and biases of the final softmax layer respectively. We minimize the cross entropy loss for the object motion classification as shown in Eq. (9):

$$J = -E_{x_h} \left[ d_m \log(y_m) \right]$$  \hspace{1cm} (9)

where $d_m = d_m(t, k)$ is the GT value.

**D. Testing: Motion Object Detection**

In the testing phase, each frame is processed one at a time using the trained YOLOx object detector for detection, which is followed by MGF feature extraction on the detected objects. Finally, the object motion is predicted using the trained motion predictor. Given testing image $x \in D_t$ from testing dataset $D_t$, the YOLOx object detector output is presented in Eq. (10):

$$y_v, y_c, y_s = \text{YOLOx}(x)$$  \hspace{1cm} (10)

where $y_s = y(t, k)$ is the confidence value of the $k$th object at frame $t$. The bounding box $y_v$ with a confidence value larger than $\tau$ is selected to reduce false alarms. The MGF feature of the selected bounding box is derived as:

$$x'_h = \text{HOG} \left( \text{ROI}(x_o, y_v) \right)$$  \hspace{1cm} (11)

The motion of the $k$th object at frame $t$ is calculated as:
where $\hat{y}_m = \hat{y}_{m(t,k)}$ is the motion class of $k$th object at frame $t$; the output of the MEP is presented in Eq. (13):

$$O^{\text{MEP}}_k = (y_v(t,k), y_c(t,k), y_s(t,k), \hat{y}_m(t,k)) ;$$

$$k = 1, ..., K$$

where $O^{\text{MEP}}_k$ is the MEP output of the $k$th object at frame $t$; $K$ is the total detected object quantity at frame $t$. The pseudocode of the testing phase is presented in Algorithm 1.

**Algorithm 1**: Real-time Motion Object Detection for testing phase

**Input:**

1. YOLOx: the trained YOLOx detector from the training phase;
2. $W1, b1, W3, b3$: the trained SAE classifier parameters from the training phase;
3. $D_T$: testing dataset;
4. $x(t)$: input image at frame $t$;
5. $\tau$: manually set class confidence threshold;

**for** $x(t) \in D_T$ **do**

1. $x_o(t) = \text{OF}(x(t), x(t-1))$;
2. $K, y_v, y_c, y_s = \text{YOLOx}(x(t))$;
3. // $K$: the quantity of detected objects at frame $t$;
4. $y_v = [y_v(t,1), ..., y_v(t,K)]$: bounding box vectors;
5. $y_c = [y_c(t,1), ..., y_c(t,K)]$: class vectors;
6. $y_s = [y_s(t,1), ..., y_s(t,K)]$: class confidence vectors;

**for** $k = 1$ to $K$ **do**

7. if $y_v(t,k) > \tau$ then
8. $\hat{x}_h(t,k) = \text{HOG} (\text{ROI}(x_o, y_v(t,k)))$;
9. // $\hat{x}_h(t,k)$: MGF feature
10. $z'(t,k) = f(W_1 \hat{x}_h(t,k) + b_1)$;
11. $\hat{y}_m(t,k) = \text{softmax}(W_3 z'(t,k) + b_3)$;
12. // $\hat{y}_m(t,k)$: Motion class

else

13. $\hat{y}_m(t,k) = \emptyset$;
14. $y_c(t,k) = \emptyset$;
15. $y_v(t,k) = \emptyset$;
16. $y_s(t,k) = \emptyset$;

end

**Output:**

$O^{\text{MEP}}_k = (y_v(t,k), y_c(t,k), y_s(t,k), \hat{y}_m(t,k))$

end

**III. EXPERIMENTS**

Experiments are conducted to evaluate the performance of the proposed MEP architecture on the SG-MOD dataset and KITTI-MOD dataset. Its performances are compared to those of other pipeline based architecture methods including CMD, MBH, MPNet, and the end-to-end architecture method, MODNet. We also investigate the efficacy of the proposed MEP on the different scenarios and environment conditions in the SG-MOD dataset. All experiments are conducted in a server with a Xeon Gold 6128 3.4 GHz CPU with 256 GB memory and four Nvidia P100 12 GB GPU.

**A. Datasets and evaluation metrics**

We evaluate MEP on two datasets: (1) KITI-MOD; (2) SG-MOD. The public dataset KITTI-MOD [5] contains 1,950 frames, with a resolution of 1242×375 which provides both object bounding boxes and motion segmentation maps. It can be used to train both object detection and segmentation-based MOD algorithms. Since the KITI-MOD does not provide continuous frames for 200 images in the training data, the rest continuous frames are used for training purpose.

Existing public datasets contain videos in common on-road scenarios, but does not provide large variation in complex urban scenarios. They have large class imbalance problem for moving and static objects data. Hence, SG-MOD is created from urban driving environments with much better moving-static objects class ratio at about 1.2 : 1.

We create SG-MOD dataset by placing an in-vehicle camera with 1920×1080 resolution at 30 frames-per-second speed in front of the windshield of the driven vehicle. There are 447 urban driving videos with one minute length each collected. From these videos, number of 17,943 frames are selected in the SG-MOD dataset. We select 2/3 of the frames as training and 1/3 as testing. We provide only bounding box labels for training objects. Furthermore, SG-MOD also include large amount of different images in four scenarios with: (1) illumination changes (IC); due to building, underground driveways, passing by vehicles; (2) scale variations (SV) due to vehicles accelerating or deaccelerating; (3) turning vehicles (TV); (4) vehicles in junctions (VJ) as shown in Fig. 3. The number of frames available under the four scenarios in the SG-MOD dataset is listed in TABLE I.

In the experiments, the mean average precision ($mAP$) is adopted as the evaluation metric.
TABLE I: Data under four scenarios in the SG-MOD dataset.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Quantity (frame)</th>
<th>IC</th>
<th>SV</th>
<th>TV</th>
<th>VJ</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1483</td>
<td>8274</td>
<td>4256</td>
<td>3930</td>
<td>17943</td>
</tr>
</tbody>
</table>

TABLE II: Performance comparisons between our proposed MEP and other SOTA

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Methods</th>
<th>AP moving</th>
<th>AP static</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG-MOD</td>
<td>CMD [9]</td>
<td>65.53%</td>
<td>56.18%</td>
<td>60.86%</td>
</tr>
<tr>
<td></td>
<td>MBH</td>
<td>76.81%</td>
<td>66.64%</td>
<td>71.75%</td>
</tr>
<tr>
<td></td>
<td>Our MEP</td>
<td>79.82%</td>
<td>65.57%</td>
<td>72.70%</td>
</tr>
<tr>
<td>KITTI-MOD</td>
<td>MPNet [10]</td>
<td>50.23%</td>
<td>31.94%</td>
<td>41.03%</td>
</tr>
<tr>
<td></td>
<td>MODNet [5]</td>
<td>66.54%</td>
<td>58.0%</td>
<td>62.57%</td>
</tr>
<tr>
<td></td>
<td>MBH</td>
<td>44.79%</td>
<td>71.93%</td>
<td>58.36%</td>
</tr>
<tr>
<td></td>
<td>Our MEP</td>
<td>56.77%</td>
<td>75.47%</td>
<td>66.12%</td>
</tr>
</tbody>
</table>

B. Hyperparameter selection

The hyper-parameter selection is performed in a grid search. The parameters resulting in the best accuracy are selected. For the number of hidden neurons of the SAE, we choose the optimal number from [100, 100], [500, 500], [1000, 1000], [2000, 2000] and the dropout from [40%, 60%, 80%]. We set the number of iterations as 500 for the greedy layer-wise training, 1000 for the SAE fine-tuning. The learning rate \( \eta \) is set to 0.1. For the ROI selection, the value \( \alpha \) is set as \( \sqrt{2} \). In the HOG calculation, the cell size is set to 4; blocksize is set as 8. The stride for each block is set to 4 and the gradient in each cell is counted into 9 bins.

C. Comparison with the SOTA methods

TABLE II shows the comparison of the proposed MEP with other methods CMD [9], MBH, MPNet [10] and MODNet [5]. In the experiment, the IOU is set as 0.5 between predicted and GT bounding boxes for calculating \( mAP \). TABLE II shows that the proposed MEP outperforms CMD and MBH in terms of \( mAP \) by 11.84% and 0.97% respectively on the SG-MOD dataset. While the proposed MEP outperforms MPNet, MODNet, and MBH in terms of \( mAP \) by 25.09%, 3.55%, and 7.76% respectively on the KITTI-MOD dataset.

It is also observed in TABLE II that the proposed MEP outperforms MBH in terms of \( AP \) moving by 3.01% and 11.98% respectively on the SG-MOD and KITTI-MOD datasets. It shows that our MEP is suitable for recognizing moving objects as the MEP uses MGF features to capture motion boundaries describing relative motions of the object to background. While MBH only captures the motions along the \( x \) and \( y \)-axis without describing relative motions well.

TABLE II further shows that MBH outperforms MEP in terms of \( AP \) static by 0.93% on the SG-MOD dataset. We believe this is because the SG-MOD dataset contains slow-moving vehicles. The MGF feature is not good at capturing slow motions. It is also observed that our MEP achieves lower result of the \( AP \) moving by 9.77% than that of the MODNet on the KITTI-MOD dataset. We believe it is because KITTI-MOD dataset contain highly unbalanced class ratio and heavy illumination changes. Yet the proposed MEP still obtains 3.5% better in \( mAP \) than MODNet.

Furthermore, TABLE III shows that the MEP is about 30 times faster than CMD with optical flow estimation and 43.5% faster than MBH without optical flow estimation. It shows that the proposed MEP outperforms MPNet, MODNet, and MBH algorithms in most metrics on both datasets.

Examples of qualitative results of our MEP algorithm on the SG-MOD dataset are shown in Fig. 4: (a) shows the GT result using our motion annotations. (b) represents the optical flow image that describes the object’s motion in pixels. (c) demonstrates the motion modeling result based on our MGF feature, which clearly describes motion boundaries to facilitate motion prediction. (d) demonstrates the motion detection result using the MEP.

D. Comparison of different Motion Predictors

In this sub-section, the performances are compared with different algorithms for the motion predictor stage. We change the classifier in the motion predictor stage with a decision tree (DT), the SVM with a sigmoid kernel, and the random forest (RF). TABLE IV shows that the MEP outperforms RF by 5.74% and 0.33% for \( mAP \) on SG-MOD and KITTI-MOD datasets respectively. Furthermore, the MEP largely outperforms DT and SVM approaches in terms of \( mAP \). While the MEP obtains lower result of the \( AP \) static than the RF by 0.2% under the KITTI-MOD dataset. We believe it is because the MEP is better.
TABLE V: Motion detection performances under different scenarios in the SG-MOD dataset

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>AP moving</th>
<th>AP static</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC</td>
<td>67.34%</td>
<td>1.07%</td>
<td>34.21%</td>
</tr>
<tr>
<td>SV</td>
<td>81.78%</td>
<td>71.94%</td>
<td>76.86%</td>
</tr>
<tr>
<td>TV</td>
<td>80.45%</td>
<td>65.86%</td>
<td>73.16%</td>
</tr>
<tr>
<td>VJ</td>
<td>83.07%</td>
<td>54.7%</td>
<td>68.89%</td>
</tr>
</tbody>
</table>

at recognizing moving objects than static objects, thus it has a 0.86% improvement in AP moving. We could also observe that SVM outperforms MEP under the SG-MOD dataset by 4.07% in terms of AP static. It could be because the MEP may not successfully recognize some objects with slow motions. SVM is better at recognizing static objects but it works much poorer in moving object recognition. Overall, the MEP has the best performance among these classifiers. The SAE representation learning on MGF motion feature is superior to other non-representation learning approaches.

E. Evaluating the MEP on Four Scenarios

We evaluate the MEP efficacy on four scenarios of environment conditions: IC, SV, TV, and VJ of the developed SG-MOD dataset. The experiment performances are shown in TABLE V. It is observed that the MEP performs poorly in the IC scenario with only 34.21% at mAP. In contrast, it performs much better in SV, TV, and VJ scenarios that have better illumination conditions. We believe this is because the MGF feature has limitations at capturing the relative motion between objects and background under bad lighting conditions. Another relatively poor performance occurs on VJ scenarios. We believe this because many vehicles turn slowly in junctions, and the MEP is not good at recognizing these slow-moving vehicles.

IV. CONCLUSION

The MEP architecture has been proposed to recognize the motion of objects. The MEP is a pipeline-based architecture that consists of three stages. In the first stage, we find the objects using the YOLOX object detector. In the second stage the MGF extracts features of detected objects. While in the third stage, the SAE learns the representation and classifies the motion of the objects. The MGF feature captures the motion boundaries describing the relative motion between objects and background which is suitable to determine the motion of objects. The SG-MOD dataset has been created in this work by driving in urban environment of Singapore to capture a large amount of objects with changing speed due to acceleration, deacceleration, and turning. Various experiments have been conducted to evaluate the performances of the MEP. Experimental results show that our MEP outperforms other SOTA methods on SG-MOD and KITTI-MOD datasets in most metrics.

ACKNOWLEDGEMENTS

The authors would like to thank Yue Li, Xiaofan Jia, Zixiao Zhu, Jiatong Zhu and Tianhao Ang for helping data collections, and also to thank Urban Redevelopment Authority, Singapore for their supports in this research.

REFERENCES