A VT-HMM-Based Framework for Countdown Timer Traffic Light State Estimation

Shuyang Zhang, *Student Member, IEEE*, Qingwen Zhang, *Graduate Student Member, IEEE*, Feiyi Chen, Jin Wu, *Member, IEEE*, Jianhao Jiao, *Member, IEEE*, and Lujia Wang, *Member, IEEE*

***Abstract*— Traffic lights are important components of traffic systems, and perceptual tasks on traffic lights are crucial for intelligent agents on the road. Auxiliary countdown timers, providing the remaining time of the current traffic phase, improve the safety and smoothness of the entire traffic system. This work proposes a state estimation framework for countdown timer traffic lights. Time-domain information is adequately integrated into a variable transition Hidden Markov Model (VT-HMM), and our system provides optimal estimates of traffic light colors and countdown numbers based on noisy detection inputs. A dynamic state transition matrix is designed based on a 1-step transition logic and a probability of the number of transitions related to the current state sojourn duration. A recursive decoding method based on the Viterbi algorithm is proposed to update all the state candidates and select the optimal state chain. Extensive experiments evaluate the robustness and effectiveness of the proposed work. The performance boundaries of this system are also found under various input noise levels. The source code is available here: https://github.com/ShuyangUni/countdown-timer-traffic-light-estimation**

***Index Terms*— State estimation, countdown timer traffic light, hidden semi-Markov Model, probabilistic model, autonomous driving.**

I. Introduction

TRAFFIC light perception is fundamental but indispensable for autonomous driving. Like other traffic participants, autonomous vehicles recognize traffic indications to ensure their safety and smoothness in traffic flows. V2X (Vehicle-to-Everything) traffic lights, aiming to solve traffic light perception tasks by transforming these tasks into robust V2X communication [1], [2], [3], [4], are the current development trend. However, V2X traffic lights, suffering from their expensive retrofit costs, have the challenge of becoming popular in recent years. High-performance vehicle-end traffic light perception modules will still dominate the field in the future.

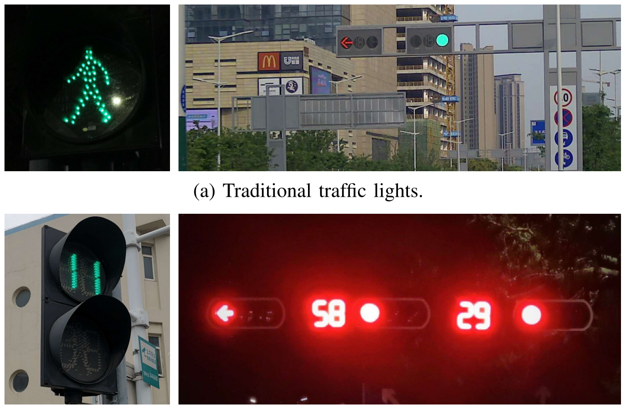

(a) Traditional traffic lights.
(b) Countdown timer traffic lights.

Fig. 1. Various traffic lights. (a) Traditional traffic lights only provide instantaneous hints. (b) Countdown timers provide the remaining time for current color phase.

Traditional traffic lights (Fig. 1a) only provide move and stop signals. Drivers wait for green lights and react after color switching. The brief pauses accumulated by their reactions in queues eventually cause traffic jams [5], [6].

Traffic lights with countdown timers (Fig. 1b), providing the rest time (in seconds) before traffic signal shifting, bring more benefits than the traditional ones. Red countdown displays cause seemingly less waiting time and help ease human driver anxiety [7], [8]. Green countdown signals allow drivers to make precise decisions on whether to stop before the traffic lights or pass through the intersection [9], [10], which finally reduces red light and yellow light violations. Meanwhile, human drives smarter with the help of countdown timers. Without violating traffic rules, human drivers release the brakes just before the end of red signals and have already speeded up when green signals arrive. The ability to respond in advance shortens the reaction time, significantly improving the efficiency of traffic systems and easing the formation of traffic jams. Autonomous vehicles can mimic human driving behaviors by sensing countdown timers. With a countdown time estimator, autonomous vehicles can also launch early and pass the stop line precisely at the arrival of the green light. Autonomous vehicles can be more intelligent and elegant like humans when placed in scenarios containing countdown timer traffic lights.

Manuscript received 18 March 2022; revised 18 September 2022, 7 December 2022, and 28 February 2023; accepted 4 April 2023. The work of Lujia Wang was supported by the Guangdong Basic and Applied Basic Research Foundation under Project 2021B1515120032. The Associate Editor for this article was Z. Duric. *(Corresponding author: Lujia Wang.)*

Shuyang Zhang, Feiyi Chen, Jin Wu, and Jianhao Jiao are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, SAR 999077, China (e-mail: szhangcy@connect.ust.hk).

Qingwen Zhang is with the Thrust of Robotics and Autonomous Systems, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511442, China.

Lujia Wang is with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, SAR 999077, China, and also with the Clear Water Bay Institute of Autonomous Driving, Nanshan, Shenzhen 518057, China (e-mail: eewanglj@ust.hk).

Digital Object Identifier 10.1109/TITS.2023.3265985

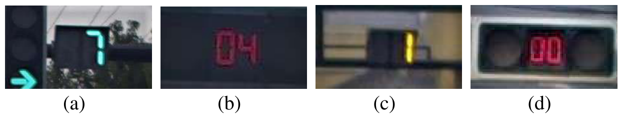


Fig. 2. Various digit patterns of countdown timer traffic lights. (a) **null** on tens place. (b) 0 on tens place. (c) Color ends with 1. (d) Color ends with 0.

State estimation for countdown timer traffic lights faces several challenges. **The system logics are complex.** Traditional traffic lights only provide color transition logic (red → green → yellow → red), while countdown timers provide underlying countdown information. After status shifting, we implicitly know the next transition will happen in about 1 second. Meanwhile, the countdown timer resets to an unknown value after color shifting. It is not easy to integrate these logics into a system using the language of probability. **There are various temporal patterns.** When displaying numbers less than 10, some countdown timers show nothing on tens place (Fig. 2a). Others replace the null displays with 0 (Fig. 2b). Some countdown timers terminate a color at 1 (Fig. 2c), and others end at 0 (Fig. 2d). It is challenging to design a system that adapts to the diversity of temporal patterns. **There are more input errors.** Errors from detection, reflected in the sizes and offsets of bounding boxes, seldom affect color classification in traditional traffic light systems. However, traffic light systems with countdown timers have extra digit classification sub-tasks, of which the pixel-level offsets may cause digit misclassification. Meanwhile, lighting conditions and motion blur increase the challenge of digit classification but rarely affects color classification. It is more challenging to consider the input errors from the digit classification sub-task than traditional traffic light systems.

In this work, we propose a state estimator for countdown timer traffic lights modeling by a Hidden semi-Markov Model (HSMM) to tackle the abovementioned challenges. Our system presents the following contributions:

1) An HSMM-based state estimator, which is a general framework for countdown timer traffic lights with various patterns (Section V). To the best of our knowledge, this paper is the first work on state estimation tasks for countdown timer traffic lights using temporal modeling.
2) Dynamic state transition modeling, which establishes the relationship between transition probability and state sojourn duration by a start time distribution (Section VI).
3) A recursive decoding method extended from the Viterbi algorithm, which updates the optimal chains for all state candidates and their corresponding start time distributions iteratively (Section VII).

II. Related Work

A. Traffic Light Perception in One Shot

From the perspective of computer vision, traffic light perception consists of two subtasks. One is traffic light detection, locating the device's position in each image frame. Another is traffic signal recognition, which recognizes control signals.

Traditional traffic light detection methods depended on color segmentation [11], [12], morphological characteristics [13], [14], or their combinations [15], [16]. Structured global features, such as Histogram of Oriented Gradients (HOG) [17], [18], Hough Transform [19], and self-defined templates [13], [20], are also selected in traditional machine learning frameworks. Deep learning techniques [21], [22], [23], [24], [25] were involved in traffic light detection tasks as well. Additional information, such as offline prior maps [19], [26], [27], [28] and configuration files [29], provides precise prior locations of traffic lights and significantly reduces difficulty under complex urban scenarios.

Traffic signal recognition is considered as a classification task in computer vision. Color space is the most crucial feature in distinguishing signals. RGB color space was commonly used, while other color spaces, such as Hue-Saturation-Value (HSV) [15], [18], [30], YUV [29], [31], and YCbCr [16], were selected to handle the luminance and chrominance separately. Signal status is defined according to the system design. In addition to the original three signal statuses (*red*, *green* and *yellow*/*amber*), the status *off* [14], [21] was added when traffic lights are turned off. A status *red*/*amber* [14] was considered for the alternate blinks. The not-detected status *N.D.* [30], or the background status *background* [21] was set for false recognition cases. Classifiers were also differently implemented, including color thresholds [12], [15], [29], [30], SVM [32], and AdaBoost [15]. CNN-based classifiers [18], [21], [24], benefiting from increasing datasets, provide higher precision than traditional ones.

Traffic light perception methods using a single image cannot provide temporally stable outputs. Results from different images give inconsistent detection outputs, and it is difficult to decide which outcome to trust. Our method estimates the traffic light signals using all the countdown observations and provides the optimal estimation over an entire image sequence.

B. Traffic Light Perception in Image Sequence

Temporal information from image sequences enhances the stability of traffic light detection in the time domain. Trackers attached after detectors provide temporal priors [15], [27], [14], [21]. Nienhüser et al. [14] considered the traffic light association and update into a multiple object tracking (MOT) problem. Gong et al. [15] adopted Continuously Adaptive Mean Shift (CAMSHIFT) to track traffic lights. Levinson et al. [27] modeled perceptual offsets using the Gaussian motion model and updated the belief under a histogram filter framework. Behrendt et al. [21] employed an odometry-based motion model and tracked traffic devices by a neural network.

Image sequences also enhance the classification accuracy of traffic signals. Temporal cues establish the connection of adjacent statuses and reduce misclassifications. Meanwhile, specific patterns in the time domain (such as blinking [14]) can only be defined in image sequences. Trehard et al. [33] considered the detection and recognition of each traffic light as object tracking tasks using the Interactive Multiple Model (IMM). Bach et al. [34] proposed a multiple traffic light recognition system using the Dempster-Shafer Theory of Evidence. Labeled Multi-Bernoulli filters were implemented to

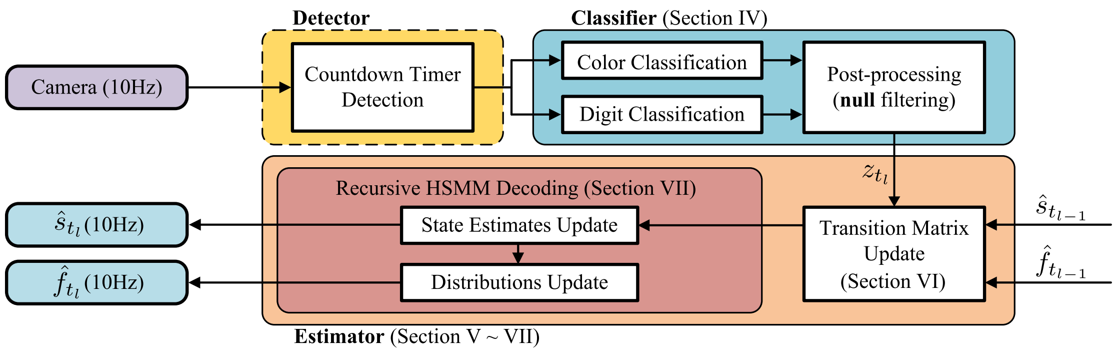


Fig. 3. Our System overview.

update the bounding boxes and color statuses simultaneously. Gómez et al. [30] described traffic light color transition rules using Hidden Markov Model (HMM) to improve the detection results. Nienhüser et al. [14] also considered modeling the temporal information by HMMs. and they specifically focused on the patterns of German traffic signals.

Similar to the work of [14] and [30], our method uses an HMM-based framework to enhance temporal stability. The difference is that we focus on the colored countdown numbers instead of only the three-color traffic signals. The status dimension in our problem is larger, and the system logic is more complex.

C. V2X-Based Traffic Light System

The problem sets of traffic light perception are different in the field of V2X communication. V2X communication systems provide more accurate and richer information to associated autonomous vehicles. The crucial points turn to what equipments for communication, such as visible light communication devices [35] and wireless devices [36], and what communication protocols to comply, such as Dedicated Short-range Communication (DSRC) [37] and IEEE 802.11p [38].

Signal Phase and Timing (SPaT) message [36], [39], [40], which describes the traffic light phase of a signalized intersection, is a solution perfectly matched to our problem. The SPaT messages are broadcasted to vehicles close to the intersection and provide the real-time signal phase with its remaining time. However, due to the expensive retrofit costs, our system will still be meaningful before the popularization of V2X devices.

D. Research on Countdown Timer Traffic Light Perception

The perception topics on countdown timer traffic lights are rarely studied. To our knowledge, there are only two related works in the literature. Sathiya et al. [41] used a segmentation-based color classifier on RGB space and implemented a 7-segment digit detector. Chaud et al. [42] proposed TSCTNet, an end-to-end deep learning framework for digit detection composed of Mask RCNN and RetinaNet. These two works both noticed the auxiliary role of countdown timers. However, they only considered the countdown timers under an object detection task, and did not consider the use of temporal cues.

In this paper, we propose a state estimator for countdown timer traffic lights, providing optimal estimates over time for both digits and colors. Inspired by the work of [14] and [30], we model the countdown logic into a Variable Transition HMM (VT-HMM) [43]. The dynamic transition matrix is probabilistically designed, based on the sojourn duration of the traffic light state. A recursive decoding method ensures the optimal state estimation over time.

III. System Overview

A. Problem Formulation

Let $z_{t_1:t_l}$ be the observations of countdown timer traffic lights from t_1 to t_l. These observations are composed of colors and digits and are independent of each other. The actual states $s_{t_1:t_l}$ are hidden and cannot be directly observed. The state estimation problem, finding the most likely estimate of the state sequence $\hat{s}_{t_1:t_l}$ given the available observations $z_{t_1:t_l}$, is modeled as a maximum a posteriori (MAP) problem

$$\hat{s}_{t_1:t_l} = \arg\max_{s_{t_1:t_l}} \; p(s_{t_1:t_l} \mid z_{t_1:t_l}). \tag{1}$$

In this paper, we choose the Hidden semi-Markov Model (HSMM), which contains sojourn durations for each state. Additional components $f_{t_1:t_l}$, which represent the start time distributions (detailed in Section V-A), are appended to our model. Our MAP problem in (1) is reformulated as

$$< \hat{s}_{t_1:t_l}, \hat{f}_{t_1:t_l} >= \arg\max_{s_{t_1:t_l}, f_{t_1:t_l}} \; p(s_{t_1:t_l}, f_{t_1:t_l} \mid z_{t_1:t_l}). \tag{2}$$

B. Overview of Proposed System

The overview of our system is shown in Fig. 3. Our perception system is separated into three sequential steps. First, a **detector**, implemented by Yolo_v5 [44], detects the region of countdown timers from raw images at each timestamp. After engineering optimization, it reaches 8ms/frame on NVIDIA GTX 1080Ti with TensorRT FP16 acceleration. In this paper, we concentrate on the system's temporal modeling and will not give the implementation details of our detector. Afterward, our **classifier** recognizes the colors and the digits separately. The color classifier distinguishes traffic light colors by HSV thresholds, while the digit classifier is based on a CNN-based digit classification network. Finally, an **estimator** estimates the hidden state using all available observations. The most likely state chain from t_1 to t_l noted as $\hat{s}_{t_1:t_l}$ is estimated by our recursive HSMM decoding method. The system output is represented as $\hat{s}_{t_l}$, the latest vertex of the best chain at t_l. Our system is online deployed on real platforms. The

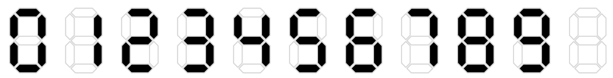


Fig. 4. Digit observations. Its space contains 0 to 9, which is equivalent to our digit state space. A **null** observation is appended to cover the unobserved cases on tens place.

details of our system's operating consumption are described in Section VIII-G.

Our proposed system is based on several assumptions. The essential assumption is that our system's input contains only detection and classification noise. For instance, bad weather and extreme lighting conditions may cause the disability of traffic light detection and classification over the whole sequence. These cases are not considered in this work. Meanwhile, we assume the countdown timers are double-digit, meaning only digits in tens and units places are considered. In the real world, we find triple-digit countdown timers, and some traffic light systems are specially designed so that they do not show the countdown digits until the last few seconds of a color. These special cases are not considered in this paper.

IV. Digit Traffic Light Classification

A. Color Classification

Similar to [30], our classifier distinguishes the traffic light colors by thresholds in the HSV color space. The color classification output is marked as $Z^c \in \mathbb{C}_z$. In addition to the three standard traffic light colors, a status **unknown** is designed for failure cases. In hue space, the color **green** ranges from $[50, 100]$ and **yellow** is bounded by $[21, 40]$. The color **red** is split into two parts. The lower part is $[0, 20]$ while the upper one is $[170, 180]$. In saturation and value space, the range of all three colors is set as $[50, 255]$.

We firstly transform the input image into HSV color space, and each pixel is classified with HSV thresholds. The pixel amount for each color is counted, and the majority color wins this vote. If the proportion sum of three colors is less than a threshold $t_r = 0.005$, the color status is marked as **unknown**.

B. Digit Classification

In this part, our classifier recognizes the 7-segment digits of countdown timers. We select a lightweight CNN with only two convolutional and two fully connected layers. The inputs of the CNN are countdown timer regions, and its output $Z^v \in \mathbb{D}_z$ has 11 classes, as shown in Fig. 4, including digits from 0 to 9 and a **null** class for the special cases on tens place. A pre-processing step is deployed before feeding pixels into the network. We convert the RGB image patch (detection bounding box) into grayscale and resize it to 32×32 pixels. Since we only consider the double-digit traffic lights, we vertically halve an input into tens and units places. The CNN classifier separately detects two digits, and its outputs are the inputs of our back-end state estimator.

C. Post-Processing

Before inputting the color and digit classification results into our VT-HMM state estimator, we filter a special observation

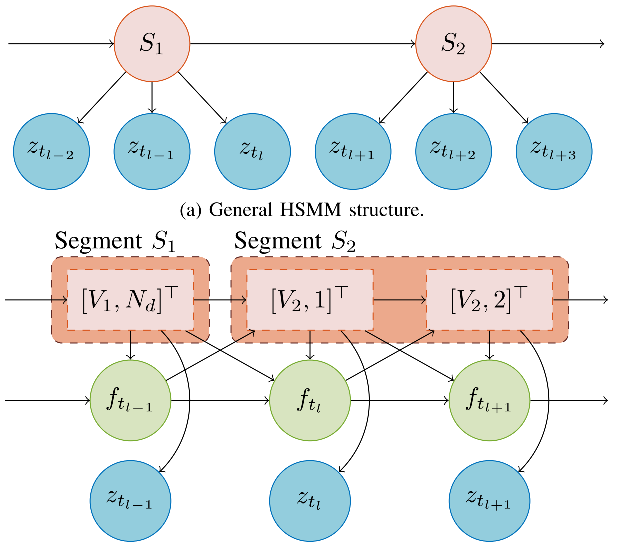


Fig. 5. HSMM structures. The red vertexes are the states, while the green ones are their corresponding start time distributions. The blue vertexes indicate the observations. The vertexes marked with dashed and solid lines represent hidden and observed variables.

with **unknown** color and **null** for both tens and digits places. This empty observation usually occurs in two situations. One is the traffic light blinking at the end of green lights, and another is due to misclassification. Ambiguity happens when designing the *emission probability* matrix. The system classifies some blinking observations as empty only because they are unobserved. They should have an equal probability of emitting all possible states. For misclassification cases, the emission possibility should relate to the differences between observations. For instance, digit 1 is more likely to be false-detected as **null** than digit 8.

It is demanding to balance both cases under a probabilistic model. We filter the empty observations before inputting classification results into our estimator. For blinking cases, no input provides no prior information. The **null** state is uniformly distributed, and meets the condition of equal probability to every state. For misclassification cases, although we lose the information from observations, the impact is controlled by our further estimator design.

V. HSMM-Based State Estimator

A. Variable Transition HMM

HSMM is an extension of a Hidden Markov Model (HMM), of which the hidden process is semi-Markovian. Each HSMM state maintains a period of time and can be correlated by a series of consecutive observations instead of a single one. For modeling convenience, an HSMM state is expanded as a sequence of states with the same status to ensure one-to-one correspondences with observations, called state segment. A general HSMM structure is shown in Fig. 5a.

Specifically, we choose the variable transition HMM (VT-HMM). The state $\boldsymbol{s}_{t_l}$ is extended as a 2-tuple vector $\boldsymbol{s}_{t_l} = [\boldsymbol{v}_{t_l}, d_{t_l}]^\top$. The current traffic light status $\boldsymbol{v}_{t_l}$ is a discrete random variable and will be described in Section V-B. The sojourn duration d_{t_l} since entering the current traffic light status is a continuous random variable. To reduce the searching complexity in a continuous space of d_{t_l}, we sample it into several discrete searching intervals via a system frequency f_{sys} (commonly 10Hz). The sampling ID is marked as D_i, and its corresponding sampling range is

$$d_{t_l} \in \left([d_{t_l}]_{\inf}, [d_{t_l}]_{\sup}\right] = \left(\frac{D_i - 1}{f_{\text{sys}}}, \frac{D_i}{f_{\text{sys}}}\right]. \tag{3}$$

For convenience, we simplify the expression as $d_{t_l} = D_i$.

The sojourn duration d_{t_l} is a hidden variable and cannot be directly observed. We introduce an intermediate random variable $t_{s,i}$, representing the start time of the current status, also the end time of the last status. The relationship between the start time $t_{s,i}$ and the sojourn duration d_{t_l} is $d_{t_l} = t_l - t_{s,i}$, where t_l is the observation timestamp. The modeling details of our random variables will be described in Section VI-A.

Since our method is under a HMM-based framework, the probability of every state chain $p(\boldsymbol{s}_{t_1:t_l} \mid \boldsymbol{z}_{t_1:t_l})$ in (1) needs to be calculated during each update step. In order to compute the state chain probability incrementally, we also update the distributions of the start time $t_{s,i}$ in update step. The distributions are complicated to be parameterized, and we represent them in a discrete form under a resolution of 0.01. The distribution over $t_{s,i}$ is globally updated and then normalized for each processing step. Afterward, it is truncated by the discrete sampling range D_i. The corresponding distribution component is marked as $F_{t_l,i} = f_{t_{s,i}}(t)$, and it pairwisely used with D_i in the remainder of this paper. The model structure of our VT-HMM is shown in Fig. 5b.

B. State Modeling and 1-Step Connection Matrix

The state of our VT-HMM is defined as $\boldsymbol{S}_i = [\boldsymbol{V}_i, D_i]^\top$. It should be emphasized that we only mark the discrete sampling interval D_i in our state, while its corresponding distribution $F_{t_l,i}$ is also utilized. The traffic light status $\boldsymbol{V}_i$ comprises a triplet $[V_i^c, V_i^{d_2}, V_i^{d_1}]^\top$, and $V_i^c \in \mathbb{C}_s$ is the color status. Since we focus on the double-digit countdown timers, only tens and units places are considered, representing as $V_i^{d_2}, V_i^{d_1} \in \mathbb{D}_s$. The cardinality of traffic light status is $N_v = 3 \times 10 \times 10$, and the entire volume of our state space is $N_s = N_v \times N_d$. The *transition probability* matrix $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{F}_{t_l}) \in \mathbb{P}^{N_s \times N_s}$ is a dynamic matrix. Its element is represented as

$$a_{ij} = p(\boldsymbol{s}_{t_{l+1}} = \boldsymbol{S}_j \mid \boldsymbol{s}_{t_l} = \boldsymbol{S}_i, f_{t_l} = F_{t_l,i}). \tag{4}$$

Due to camera orientation, traffic lights are sometimes unobservable for a few seconds, and their hidden states may alter several times without observation. The *transition probability* matrix should be time-related and need to be dynamically designed. In our system, we model the *transition probability* matrix $\boldsymbol{A}$ using the sojourn duration of the current state and a 1-step state connection matrix $\boldsymbol{C}^{(1)} \in \mathbb{N}^{N_v \times N_v}$. The details of how to calculate a_{ij} in (4) will be shown in Section VI.

The 1-step state connections $\boldsymbol{C}^{(1)}$ are shown in Fig. 6. An edge in $\boldsymbol{C}^{(1)}$ represents the number of chains from $\boldsymbol{V}_i$ to $\boldsymbol{V}_j$ with only a 1-step transition. It differs from the edge in a Markov Chain in which the value represents the transmit

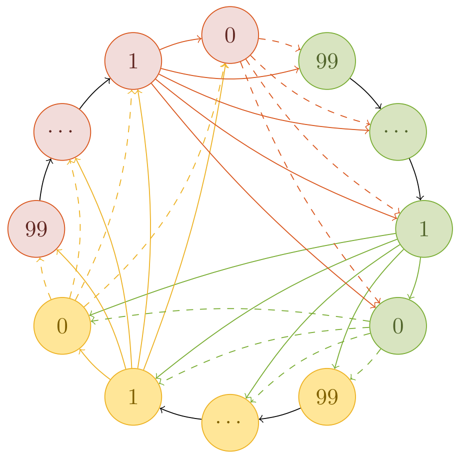


Fig. 6. Our 1-step state connection matrix. The vertex color represents color status, while the value represents the double-digit display. The transition edges of standard vertexes are in black. The connections from special vertexes to value 0 and 1 are marked as dashed and solid lines, respectively.

probability. There are two types of vertexes. The standard vertexes only have one output edge, which points to the next status following the countdown model. For instance, the vertex $[\textbf{red}, 2, 1]^\top$ can only be transformed to $[\textbf{red}, 2, 0]^\top$. The special vertexes are used at the moment of color shifting. The start number for each color varies from traffic light equipment settings. Since we do not load their start time priors offline in our system, we consider that each digit status starts with the same probability. Section I mentions that some traffic lights end the color at 0, while some end at 1. Therefore, the vertex for 0 has 100 output connections and 101 for 1.

One should be mentioned that the matrix dimension of $\boldsymbol{C}^{(1)}$ $(N_v \times N_v)$ is different from the one of $\boldsymbol{A}$ $(N_s \times N_s)$. The matrix $\boldsymbol{C}^{(1)}$ is time-independent and offline generated. The connection relationships come only from the inherent countdown logic of traffic lights, and is not affected by the sojourn duration D_i in our state. In Section VI, we use $\boldsymbol{A}$'s subscripts in the expression of $\boldsymbol{C}^{(1)}$ for convenience.

C. Observation Modeling

Our observation contains one color and two digits, representing as $\boldsymbol{Z}_k = [Z_k^c, Z_k^{d_2}, Z_k^{d_1}]^\top$. The cardinality of observation space is $N_z = 4 \times 11 \times 11$. The *emission probability* matrix $\boldsymbol{B} \in \mathbb{P}^{N \times M}$, also known as the *confusion matrix*, is static. Its component is represented as

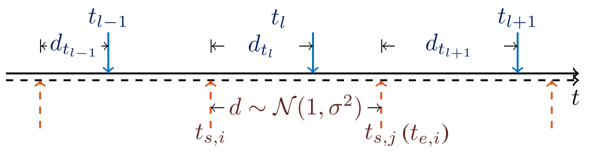
$$b_{ik} = p(z_{t_l} = \boldsymbol{Z}_k \mid \boldsymbol{s}_{t_l} = \boldsymbol{S}_i). \tag{5}$$

Since its three observation components are independent, the *emission probability* matrix $\boldsymbol{B}$ is decomposed as $\boldsymbol{B}^c \otimes \boldsymbol{B}^{d_2} \otimes \boldsymbol{B}^{d_1}$, where $\otimes$ is the Kronecker matrix product. The element of $\boldsymbol{B}$ in (5) is decomposed as

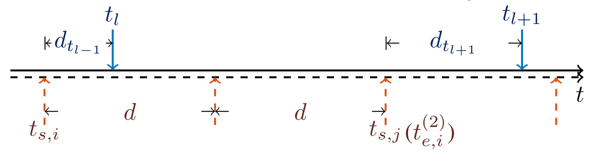
$$\begin{aligned} b_{ik} &= p(Z_k^c, Z_k^{d_2}, Z_k^{d_1} \mid V_i^c, V_i^{d_2}, V_i^{d_1}) \\ &= p(Z_k^c \mid V_i^c) \times p(Z_k^{d_2} \mid V_i^{d_2}) \times p(Z_k^{d_1} \mid V_i^{d_1}). \end{aligned}$$

The color *emission probability* matrix $\boldsymbol{B}^c \in \mathbb{P}^{3\times 4}$ is

$$\boldsymbol{B}^c = \begin{bmatrix} 0.85 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.85 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{bmatrix},$$



(a) Our common random variable modeling.



(b) Our random variable modeling with 2-step transition.

Fig. 7. Our random variable visualization. The arrows in solid blue represents the observation timestamp, while the ones in dashed red represent the switching time of traffic light statuses.

where its last column is the probability from each state to **unknown** observation. The elements of $\boldsymbol{B}^c$ are manually set by experience. The 7-segment digits are encoded as 7-bit binary strings. Their differences are measured by Hamming distance, and the difference matrix for units place digit is

$$\boldsymbol{H} = \begin{bmatrix} 0 & 4 & 3 & 3 & 4 & 3 & 2 & 3 & 1 & 2 & 6 \\ 4 & 0 & 5 & 3 & 2 & 5 & 6 & 1 & 5 & 4 & 2 \\ 3 & 5 & 0 & 2 & 5 & 4 & 3 & 4 & 2 & 3 & 5 \\ 3 & 3 & 2 & 0 & 3 & 2 & 3 & 2 & 2 & 1 & 5 \\ 4 & 2 & 5 & 3 & 0 & 3 & 4 & 3 & 3 & 2 & 4 \\ 3 & 5 & 4 & 2 & 3 & 0 & 1 & 4 & 2 & 1 & 5 \\ 2 & 6 & 3 & 3 & 4 & 1 & 0 & 5 & 1 & 2 & 6 \\ 3 & 1 & 4 & 2 & 3 & 4 & 5 & 0 & 4 & 3 & 3 \\ 1 & 5 & 2 & 2 & 3 & 2 & 1 & 4 & 0 & 1 & 7 \\ 2 & 4 & 3 & 1 & 2 & 1 & 2 & 3 & 1 & 0 & 6 \end{bmatrix}.$$

Its element h_{ik} represents the numbers of substitutions from $\boldsymbol{S}_i^{d_1}$ to $\boldsymbol{Z}_k^{d_1}$. Afterward, the *emission probability* matrix for units place $\boldsymbol{B}^{d_1} \in \mathbb{P}^{10\times 11}$ is derived by projecting $\boldsymbol{H}$ into the probability space, as $\boldsymbol{B}^{d_1} = \exp(-\alpha * \boldsymbol{H})$, where α is a hyper-parameter to control the magnitude of emission probability. Each row of $\boldsymbol{B}^{d_1}$ needs to be normalized. The *emission probability* matrix for the tens places $\boldsymbol{B}^{d_2} \in \mathbb{P}^{10\times 11}$ is almost identical to as $\boldsymbol{B}^{d_1}$. The only difference is that **null** appears instead of 0 in some traffic lights for the tens digits. The distance between 0 to **null** should be eliminated, so we change $\boldsymbol{H}(1, 11) = 0$ for $\boldsymbol{B}^{d_2}$.

D. Initialization

The *start probability* of our VT-SMM model is $\boldsymbol{\Pi} \in \mathbb{P}^{N_s}$. Since each state has an equivalent probability of being visited at the beginning, the initial value for each element of $\boldsymbol{\Pi}$ is set as $\pi_i = \frac{1}{N_s}$. The start time distribution $\boldsymbol{F}_{t_l}$ is separately calculated and maintained for each state samplings. Since there is no prior information for the initial frame, we initialize $\boldsymbol{F}_{t_1}$ as a uniform distribution according to each sampling sojourn duration D_i.

VI. Dynamic State Transition Probability

A. n-Step Transition Modeling

According to the countdown logic, the sojourn duration for each traffic light status should be 1 second. However, this cannot be guaranteed due to production errors. Therefore, we model the sojourn duration as a Gaussian random variable $d \sim \mathcal{N}(1, \sigma^2)$, where σ is the duration uncertainty. Considering the start time of the i^{th} traffic light status as $t_{s,i}$, the end time $t_{e,i}^{(1)}$ after 1-step transition, also the start time of the next state, is a random variable. Their relationship is $t_{e,i}^{(1)} = t_{s,i} + d$, and is visualized in Fig. 7a. The probabilistic density function (PDF) of $t_{e,i}^{(1)}$ is the convolution between the distributions of random variables $t_{s,i}$ and d, where

$$f_{t_{e,i}^{(1)}}(t) = (f_{t_{s,i}} * f_d)(t) = \int_{-\infty}^{\infty} f_{t_{s,i}}(\tau) f_d(t-\tau) d\tau.$$

The operator $(*)$ is the convolution operator.

As we mentioned in Section V-B, traffic lights may be miss detected, and the system jumps over several statuses without observation. Similar to 1-step transition cases, we model the end time for an n-step transition $t_{e,i}^{(n)}$ as the combination of $t_{s,i}$ and n independent random variable d, representing

$$t_{e,i}^{(n)} = t_{s,i} + \underbrace{d + \cdots + d}_{n}. \qquad (6)$$

A visualization of a 2-step transition case is shown in Fig. 7b. The PDF of the n-step transition in (6) is calculated as

$$f_{t_{e,i}^{(n)}}(t) = (f_{t_{s,i}} * \underbrace{f_d * \cdots * f_d}_{n})(t) = (f_{t_{s,i}} * f_{dn})(t), \qquad (7)$$

where $f_{dn}(t) = \mathcal{N}(n, n\sigma^2)$. Its cumulative distribution function (CDF) is

$$F_{t_{e,i}^{(n)}}(t) = (f_{t_{s,i}} * F_{dn})(t), \qquad (8)$$

where $F_{dn}(t)$ is the CDF of $f_{dn}(t)$. The PDFs and CDFs of $t_{e,i}^{(n)}$ with $n = 1, \ldots, 5$ are visualized in Fig. 8. A special case of $n = 0$, meaning no transition happens, remains the previous distribution, where $f_{t_{e,i}^{(0)}}(t) = f_{t_{s,i}}(t)$.

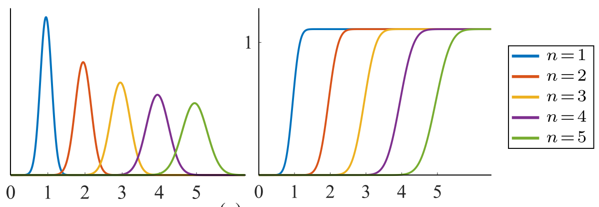


Fig. 8. PDFs and CDFs of $t_{e,i}^{(n)}$. In this visualization, the start time follows a uniform distribution $t_{s,i} \sim \mathcal{U}(-0.1, 0.0)$ and the Gaussian uncertainty is set as $\sigma = 0.15$.

B. The Probability of the Number of Transitions

In this section, we calculate the probability

$$p(n_{ij} \mid F_{t_l,i}, t_{l+1}) = p(n \mid \boldsymbol{s}_{t_l} = \boldsymbol{S}_i, \boldsymbol{s}_{t_{l+1}} = \boldsymbol{S}_j, F_{t_l,i}), \qquad (9)$$

representing the chance of that a n-step transition happens between t_l and t_{l+1} with the start time distribution $F_{t_l,i}$. We change the view to a random variable $t_{e,i}$, the end time of $\boldsymbol{S}_i$ and also the start time of $\boldsymbol{S}_j$. It is intuitive to find $t_{e,i}^{(n)} = t_{s,j} = t_{l+1} - d_{t_{l+1}}$. The sojourn duration $d_{t_{l+1}}$ is updated at time t_{l+1} and is determined by the sampling range D_j in $\boldsymbol{S}_j$ using (3). The estimated end time $\hat{t}_{e,i}$ is decomposed as

$$\begin{aligned}\hat{t}_{e,i} &\in \left[\left[\hat{t}_{e,i}\right]_{\inf}, \left[\hat{t}_{e,i}\right]_{\sup}\right) \\ &= \left[t_{l+1} - \left[d_{t_{l+1}}\right]_{\sup}, t_{l+1} - \left[d_{t_{l+1}}\right]_{\inf}\right).\end{aligned}$$

By definition in (6), the start time $t_{s,i}$ and $t_{s,j}$ ought to differ by $d \times n$ exactly ($n = 0$ without any transition, $n = 1$ if adjacent statuses are both observed, and $n > 1$ if missing observations of several statuses). We set $t_{e,i}^{(n)} = \hat{t}_{s,i} + d \times n$, where $\hat{t}_{s,i}$ is the estimated start time calculated in our recursive decoding method from the last update, and it contains prior information on all the past observations. The probability in (9) is transferred to the chance that $t_{e,i}^{(n_{ij})}$ is bounded by $\hat{t}_{e,i}$,

$$\begin{aligned}p(n_{ij} \mid F_{t_l,i}) &= \Pr(\left[\hat{t}_{e,i}\right]_{\inf} < t_{e,i}^{(n_{ij})} \le \left[\hat{t}_{e,i}\right]_{\sup}) \\ &= F_{t_{e,i}^{(n_{ij})}}(\left[\hat{t}_{e,i}\right]_{\sup}) - F_{t_{e,i}^{(n_{ij})}}(\left[\hat{t}_{e,i}\right]_{\inf}), \quad (10)\end{aligned}$$

where $F_{t_{e,i}^{(n_{ij})}}$ is derived from (8) using the sampling start time distribution $F_{t_l,i} = f_{t_{s,i}}(t)$.

We set a maximum number of transitions $n_{\max}$ to constrain the transition range. If the duration of losing observation is too long that $n_{\max}$-step transitions happen, the system restarts because the uncertainty $\sigma^2 n_{\max}$ is considered extremely high.

C. Dynamic State Transition Matrix

We mark the 0-step connection matrix as $\boldsymbol{C}^{(0)} = \boldsymbol{I}$, indicating no transition happens. The n-step connection matrix $\boldsymbol{C}^{(n)}$ is calculated by the 1-step connection matrix $\boldsymbol{C}^{(1)}$, where

$$\boldsymbol{C}^{(n)} = \underbrace{\boldsymbol{C}^{(1)} \times \cdots \times \boldsymbol{C}^{(1)}}_{n}.$$

The element in $\boldsymbol{C}^{(n)}$ may exceed 1 because there exists several possible paths when crossing different colors. For instance, there are two possible chains from $[\textbf{red}, 0, 1]^\top$ to $[\textbf{green}, 2, 1]^\top$ with a 2-step transition. One possible intermediate state is $[\textbf{red}, 0, 0]^\top$ while the other is $[\textbf{green}, 2, 2]^\top$. The element of our *transition probability* matrix in (4) is decomposed as

$$\begin{aligned}a_{ij} &= \max\left\{\Pr(\boldsymbol{S}_i \to \boldsymbol{S}_j \text{with } n_{ij}\text{-transitions})\right\}_{n_{ij}} \\ &= \max\left\{\mathbb{I}(c_{ij}^{(n_{ij})} > 0) \cdot p(n_{ij} \mid t_{l+1}, F_{t_l,i})\right\}_{n_{ij}}, \quad (11)\end{aligned}$$

where $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if the input is true and 0 otherwise. The estimated number of transitions from the i^{th} to the j^{th} state at t_l is

$$n_{ij} = \arg\max_{n_{ij}} \left\{\mathbb{I}(c_{ij}^{(n_{ij})} > 0) \cdot p(n_{ij} \mid t_{l+1}, F_{t_l,i})\right\}_{n_{ij}}. \quad (12)$$

The distribution $F_{t_l,i}$ is different for each row of $\boldsymbol{A}(\boldsymbol{F}_{t_l})$. We calculate $\boldsymbol{A}(\boldsymbol{F}_{t_l})$ row by row rather than using matrix operations. Each row of $\boldsymbol{A}(\boldsymbol{F}_{t_l})$ is independently normalized.

VII. Recursive HSMM Decoding Method

In this section, we introduce an recursive decoding method for our VT-HMM model based on the Viterbi algorithm [45].

A. Model Factorization

The decoding problem, formulated in (2), is transformed as

$$\begin{aligned}< \hat{\boldsymbol{s}}_{t_1:t_l}, \hat{\boldsymbol{f}}_{t_1:t_l} > &= \arg\max_{\boldsymbol{s}_{t_1:t_l}, \boldsymbol{f}_{t_1:t_l}} p(\boldsymbol{s}_{t_1:t_l}, \boldsymbol{f}_{t_1:t_l} \mid \boldsymbol{z}_{t_1:t_l}) \\ &= \arg\max_{\boldsymbol{s}_{t_1:t_l}, \boldsymbol{f}_{t_1:t_l}} \frac{p(\boldsymbol{s}_{t_1:t_l}, \boldsymbol{f}_{t_1:t_l}, \boldsymbol{z}_{t_1:t_l})}{p(\boldsymbol{z}_{t_1:t_l})} \\ &= \arg\max_{\boldsymbol{s}_{t_1:t_l}, \boldsymbol{f}_{t_1:t_l}} p(\boldsymbol{s}_{t_1:t_l}, \boldsymbol{f}_{t_1:t_l}, \boldsymbol{z}_{t_1:t_l}). \quad (13)\end{aligned}$$

An incremental expression decomposes the joint distribution in (13). If the previous estimates $\hat{\boldsymbol{s}}_{t_1:t_l}$ and $\hat{\boldsymbol{f}}_{t_1:t_l}$ are given, the estimate $\hat{\boldsymbol{s}}_{t_{l+1}}$ and $\hat{f}_{t_{l+1}}$ are represented as

$$\begin{aligned}&< \hat{\boldsymbol{s}}_{t_{l+1}}, \hat{f}_{t_{l+1}} > \\ &= \arg\max_{\boldsymbol{s}_{t_{l+1}}, f_{t_{l+1}}} p(\boldsymbol{s}_{t_{l+1}}, f_{t_{l+1}}, \hat{\boldsymbol{s}}_{t_1:t_l}, \hat{\boldsymbol{f}}_{t_1:t_l}, \boldsymbol{z}_{t_1:t_{l+1}}) \\ &= \arg\max_{\boldsymbol{s}_{t_{l+1}}, f_{t_{l+1}}} p(\boldsymbol{s}_{t_{l+1}}, f_{t_{l+1}}, \boldsymbol{z}_{t_{l+1}} \mid \hat{\boldsymbol{s}}_{t_1:t_l}, \hat{\boldsymbol{f}}_{t_1:t_l}, \boldsymbol{z}_{t_1:t_l}) \\ &= \arg\max_{\boldsymbol{s}_{t_{l+1}}, f_{t_{l+1}}} p(f_{t_{l+1}} \mid \hat{f}_{t_l}, \boldsymbol{s}_{t_{l+1}}, \hat{\boldsymbol{s}}_{t_l}) \\ &\qquad p(\boldsymbol{z}_{t_{l+1}} \mid \boldsymbol{s}_{t_{l+1}}) p(\boldsymbol{s}_{t_{l+1}} \mid \hat{\boldsymbol{s}}_{t_l}, \hat{f}_{t_l}). \quad (14)\end{aligned}$$

Like the classical Viterbi algorithm, for a possible state sampling, our decoding algorithm calculates the chain with the highest probability of ending at this certain state under the observation $\boldsymbol{z}_{t_l}$. After updating all the states, the one with the highest probability chain is chosen as the estimation result. We define a time-variant vector $\boldsymbol{\delta}_{t_l}$, whose element $\delta_{t_l,i}$ represents the maximum probability for all the possible chains from t_1 to t_l with the i^{th} state at time t_l. For the initial frame, the maximum probability $\delta_{t_1,i}$ is set as $\delta_{t_1,i} = \pi_i b_{ik}$, and the initial start time distributions $\boldsymbol{F}_{t_1,i}$ is uniformly distributed. For other frames, we firstly estimate $\hat{\boldsymbol{s}}_{t_{l+1}}$ in (14) with

$$\hat{\boldsymbol{s}}_{t_{l+1}} = \arg\max_{\boldsymbol{s}_{t_{l+1}}} p(\boldsymbol{z}_{t_{l+1}} \mid \boldsymbol{s}_{t_{l+1}}) p(\boldsymbol{s}_{t_{l+1}} \mid \hat{\boldsymbol{s}}_{t_l}, \hat{f}_{t_l}),$$

and then update $\hat{f}_{t_{l+1}}$ in (14) with

$$\hat{f}_{t_{l+1}} = \arg\max_{f_{t_{l+1}}} p(f_{t_{l+1}} \mid \hat{f}_{t_l}, \hat{\boldsymbol{s}}_{t_{l+1}}, \hat{\boldsymbol{s}}_{t_l}).$$

Algorithm 1 shows our decoding method for a single update.

B. Distribution Update

Suppose the estimated state $\hat{\boldsymbol{s}}_{t_l} = \boldsymbol{S}_i$ and its corresponding distribution $\hat{f}_{t_l} = \boldsymbol{F}_{t_l,i}$, the estimated state at t_{l+1} is $\hat{\boldsymbol{s}}_{t_{l+1}} = \boldsymbol{S}_j$. The distributions are updated under a Bayes filter framework. In the propagation step, the start time distribution $\bar{f}_{t_{l+1}}$ is predicted with f_{t_l}. There may exist several chains from state $\boldsymbol{S}_i$ to $\boldsymbol{S}_j$ with different numbers of transitions. The optimal n_{ij} is derived via (12), and the predicted distribution $\bar{f}_{t_{l+1}}$ is calculated by (7). In the update step, $\hat{\boldsymbol{f}}_{t_{l+1}}$ is calculated with $\bar{\boldsymbol{f}}_{t_{l+1}}$. Since the distribution of each sojourn duration is bounded by its sampling interval, we model the observation distribution within the corresponding sampling interval as a

Algorithm 1 One Step of Our Recursive HSMM Decoding Method

```
Input: $\delta_{t_l}$, $\boldsymbol{F}_{t_l}$, $z_{t_{l+1}} = \boldsymbol{Z}_k$, $\boldsymbol{B}$,
Output: $\delta_{t_{l+1}}$, $\boldsymbol{F}_{t_{l+1}}$, $\hat{s}_{t_{l+1}}$
1  Calculate $\boldsymbol{A}(\boldsymbol{F}_{t_l})$;
2  foreach $j \in \{1, \ldots, N_s\}$ do
       // Update $\delta_{t_{l+1},j}$
3      $\delta_{t_{l+1},j} = \max_{1 \le i \le N_s} [\delta_{t_l,i} \cdot a_{ij}] b_{jk}$ ;
4      $i_{\max} = \arg\max_{1 \le i \le N_s} \delta_{t_l,i} \cdot a_{ij}$ ;
       // Update $\boldsymbol{F}_{t_{l+1},j}$
5      if $\boldsymbol{S}_j = \boldsymbol{S}_{i_{\max}}$ then
           // No state shifting
6          $\boldsymbol{F}_{t_{l+1},j} = \boldsymbol{F}_{t_l,i_{\max}}$;
7      else
           // State shifting
8          Propagate $\bar{\boldsymbol{F}}_{t_{l+1},i_{\max}}$ ;
9          Update $\boldsymbol{F}_{t_{l+1},j}$ by (15) and (16) ;
10     end
11 end
12 $j_{\max} = \arg\max_{1 \le j \le N} \delta_{t_{l+1},j}$ ;
13 $\hat{s}_{t_{l+1}} = \boldsymbol{S}_{j_{\max}}$.
```

uniform distribution, where

$$f^z_{t_{l+1}} = \begin{cases} \dfrac{1}{[\hat{t}_{e,i}]_{\sup} - [\hat{t}_{e,i}]_{\inf}}, & [\hat{t}_{e,i}]_{\inf} < t \le [\hat{t}_{e,i}]_{\sup}, \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

The start time distribution is estimated as

$$\hat{f}_{t_{l+1}} = \eta \cdot f^z_{t_{l+1}} \cdot \bar{f}_{t_{l+1}}, \quad (16)$$

where η is the normalization to ensure the property of a PDF.

VIII. Experimental Evaluation

A. Dataset

For experimental evaluation, we collected a toy dataset with 10 sequences. The total frame size is 1702. The first 7 sequences are collected by a camera module (Fig. 9b) attached to our autonomous vehicle platform (Fig. 9a), daily operated in Wuhu, China. The rest 3 sequences are captured by a smartphone in Shenzhen, China. The capture frequency of both devices is 10Hz. We split the dataset into two levels according to lighting conditions and image quality:

- The *normal* part contains Sequence 01, 05, 06, 08, 09, and 10. The image patches of countdown timers are over 40×40 pixels. Images are captured under good lightning conditions, and digits are clear to be recognized.
- The *hard* part contains Sequence 02, 03, 04, and 07. Since our platform is far from the traffic lights, the countdown timer regions have averagely 20×20 pixels. Some sequences are captured at dusk, and these images are in low contrast. Some digits are blurred due to the platform's motion.

The stability of the input detection boxes affects the classification and estimation modules. To simulate the errors of bounding box detection, we add random pixel offsets on our manually-labeled detection ground truth, with different magnitudes of 0-pixel, 3-pixel and 5-pixel errors, respectively.

(a) Our platform.

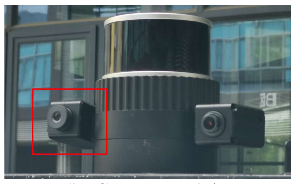

(b) Camera module.

Fig. 9. Our platform and camera module. The Sensing SG2-AR0231C-0202-GMSL-H30S (marked in red box) is used to collect our toy dataset. This camera has a horizontal field of view (FOV) of 30° and a vertical FOV of 16°. The camera's effective focal length (EFL) is 11.9mm.

B. Implementation Details

The parameter settings in our system are as follows. System frequency f_{sys} is set as 10Hz, equal to the data capturing frequency. The hyper-parameter for digit observation $\alpha = 4$, and the maximum number of transitions $n_{\max} = 5$. The number of duration samples N_d is related to the sojourn duration d and its uncertainty $\sigma = 0.15$. The sample range ought to cover all possible duration, and we use the 3-sigma empirical rule of Gaussian distribution. The maximum number of required samples is $N_d = (\mathbb{E}(d) + \sigma \times 3)/(1/f_{\text{sys}}) = 13$.

The distribution update step in Section VII-B is extremely time-consuming. For each update, our system needs to calculate the distributions N_s times (practically $N_s = 3900$ with our parameter settings), which contains $N_s \times n_{\max} = 19500$ times of convolution operations. In order to run the system online with limited computing resources, we also provide a simplified version of our system implementation. We simplify the distribution update step by assuming that the sampling distribution at any time is uniformly distributed. The distribution's range is fixed to the sampling interval $\frac{1}{f_{\text{sys}}}$ and the distribution is represented as $\mathcal{U}(0, \frac{1}{f_{\text{sys}}})$ after shifting the lower boundary to 0. The convolutions between uniform and Gaussian distributions are calculated offline to save time. The comparison of time consumption between these two versions will be given in Section VIII-G. We also provide a theoretical discussion on our simplification later in that section.

C. Evaluation Metrics

We introduce two metrics to assess the performance of our traffic light estimation system:

- **Overall Accuracy (OA)** is the portion of the positive estimations among the total number of estimations, where
$$\text{OA} = \frac{\text{number of positives}}{\text{number of total frames}}.$$
- **Keyframe Accuracy (KA)** is the portion of the positive keyframe estimations among the total number of keyframe estimations, representing as
$$\text{KA} = \frac{\text{number of keyframe positives}}{\text{number of total key frames}}.$$
The keyframes are frames within 5 seconds before traffic lights switch colors, during which human drivers tend to react in advance.

We also separately evaluate the performance of color and digit on these two metrics, nominated as $\mathbf{OA}_c$, $\mathbf{OA}_d$, $\mathbf{KA}_c$, and $\mathbf{OA}_d$.

TABLE I
ACCURACY EVALUATION ON REAL-WORLD DATA

Datasets	Methods	Overall Accuracy									Keyframe Accuracy								
		0-pixel			3-pixel			5-pixel			0-pixel			3-pixel			5-pixel		
		OA_c	OA_d	OA	OA_c	OA_d	OA	OA_c	OA_d	OA	KA_c	KA_d	KA	KA_c	KA_d	KA	KA_c	KA_d	KA
Normal	Baseline	**0.999**	0.487	0.487	**0.999**	0.387	0.387	**0.999**	0.298	0.298	**0.997**	0.109	0.109	**0.997**	0.141	0.141	**0.997**	0.112	0.112
	Our-C	**0.999**	0.966	0.966	**0.999**	0.942	0.942	**0.999**	0.786	0.786	**0.997**	0.994	0.994	**0.997**	0.984	0.984	**0.997**	0.824	0.824
	Our-C+E	**0.999**	**0.972**	**0.972**	**0.999**	**0.983**	**0.983**	0.995	**0.931**	**0.931**	**0.997**	**0.997**	**0.997**	**0.997**	**0.997**	**0.997**	0.984	**0.946**	**0.946**
Hard	Baseline	**1.000**	0.240	0.240	**1.000**	0.104	0.104	**1.000**	0.065	0.065	**1.000**	0.169	0.169	**1.000**	0.055	0.055	**1.000**	0.015	0.015
	Our-C	**1.000**	0.859	0.859	**1.000**	0.717	0.717	**1.000**	0.442	0.442	**1.000**	0.811	0.811	**1.000**	0.736	0.736	**1.000**	0.388	0.388
	Our-C+E	0.994	**0.945**	**0.939**	0.996	**0.892**	**0.890**	0.898	**0.615**	**0.607**	**1.000**	**0.960**	**0.960**	0.995	**0.915**	**0.915**	0.831	**0.577**	**0.577**

TABLE II
SEQUENCE-WISE ABLATION STUDY

Detection Errors	Methods	*Normal*						*Hard*			
		01	05	06	08	09	10	02	03	04	07
0-pixel	Our-C	**.956 / 1.00**	**1.00 / 1.00**	**1.00 / 1.00**	**1.00 / 1.00**	**.988 / .980**	.889 / .979	.540 / .294	.967 / .956	.960 / **1.00**	**.960 / 1.00**
	Our-C+E	.951 / **1.00**	**1.00 / 1.00**	**1.00 / 1.00**	**1.00 / 1.00**	**.988 / .980**	**.939 / 1.00**	**.871 / .863**	**.984 / .978**	**.988 / 1.00**	.921 / **1.00**
3-pixel	Our-C	.905 / **1.00**	.975 / .957	.993 / **1.00**	**1.00 / 1.00**	**.988 / .980**	.872 / .979	.557 / .471	.803 / .733	.740 / .915	.788 / .828
	Our-C+E	**.978 / 1.00**	**.994 / 1.00**	**1.00 / 1.00**	**1.00 / 1.00**	**.988 / .980**	**.956 / 1.00**	**.839 / .824**	**.967 / .956**	**.902 / .957**	**.887 / .931**
5-pixel	Our-C	.697 / .660	.658 / .681	.842 / .854	.913 / .898	.944 / .959	.828 / .957	.298 / .137	.459 / .378	.457 / .511	.536 / .517
	Our-C+E	**.898** / .840	**.857 / .928**	**.987 / .979**	**.984 / .959**	**.988 / .980**	**.939 / 1.00**	**.395 / .353**	**.574 / .422**	**.694 / .787**	**.695 / .655**

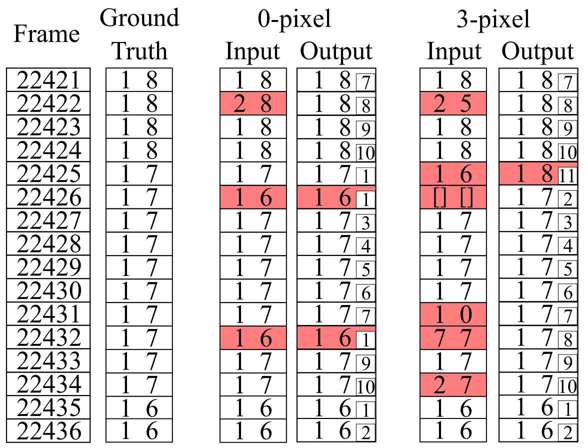
Each element represents a pair of OA / KA.

D. Evaluation: Real-World Experiments

1) Evaluation Methods: The work on countdown timer traffic lights has rarely been researched before. As mentioned in Section II, we have found only two works on this topic from Sathiya et al. [41] and Chaud et al. [42]. We design a **Baseline** method referring to Sathiya's work. We replace the RGB thresholds in their color classification part with our HSV thresholds. In the digit classification part, we first transform RGB images to grayscales and then binarize the images using Otsu's method [46]. Afterward, we use their seven key regions and encode the status of the LED tubes as a vector. Finally, the 7-bit vector is classified as 11 classes in $\mathbb{D}_z$ by finding the minimum difference in Hamming distance. We do not reproduce TSCTNet from Chaud as a comparison baseline because we cannot ensure its performance without their training data and parameter tuning details.

We nominate our complete state estimation system as **Our-C+E**. To ablatively demonstrate the performance of our estimator, we evaluate the intermediate results given by our classifier (**Our-C**), which is also the input of our estimator. The evaluation results on real-world data are shown in Table I.

2) Color Accuracy: The color results of Baseline are equal to Our-C's because they share the same color classification module. Their OA_c and KA_c are uncorrelated to detection errors because our color classifier uses overall color voting and is resistant to large detection offsets. Differently, our estimator jointly estimates the colors with digits. The color accuracies of Our-C+E are correlated with digit accuracies and are slightly lower than the previous two methods. In *normal* sequences with no detection offset, the color accuracy is equivalent to the other two because the input digit accuracy is extremely



Frame	Ground Truth	0-pixel Input	0-pixel Output	3-pixel Input	3-pixel Output
22421	1 8	1 8	1 8 $\boxed{7}$	1 8	1 8 $\boxed{7}$
22422	1 8	2 8	1 8 $\boxed{8}$	2 5	1 8 $\boxed{8}$
22423	1 8	1 8	1 8 $\boxed{9}$	1 8	1 8 $\boxed{9}$
22424	1 8	1 8	1 8 $\boxed{10}$	1 8	1 8 $\boxed{10}$
22425	1 7	1 7	1 7 $\boxed{1}$	1 6	1 8 $\boxed{11}$
22426	1 7	1 6	1 6 $\boxed{1}$	[] []	1 7 $\boxed{2}$
22427	1 7	1 7	1 7 $\boxed{3}$	1 7	1 7 $\boxed{3}$
22428	1 7	1 7	1 7 $\boxed{4}$	1 7	1 7 $\boxed{4}$
22429	1 7	1 7	1 7 $\boxed{5}$	1 7	1 7 $\boxed{5}$
22430	1 7	1 7	1 7 $\boxed{6}$	1 7	1 7 $\boxed{6}$
22431	1 7	1 7	1 7 $\boxed{7}$	1 0	1 7 $\boxed{7}$
22432	1 7	1 6	1 6 $\boxed{1}$	7 7	1 7 $\boxed{8}$
22433	1 7	1 7	1 7 $\boxed{9}$	1 7	1 7 $\boxed{9}$
22434	1 7	1 7	1 7 $\boxed{10}$	2 7	1 7 $\boxed{10}$
22435	1 6	1 6	1 6 $\boxed{1}$	1 6	1 6 $\boxed{1}$
22436	1 6	1 6	1 6 $\boxed{2}$	1 6	1 6 $\boxed{2}$

Fig. 10. Case study of Sequence 01 with 0-pixel and 3-pixel detection errors. The two digits in each block represents the digits in tens and units places. The character '[]' indicates the **null**. The color statuses are *red* and we omit the color information to clearly display. The block marked as red represents the fault estimates. The number in the lower right corner indicates the estimated sojourn duration sampling ID.

high. If we add 5-pixel random offsets, the OA_c of Our-C+E drops to 0.995 due to an input digit accuracy of 0.786. This phenomenon is more evident in *hard* sequences, where the OA_c drops to 0.898 with an input digit accuracy of 0.442.

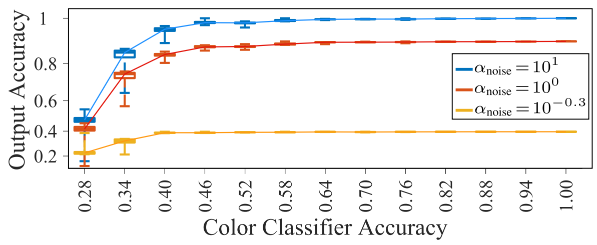
3) Digit Accuracy: Unlike color accuracies, digit accuracies are highly related to detection errors. When the simulated detection offsets augment, the digit accuracies for all three methods reduce, both in *normal* or *hard* sequences. Digit accuracies of *hard* sequences are generally lower than those of *normal* sequences. First, the *hard* sequences have more blurry images, directly affecting CNN classifier results. Second, the *hard* sequences are captured far from the traffic lights. The detection box size of *hard* sequences (approximately 20×10 pixels) is much smaller than the *normal* ones (generally 50×30 pixels and can be up to 100×50 pixels for sequence 08, 09, and 10). Digit accuracy in *hard* sequences drops more severely with the augmentation of detection offsets.

Among the three methods, Our-C+E best performs on digit accuracies for all data cases. Our-C provides inferior results to Our-C+E but is better than Baseline. Baseline suffers from a high expectation of the input detection boxes because it needs to wrap the entire digits within the boxes and has a low tolerance to noise. Meanwhile, the input digits for Baseline should be upright because deformations caused by capturing perspective are invalid. Therefore, the overall digit accuracy of Baseline is only 0.487, even in *normal* sequences without detection error. Our-C performs well for low detection noise (0-level and 3-level) in *normal* sequences, and Our-C+E has limited performance gains. However, for 5-level noise in *normal* sequences and all the *hard* sequences, Our-C+E has significantly improved from Our-C (maximumly from 0.442 to 0.615 of OA_d in *hard* sequences, 5-pixel noise).

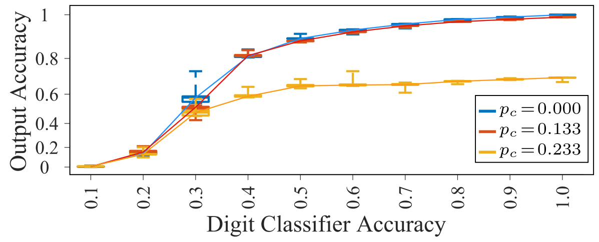
4) Total Accuracy: Our-C+E best performs among the three methods in overall and keyframe accuracies. Even for the most challenging cases (*hard* sequences, 5-pixel noise), the OA of Our-C+E still improves 39.14% (from 0.442 to 0.607) than Our-C's. The OA and KA of Our-C are equal to its OA_d and KA_d because its color classification is hardly wrong, and the major influence derives from the digit classification faults. The OA and KA of Our-C+E are slightly lower than its digit accuracies because of the color accuracy decline we mentioned before. However, the impact is limited, and the OA and KA of Our-C+E are still higher than those of Our-C.

To better study the performance improvement, we make an ablative comparison on OA and KA between Our-C and Our-C+E. The sequence-wise results are shown in Table II. For most sequences, Our-C+E enhances the output of Our-C and significantly improves the accuracies, meaning that Our-C+E efficiently corrects a certain portion of false classifications. In some special cases (Sequence 01 and 07), Our-C+E performs worse than Our-C. At the beginning of a sequence, our estimator has not enough input to stabilize the estimation over time. When classification faults happen at this stage, it takes several frames to recover from these erroneous inputs, and the estimates during the correction procedure are wrong.

5) Case Study: A special case can be found in Table II. In Sequence 01, our estimator reaches an overall accuracy of 0.978 with 3-pixel detection errors, while the one with 0-pixel detection errors is only 0.951. This phenomenon is caused by the system falsely detecting digit 7 as 6. A fragment of the estimation chain is given in Fig. 10. Although the input sequence with 3-pixel errors has more false detections, our estimator can correct most of these errors. For the 0-pixel case, our estimator cannot find a reasonable hidden chain through the frequent changes from 17 to 16. The estimator believes that the current optimal chain was wrongly chosen and switches it several times during this sequence fragment. This can be solved by training a powerful digit classifier.



(a) Relationship between input color accuracy and system output accuracy. The boxes in blue, red and yellow represent the simulations using digit classifiers with digit noise of 10^1, 10^0 and $10^{-0.3}$. Their equivalent accuracies (OA_d) are 1.000, 0.517 and 0.279 respectively.



(b) Relationship between input digit accuracy and system output accuracy. The boxes in blue, red and yellow represent the simulations using color classifiers with p_c of 0.000, 0.133 and 0.233.

Fig. 11. Misclassification simulation.

E. Evaluation: Misclassification Simulations

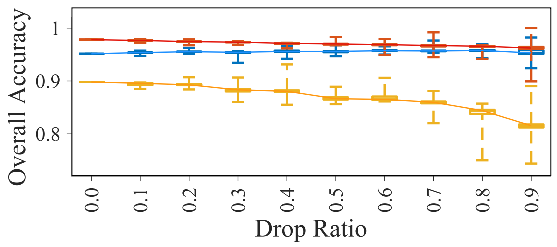
1) Simulation Settings: To test the performance boundaries of our estimator, we generate synthetic datasets by sampling the color and digit classification outputs with different noise levels. The synthetic datasets are generated from our dataset's longest sequence (Sequence 01). The color noise level is defined following the emission probability, derived from the color emission matrix in (6). It is nominated as p_c, and the color noise emission matrix is represented as

$$\boldsymbol{B}^{c}_{\text{noise}} = \begin{bmatrix} 1 - p_c \times 3 & p_c & p_c & p_c \\ p_c & 1 - p_c \times 3 & p_c & p_c \\ p_c & p_c & 1 - p_c \times 3 & p_c \end{bmatrix}.$$

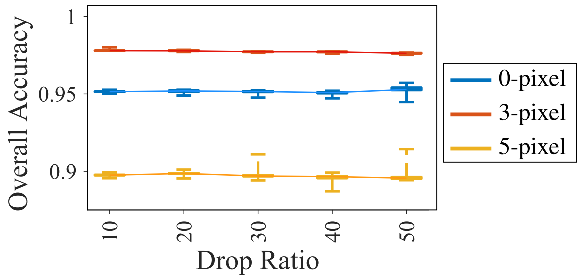
The probability of reserving the ground truth in the generated sequence is $1 - p_c \times 3$. Otherwise, it can jump to any other color, all with a probability of p_c. The digit noise level, nominated as α_{noise}, is designed by changing the hyper-parameter α. Its corresponding digit emission matrices for tens and units places are $\boldsymbol{B}^{d_2}_{\text{noise}}$ and $\boldsymbol{B}^{d_1}_{\text{noise}}$, respectively. For convenience, we express α_{noise} in the base of 10.

During the simulation, we fix one type of noise level and then traverse another to imitate the outputs of different classifiers. For each color-digit noise pair, we sample 50 synthetic sequences. The best color classifier has a color noise level of $p_c = 0$, and the best digit one shares a digit noise level of $\alpha_{\text{noise}} = 10^1$ (with the average self-emission possibility of 1.000). The worse classifiers follow a random guess test, of which $p_c = 0.25$ for color and $\alpha_{\text{noise}} = 10^{-2}$ (with the average self-emission possibility of 0.094, close to 1/11) for digit.

2) Color Noise Simulation: In color noise simulation, we traverse p_c to imitate the outputs of different color classifiers. Overall accuracy (OA) statistics are shown in Fig. 11a. Our estimator performs better under massive color noise. When color classifier accuracy is high (over 0.64), OA is hardly affected by color noise. Without digit noise ($\text{OA}_d = 1.000$),

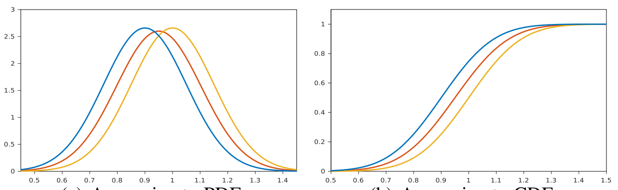


(a) Relationship between random drop ratio and output accuracy.



(b) Relationship between drop frames and output accuracy.

Fig. 12. Data missing simulation. The phenomenon that 0-pixel accuracies are lower than 3-pixel ones is explained by the case study in Section VIII-D.



(a) Approximate PDF (b) Approximate CDF.

Fig. 13. Approximation of input distributions. The distribution in red is used in our simplified version, a convolution between $\mathcal{U}(-0.1, 0)$ and $\mathcal{N}(0, 0.15^2)$. We also give the distribution by replacing the uniform function to Direc Delta functions at $t = -0.1$ (blue) and $t = 0$ (yellow), constructing the possible boundary in our standard version.

our estimator give an accuracy of over 0.9 even when the color classifier only has an accuracy of 0.4. Our estimator has a sudden performance drop when the color classifier almost fails (below 0.4), which can be considered a system failure.

3) Digit Noise Level Evaluation: The relationship between input digit accuracy and output accuracy is shown in Fig. 11b. Our estimator has good performance under certain digit noise. It provides stable and precise results when the digit accuracy exceeds 0.5. Similar to the evaluation of color noise, the output accuracy of our estimator decreases when digit noise increases.

4) Noise Factor Comparison: Comparing the performance on color and digit noise, a better digit classifier more severely impact our estimator. The sudden drop starts with a digit accuracy of 0.4, and our estimator thoroughly failed (OA close to 0) with a digit classifier of 0.1. For color noise, the system resists noise and provides at least an output accuracy of 0.2.

Three principal reasons make our estimator more sensitive to digit noise than color noise. First, our classification module is a one-color but two-digit classifier. When digit noise increases, the probability that the results of both digit classifiers are correct is much lower than a single one. Second, color classification is an easier task than digit classification. Color is more likely to be correctly ‘guessed’ when the color is difficult to be classified. Meanwhile, the switching logic of digits is more complicated than colors, increasing state estimation’s difficulty. Third, colors change less frequently than digits. Digit combination changes per second, while colors can last for dozens of seconds. Our estimator stabilizes color estimation by repeatedly observing colors within a period. Fast-switching digits with certain noise make our estimator provide several candidate chains with similar probabilities.

F. Evaluation: Data Missing Simulations

1) Random Loss Simulation: We evaluate our system’s ability to resist accidental data loss. Sequence 01 is chosen as the test sequence, and we randomly drop its frames to generate 50 samples for each missing ratio. We also evaluate the data missing resistance with different levels of detection errors. The relationship between system overall accuracies and missing ratios is shown in Fig. 12a. Our system is stable with low detection errors (0-pixel and 3-pixel), and the output accuracy hardly drops. In 5-pixel cases, the output accuracy decreases with the data loss ratio, but it still maintains a relatively high accuracy of over 0.8, even when losing 90% of input frames. When the noise level is high, the information that our estimator needs is also high. The lack of adequate information causes this accuracy decrease in retrieving the hidden chain from detection noises. When dropping a large number of frames, the system cannot receive enough observations to determine the hidden chain, finally leading to our system’s accuracy drop.

2) Long-Term Loss Simulation: We also simulate long-term data missing cases caused by the change of the camera’s view or the transient lags from camera drivers. Sequence 01 is chosen. We randomly select a starting frame and drop the following 10, 20, 30, 40, and 50 frames of the starting frame. The results of long-term data missing are shown in Fig. 12b. Our system is hardly affected by the long-term data missing cases. Although our system fluctuates in the presence of detection errors, the performance has limited degradation because the internal timer of the traffic light is accurate. The cumulative error for 5 seconds count is insignificant and is perfectly covered by our sojourn duration modeling.

G. Evaluation: System Simplification

We execute our countdown timer traffic light system on a PC with an Intel Core i5–8600K@3.60 GHz CPU and 16G RAM. The system of our PC is Ubuntu 18.04. The classifier is implemented in Python without GPU devices, and the average time consumption is 1.012ms per image. The estimator is implemented in C++, and the time consumption of the original implementation is averagely 205.25ms per frame. For the simplified version, the time consumption is reduced to 80.45ms, which guarantees the real-time performance at 10Hz.

To compare the output accuracies of these two implementations, we select Sequence 01 and 02 to compare their performance differences. In Sequence 01, both implementations have the same output accuracies on the three detection errors. In Sequence 02, the result of the complete implementation has a little drop to 0.294 with 5-pixel detection errors. However, this difference cannot indicate the pros and cons because both implementations have already failed in this sequence.

For a theoretical discussion, both implementations share comparable results due to our parameter settings. Two principal factors contribute to their implementation equivalence. First, the end time distribution $f_{t_{e,i}}$, is Gaussian-like distributed (Fig. 8). The input distribution can hardly affect the output distribution, and the end time distribution is similar to the initial Gaussian distribution. The PDF and CDF of the output distribution using a uniform input distribution are visualized in Fig. 13. We also give the output distributions by replacing the uniform inputs to Dirac Delta functions $\delta(x)$ and $\delta(x+0.1)$, which have impulses at 0 and -0.1 respectively. The output CDF from $\delta(x+0.1)$ is the upper bound of all possible input distributions, while the one of $\delta(x)$ constructs the lower bound. According to (10) and (11), we respectively calculate a_{ij} with these three distributions and find that the largest difference of a_{ij} can be 0.0505 at time 0.83. The difference in probability is insignificant and can hardly impact our system. Second, we update the distribution of a sampling interval by cutting a portion from the previous output distribution. Considering the $3-\sigma$ law, the valid range of the Gaussian distribution for a 1-step jump is $\sigma \times 6 = 0.9$, which is 9 times larger than the cutting interval (0.1 in our parameter settings). This phenomenon is more obvious during n-step jumps, and makes the cutting distributions flat. Therefore, the input distributions can be approximated as uniform distributions.

IX. Conclusion

This paper proposes a state estimation system for countdown timer traffic lights, which embeds temporal countdown cues into the VT-HMM framework. Dynamic transition relationships are probabilistically built, and the best color and digit combination estimate is decoded under an recursive approach. Our system has been evaluated by extensive experiments and shown its stability and noise immunity over time. Simulations have also studied misclassification and data missing cases. A simplified version is provided for real-time performance.

There are also several limitations of our work. First, our method only treats detection and classification errors. It cannot handle the failure sequences under bad weather and extreme lighting conditions. Second, our system is hard to be online if the countdown timer has over three digits, or if more than one countdown timers need to be processed.

Two directions for future research are proposed. One is to propose novel modelings or lightweight decoding algorithms to handle the huge state cardinality of our states. Another is to evaluate system output's confidence online and actively change processing frequency. The system can actively drop its processing frequency to save computing resources.

References

[1] M. Brambilla, M. Nicoli, G. Soatti, and F. Deflorio, "Augmenting vehicle localization by cooperative sensing of the driving environment: Insight on data association in urban traffic scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1646–1663, Apr. 2020.

[2] F. Weinert, M. During, and K. Bogenberger, "Influence of emergency vehicle preemption on travelling time and traffic safety in urban environments enabled by innovative behavioral models and V2X communication—Simulation and case study," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 4043–4048.

[3] Q. Lu, H. Jung, and K.-D. Kim, "Optimization-based approach for resilient connected and autonomous intersection crossing traffic control under V2X communication," *IEEE Trans. Intell. Vehicles*, vol. 7, no. 2, pp. 354–367, Jun. 2022.

[4] S. Aoki and R. Rajkumar, "Cyber traffic light: Safe cooperation for autonomous vehicles at dynamic intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 22519–22534, Nov. 2022.

[5] Y. Guan, S. E. Li, J. Duan, W. Wang, and B. Cheng, "Markov probabilistic decision making of self-driving cars in highway with random traffic flow: A simulation study," *J. Intell. Connected Vehicles*, vol. 1, no. 2, pp. 77–84, Dec. 2018.

[6] W. Zhu, J. Wu, T. Fu, J. Wang, J. Zhang, and Q. Shangguan, "Dynamic prediction of traffic incident duration on urban expressways: A deep learning approach based on LSTM and MLP," *J. Intell. Connected Vehicles*, vol. 4, no. 2, pp. 80–91, Sep. 2021.

[7] A. Sobota, M. J. Klos, and G. Karoń, "The influence of countdown timers on the traffic safety of pedestrians and vehicles at the signalized intersection," in *Intelligent Transport Systems and Travel Behaviour*. Cham, Switzerland: Springer, 2017, pp. 13–21.

[8] J. Jatoth, N. K. Singh, and A. Mehar, "Evaluating the performance of signalized intersection with signal countdown timer," *Int. J. Intell. Transp. Syst. Res.*, vol. 19, no. 1, pp. 182–190, Apr. 2021.

[9] S. Yu and Z. Shi, "Analysis of car-following behaviors considering the green signal countdown device," *Nonlinear Dyn.*, vol. 82, nos. 1–2, pp. 731–740, Oct. 2015.

[10] T. Krukowicz, K. Firląg, J. Suda, and M. Czerliński, "Analysis of the impact of countdown signal timers on driving behavior and road safety," *Energies*, vol. 14, no. 21, p. 7081, Oct. 2021.

[11] C. Wang, T. Jin, M. Yang, and B. Wang, "Robust and real-time traffic lights recognition in complex urban environments," *Int. J. Comput. Intell. Syst.*, vol. 4, no. 6, pp. 1383–1390, Dec. 2011.

[12] M. Diaz-Cabrera, P. Cerri, and J. Sanchez-Medina, "Suspended traffic lights detection and distance estimation using color features," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Apr. 2012, pp. 1315–1320.

[13] R. De Charette and F. Nashashibi, "Traffic light recognition using image processing compared to learning processes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 333–338.

[14] D. Nienhüser, M. Drescher, and J. M. Zöllner, "Visual state estimation of traffic lights using hidden Markov models," in *Proc. 13th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2010, pp. 1705–1710.

[15] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, and H. Chen, "The recognition and tracking of traffic lights based on color segmentation and CAMSHIFT for intelligent vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 431–435.

[16] W. Zong and Q. Chen, "Traffic light detection based on multi-feature segmentation and online selecting scheme," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2014, pp. 204–209.

[17] Z. Shi, Z. Zou, and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 690–700, Mar. 2016.

[18] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj, and N. Ganeshan, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 606–611.

[19] F. Lindner, U. Kressel, and S. Kaelberer, "Robust recognition of traffic signals," in *Proc. IEEE Intell. Vehicles Symp.*, May 2004, pp. 49–53.

[20] Z. Chen, Q. Shi, and X. Huang, "Automatic detection of traffic lights using support vector machine," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 37–40.

[21] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1370–1377.

[22] M. B. Jensen, K. Nasrollahi, and T. B. Moeslund, "Evaluating state-of-the-art object detector on challenging traffic light data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 9–15.

[23] J. Müller and K. Dietmayer, "Detecting traffic lights by single shot detection," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 266–273.

[24] M. Weber, P. Wolf, and J. M. Zöllner, "DeepTLR: A single deep convolutional network for detection and classification of traffic lights," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 342–348.

[25] O. Jayasinghe et al., "Towards real-time traffic sign and traffic light detection on embedded systems," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2022, pp. 723–728.

[26] N. Fairfield and C. Urmson, "Traffic light mapping and detection," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 5421–5426.

[27] J. Levinson, J. Askeland, J. Dolson, and S. Thrun, "Traffic light mapping, localization, and state detection for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 5784–5791.

[28] V. John, K. Yoneda, B. Qi, Z. Liu, and S. Mita, "Traffic light recognition in varying illumination using deep learning and saliency map," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 2286–2291.

[29] N. Wu and H. Fang, "A novel traffic light recognition method for traffic monitoring systems," in *Proc. 2nd Asia–Pacific Conf. Intell. Robot Syst. (ACIRS)*, Jun. 2017, pp. 141–145.

[30] A. E. Gomez, F. A. R. Alencar, P. V. Prado, F. S. Osorio, and D. F. Wolf, "Traffic lights detection and state estimation using hidden Markov models," in *Proc. IEEE Intell. Vehicles Symp. Proc.*, Jun. 2014, pp. 750–755.

[31] A. Almagambetov, S. Velipasalar, and A. Baitassova, "Mobile standards-based traffic light detection in assistive devices for individuals with color-vision deficiency," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1305–1320, Jun. 2015.

[32] C.-C. Chiang, M.-C. Ho, H.-S. Liao, A. Pratama, and W.-C. Syu, "Detecting and recognizing traffic lights by genetic approximate ellipse detection and spatial texture layouts," *Int. J. Innov. Comput. Inf. Control*, vol. 7, no. 12, pp. 6919–6934, 2011.

[33] G. Trehard, E. Pollard, B. Bradai, and F. Nashashibi, "Tracking both pose and status of a traffic light via an interacting multiple model filter," in *Proc. IEEE Int. Conf. Inf. Fusion*, May 2014, pp. 1–7.

[34] M. Bach, S. Reuter, and K. Dietmayer, "Multi-camera traffic light recognition using a classifying labeled multi-Bernoulli filter," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1045–1051.

[35] S. Arai et al., "Experimental on hierarchical transmission scheme for visible light communication using LED traffic light and high-speed camera," in *Proc. IEEE 66th Veh. Technol. Conf.*, Sep. 2007, pp. 2174–2178.

[36] N. Varga, L. Bokor, A. Takács, J. Kovács, and L. Virág, "An architecture proposal for V2X communication-centric traffic light controller systems," in *Proc. 15th Int. Conf. ITS Telecommun. (ITST)*, May 2017, pp. 1–7.

[37] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of DSRC and cellular network technologies for V2X communications: A survey," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9457–9470, Dec. 2016.

[38] J. Gozálvez, M. Sepulcre, and R. Bauza, "IEEE 802.11p vehicle to infrastructure communications in urban environments," *IEEE Commun. Mag.*, vol. 50, no. 5, pp. 176–183, May 2012.

[39] *Dedicated Short Range Communications (DSRC) Message Set Dictionary*. SAE Standard SAE J2735, DSRC Committee, 2015.

[40] S. Ibrahim, D. Kalathil, R. O. Sanchez, and P. Varaiya, "Estimating phase duration for SPaT messages," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 7, pp. 2668–2676, Jul. 2019.

[41] S. Sathiya, M. Balasubramanian, and D. V. Priya, "Real time recognition of traffic light and their signal count-down timings," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, Feb. 2014, pp. 1–6.

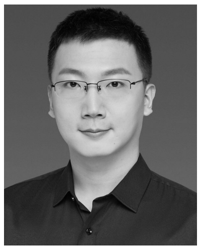
[42] D. Chand, S. Gupta, and I. Kavati, "TSCTNet: Traffic signal and countdown timer detection network for autonomous vehicles," Tech. Rep., Dec. 2020, doi: 10.13140/RG.2.2.15753.06243.

[43] S.-Z. Yu, "Hidden semi-Markov models," *Artif. Intell.*, vol. 174, no. 2, pp. 215–243, Feb. 2010.

[44] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.

[45] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[46] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. SMCS-9, no. 1, pp. 62–66, Feb. 1979.



Shuyang Zhang (Student Member, IEEE) was born in Yangzhou, Jiangsu, China, in 1995. He received the B.Eng. and M.Phil. degrees from Xi'an Jiaotong University, Xi'an, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree with the Intelligent Autonomous Driving Center (IADC), The Hong Kong University of Science and Technology, Hong Kong, China. His research interests include robotic perception, state estimation, and machine learning.

Qingwen Zhang (Graduate Student Member, IEEE) was born in Changsha, Hunan, China, in 1997. She received the B.Sc. degree from the Civil Aviation University of China, Tianjin, China, in 2020, and the M.Phil. degree from The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, in 2022. She is currently pursuing the Ph.D. degree with the KTH Royal Institute of Technology, Stockholm, Sweden. Her research interests include motion planning and end-to-end imitation learning.

Feiyi Chen was born in Enshi, Hubei, China, in 1996. He received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2019, and the M.Phil. degree from The Hong Kong University of Science and Technology, Hong Kong, China, in 2022. His research interests include multi-sensor fusion, calibration, and computer vision.

Jin Wu (Member, IEEE) was born in Zhenjiang, Jiangsu, China, in 1994. He received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China. He is currently pursuing the Ph.D. degree with the Intelligent Autonomous Driving Center (IADC), The Hong Kong University of Science and Technology, Hong Kong, China. He was with Tencent Robotics X, Shenzhen, China, from 2019 to 2020. His research interests include robot navigation, multi-sensor fusion, automatic control, and mechatronics.

Jianhao Jiao (Member, IEEE) was born in Zhongshan, Guangdong, China, in 1994. He received the B.Eng. degree in instrument science from Zhejiang University, Hangzhou, China, in 2017, and the Ph.D. degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China, in 2021. His research interests include state estimation, SLAM, sensor fusion, and computer vision.

Lujia Wang (Member, IEEE) received the Ph.D. degree from the Department of Electronic Engineering, The Chinese University of Hong Kong, Sha Tin, Hong Kong, in 2015. From 2016 to 2021, she was an Associate Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong. She is currently a Research Assistant Professor with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. Her current research interests include cloud robotics, lifelong federated robotic learning, and applications on autonomous driving.