ALGORITHM AND ARCHITECTURE DESIGN OF A KNOWLEDGE-BASED VEHICLE TRACKING FOR INTELLIGENT CRUISE CONTROL

Yi-Min Tsai, Chih-Chung Tsai, Keng-Yen Huang, and Liang-Gee Chen

DSP/IC Design Lab., Graduate Institute of Electronics Engineering,
National Taiwan University, Taipei, Taiwan
{ymtsai,cctsai,kyhuang,lgchen}@video.ee.ntu.edu.tw

ABSTRACT

The paper exploits a vision-based intelligent vehicle cruise control system from the application level to the architecture level. Firstly, design considerations of the system are addressed in both computing power and accuracy aspects. Secondly, we present an efficient knowledge-based front-vehicle tracking algorithm. The algorithm yields below 5% error rate that outperforms the state-of-the-arts. Thirdly, a run-length-based algorithm optimization flow is introduced. Finally, specific hardware architecture is developed. It achieves 1280×960/80FPS and 4096×2160/10FPS requirements for multi-vehicle tracking tasks.

Index Terms— Intelligent cruise control, vehicle tracking, architecture design

1. INTRODUCTION

From the past to the present, vehicular technologies are developed to guarantee the driving safety and energy efficiency. For safety consideration, active systems, such as Collision Warning Systems (CWS) and Blind Spot Warning Systems (BSWS), attempt to provide a safer driving assistance. On the other hand, energy-saving issues have become inevitable and emerging trends [1, 2]. Around 30% of world energy consumption is due to automotive transportation. Among various automotive applications, cruise control satisfies both goals of safety and energy efficiency. The cruise control is originally designed to reduce driver's fatigue by keeping a constant driving speed. Adaptive cruise control (ACC) [3] integrates a radar sensor to detect the preceding target's position and velocity. It performs appropriate brakes and throttle actuation to maintain a safe distance. More interestingly, researchers found ACC can also greatly improve the energy efficiency around 20% [1, 2].

Current ACC mainly uses radars for sensing front targets. However, radars have small field of view (FOV) and high sensitivity to environmental interference. Nowadays, owing to the maturity of vision sensors, video analysis technology is widely used in vehicle recognition and understanding [4, 5], which makes it potential to create an intelligent functionality. The vision-based analysis techniques can be realized with algorithm development and implemented with integrated circuit design for handling high computation of vision data.

In this paper, we propose a vision-based intelligent cruise control (ICC) system based on a knowledge-based tracking algorithm. For ICC, a tracking framework performs position estimation and trajectory prediction on targets. It is also viewed as a pre-processing stage for understanding objects' behaviors. Generally, tracking algorithms can be classified into three categories: deterministic methods, stochastic methods and feature-based methods. Deterministic methods [6, 7], such as cam-shift and mean-shift, typically track by performing an iterative search for the similarity between the referenced template and the current target. Stochastic methods use the state space to model the underlying dynamics of a tracking procedure. The particle filter [8, 9] recursively constructs the posterior probability density function of the state space using Monte Carlo integration. Feature-based methods [10, 11] track by detecting the feature correspondence between the referenced patch and the targeted patch. SIFT has become a dominant way for feature-based vehicle tracking because of the tolerance on image rotation and scale variation [10]. However, these methods are incapable of explicitly distinguishing target objects from background. They may stick at tracking a generated false alarm and cause inaccurate positioning. Moreover, these methods cannot provide an accurate size estimation of targets, which is critical to ICC.

For advanced ICC in the future, high resolution video should be supported for long-range tracking. We make an investigation on system design challenge and exploit the related hardware architecture design for high throughput specification with high resolution video.

This paper is organized as follows. We discuss system design consideration for ICC in Sec. 2. Sec. 3 and Sec. 4 introduce the proposed algorithm and demonstrate experimental results. A hardware-oriented algorithm optimization is presented in Sec. 5. We describe the architecture design and implementation in Sec. 6 and Sec. 7. Finally, Sec. 8 concludes the work.

2. SYSTEM DESIGN CONSIDERATION

For an ICC system, safety and energy efficiency are essential purposes. Two aspects, relative vehicle distance and velocity, should be concerned while constructing an ICC system.

Firstly, we consider the relative distance. For a vision-based system, to avoid the difficulties of calibration and synchronization of a stereo camera, we set up a high resolution monocular camera instead. Thus, the relative distances Z between the host and a target vehicle can be estimated according to geometry perspective of a pinhole model described by Eq. 1.

$$Z = \frac{fW}{w} \tag{1}$$

where W is the vehicle width in meters and w is the vehicle width in pixels on image plane. f denotes the camera focal length in pixels. If a w estimation algorithm induces an n-pixel error, the induced distance error Z_{err} is give by Eq. 2 [12]. It can be noted that Z_{err} increases as the square distance Z^2 and algorithmic estimation error n increase.

$$Z_{err} = Z_n - Z = \frac{fW}{\frac{fW}{Z} + n} - Z = \frac{nZ^2}{fW + nZ} \approx \frac{nZ^2}{fW} \quad (2)$$

Secondly, we discuss the relative velocity. For a vision-based system, the relative velocity error v_{err} between the host vehicle and the target vehicle is represented as Eq. 3 [12]. a is denoted as the acceleration. The objective is to decide the optimal Δt that minimizes v_{err}. By differentiating Eq. 3, the optimal Δt and the minimal v_{err} are derived in Eq. 4 and Eq. 5, respectively. Since the distance Z cannot be absolutely converted to an integer pixel value via Eq. 1. The fractional pixel difference is denoted as image alignment error s_{err} that leads to estimation uncertainty of relative velocity.

$$v_{err} = \frac{Z^2 s_{err}}{fW\Delta t} + \frac{nZv}{fW} + \frac{1}{2}a\Delta t \quad (3)$$

$$\Delta t = \sqrt{\frac{2Z^2 s_{err}}{fWa}} \quad (4)$$

$$v_{err} = Z\sqrt{\frac{2as_{err}}{fW}} + \frac{nZv}{fW} \quad (5)$$

From Eq. 2 and Eq. 5, the induced relative distance and velocity errors have relation to video (sensor) frame rate, video resolution and target's distance. According to Eq. 4, the optimal frame rate $(1/\Delta t)$ decrease as Z increases. While designing a reliable ICC system, it is reasonable to choose a specification that minimizes Z_{err} and v_{err}. For instance, considering a typical driving condition: $Z = 30, s_{err} = 0.1, W = 1.7m, a = 10m/s^2$. The specification is around 10 FPS frame rate with $4k \times 2k$ resolution. Furthermore, an 1280×960/80FPS profile also fits a similar driving condition. In this paper, we develop an efficient tracking framework for vision-based intelligent cruise control with high resolution, high frame rate and high throughput requirement. Besides, multi-target tracking is also supported.

3. PROPOSED KNOWLEDGE-BASED VEHICLE TRACKING ALGORITHM

The proposed method utilizes features with strong prior-knowledge as tracking cues. Generally, a vehicle's rear view contains windshield, rear lights, license plate, shadow on the ground, etc. Intuitively, rear lights are the most obvious features with characteristics of color invariance, constant position, and high visibility. Fig. 1 illustrates a portion of vehicle images with rear views and the mean image of 200 sampled images. The mean image reveals the two red rear lights are dominant in the appearance. More importantly, the vehicle width is just the same as the interval between two rear lights. According to Eq. 1, the estimated distance Z can be obtained. On the other hand, the color and installation position of rear lights are regulated by legislation. Above all, we proposed a knowledge-based tracking framework based on rear light detection.

The overall algorithm flow is shown in Fig. 2. Firstly, the rear lights are extracted using L*a*b* color segmentation. Secondly, a symmetric verification procedure is executed for precise width estimation. Lastly, a temporal filtering is adopted to prevent estimation glitch and maintain tracking continuity.

3.1. Rear Light Detection

Color segmentation separates the given region of interest (ROI) into rear lights and other parts. The RGB values are firstly transformed into L*a*b* color space to have a strong color discrimination. Since a* and b* stand for color values on the red-green and the yellow-blue axes respectively, they are usually employed to identify the "red" color. Eq. 6 states the segmentation criterion with adjustable thresholds α and β associated with illumination change. A binarized segmentation map $I(x, y)$ is obtained through pixel-wise comparison.

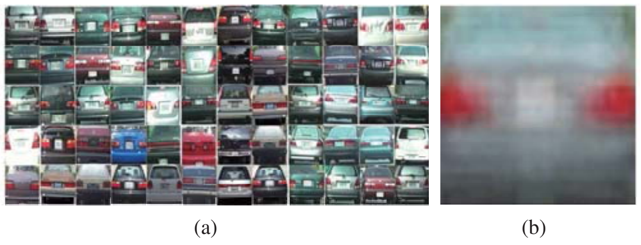


Fig. 1. Examples of dominant components. (a) Vehicle's rear views. (b) The mean image.

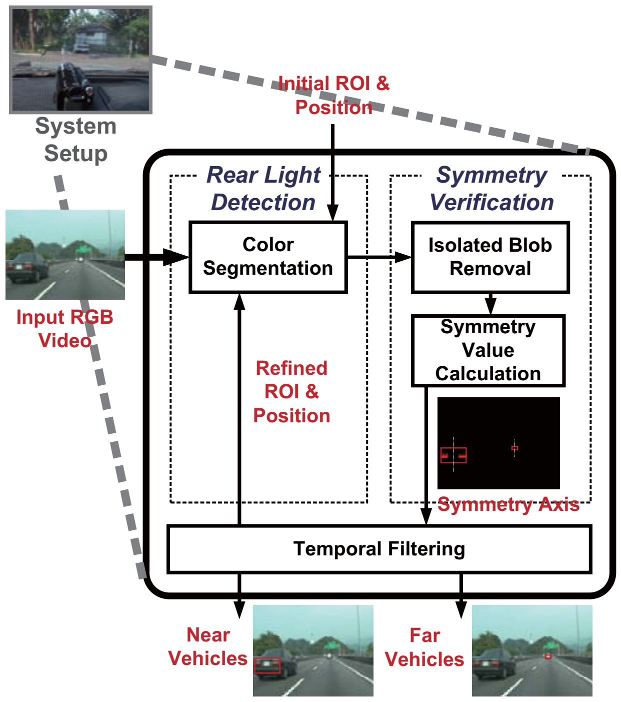


Fig. 2. System block diagram.

$$I(x, y) = \begin{cases} 1, & a^*(x, y) > \alpha \wedge a^*(x, y) - b^*(x, y) > \beta \\ 0, & otherwise \end{cases} \quad (6)$$

3.2. Symmetry Verification

The knowledge-based method not only records vehicles' position but also estimates their precise width. Knowing that vehicles' rears are in general laterally symmetrical, symmetry verification explicitly locates vertical boundaries of vehicles' rears via shrinking operators. The operators morphologically remove isolated color blobs and extract intact red regions that represents rear lights for measuring the width of a vehicle.

In a color-segmented ROI, B_L and B_R are denoted left and right boundaries of red regions. B_U and B_D are denoted upper and lower boundaries. Left and right boundaries are shrunk to form the most symmetrical appearance inside the ROI. The left and right shrinking functions are defined in Eq. 7 and Eq. 8 respectively. s means the edge shrinking value in pixels. The objective is to find s_l and s_r such that both $SymL(s)$ and $SymR(s)$ are minimized. s_l and s_r decide new boundaries and retrieve a symmetrical axis. The target width is afterwards defined by the new generated boundaries.

$$SymL(s) = \sum_{j=B_D}^{B_U} \sum_{i=1}^{(N-s)/2} I(B_L + i + s, j) \oplus I(B_R - i, j), \tag{7}$$
$$s = 0, 1, 2, ..., N/4 \wedge N = B_R - B_L$$

$$SymR(s) = \sum_{j=B_D}^{B_U} \sum_{i=1}^{(N-s)/2} I(B_L + i, j) \oplus I(B_R - i - s, j), \tag{8}$$
$$s = 0, 1, 2, ..., N/4 \wedge N = B_R - B_L$$

A FIR temporal filter is used to suppress estimated errors. This reduces the effect of factor n described in Eq. 2 and Eq. 5. Because initial region for segmentation can be generated from previous tracked results, the tracking mechanism maintains temporal coherence.

4. EVALUATION AND EXPERIMENTAL RESULTS

To evaluate whether a tracking algorithm is suitable for cruise applications, we introduce two objective criteria, width error rate (WER) and centroid departure rate (CDR). Real test sequences were captured from a CMOS high-resolution front-mounted camera on a vehicle. The ground truth, including targets' width and position, is generated by manually annotating in video frame by frame.

WER is defined as the ratio of the sum of estimated width errors over the sum of true width of a target during the entire sequence (Eq. 9). i is denoted frame index and F stands for the total number of frames. Smaller WER suggests more precise width and smaller Z_{err} can be obtained during tracking.

$$WER = \frac{\sum_{i=1}^{F} |Width_{esti}(i) - Width_{true}(i)|}{\sum_{i=1}^{F} Width_{true}(i)} \tag{9}$$

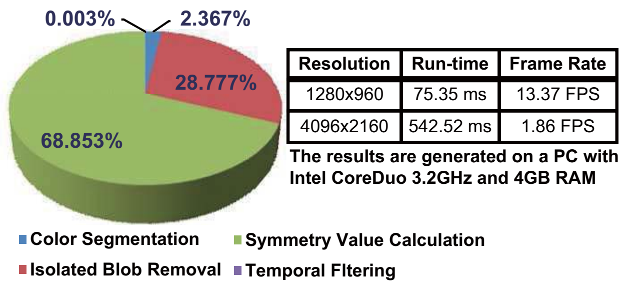
CDR is defined as the ratio of the sum of estimated centroid drifting errors over the sum of true half width of a target during the entire sequence (Eq. 10). CDR is a measurement for the closeness of the tracked object to the true target. Lower CDR implies a stable localization of objects.

$$CDR = \frac{\sum_{i=1}^{F} |Centroid_{esti}(i) - Centroid_{true}(i)|}{\sum_{i=1}^{F} Width_{true}(i)/2} \tag{10}$$

Table 1 summarizes the accuracy evaluation using a single-target tracking task. Cam-shift misses trackers within a few frames owing to the fast moving background and causes over 100% WER and CDR. Hence, it is excluded from Table 1. As observed, the proposed method outperforms others in both WER and CDR. For far vehicle tracking scenarios, three algorithms all have less than 8% WER. However, the particle filter and the SIFT-based tracking have 22.42% and 18.56% CDR respectively for far tracking scenarios. As for near tracking and overtaken conditions, the WER increases drastically for particle filter and SIFT-based tracking. In addition, these

Table 1. Accuracy comparison to other tracking algorithms.

Algorithm	Particle Filter		SIFT-based		Proposed	
	WER	CDR	WER	CDR	WER	CDR
Sequence1 (overtaken)	14.04%	25.26%	18.02%	28.32%	0.92%	0.64%
Sequence2 (near)	13.38%	4.36%	10.44%	7.24%	2.59%	1.86%
Sequence3 (far)	7.42%	10.68%	4.02%	8.82%	3.44%	3.20%
Sequence4 (far)	4.30%	22.42%	4.56%	18.56%	3.02%	2.90%



Resolution	Run-time	Frame Rate
1280x960	75.35 ms	13.37 FPS
4096x2160	542.52 ms	1.86 FPS

The results are generated on a PC with Intel CoreDuo 3.2GHz and 4GB RAM

Fig. 4. Software average run-time profiling and distribution.

two algorithms have opposite accuracy between WER and CDR. The inconsistency implies unreliability for the ICC application. On the other hand, the proposed method has superior performance with only 2.81% WER and 2.38% CDR in average. Consequently, the proposed knowledge-based method is preferred for minimization of Z_{err} and resultant v_{err}. Fig. 3 illustrates the tracking results on typical road and highway at distinct time.

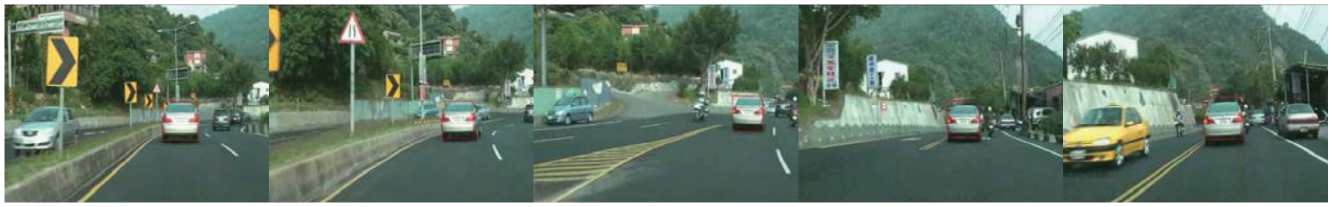
5. HARDWARE-ORIENTED ALGORITHM OPTIMIZATION

5.1. Software Run-time Analysis

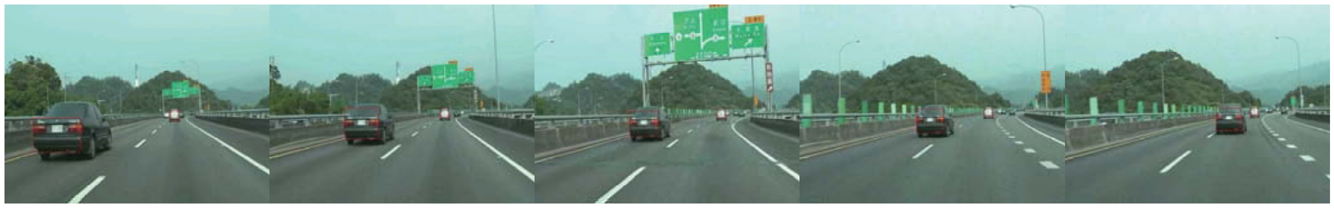
As discussed in Sec. 2, our system aims at 4096×2160/10FPS and 1280×960/80FPS profiles. Fig. 4 shows the average software run-time and frame rate analysis using single-target tracking task. Run-time distribution for four algorithmic components is also illustrated. With software implementation, only 10 to 15 FPS for 1280×960 resolution and less than 3 FPS for 4096×2160 resolution are achieved. Symmetry verification includes isolated blob removal and symmetry value calculation occupies over 90% computing power. For multi-target tracking, the run-time is approximately multiplied by the number of targets. As analyzed, it cannot meet the target specification even for single-target tracking. Consequently, a hardware-oriented algorithm optimization scheme and the corresponding hardware development are proposed to gain higher computing capability.

5.2. Run-length Domain Processing

We establish a run-length-based algorithm optimization scheme. Morphological operations and the symmetry verification are integrated with run-length encoding. The run-length encoding benefits data buffer size. Original data is stored as binaries with data quantity $O(N)$. Run-length encoding compresses the binaries to different run-length series, which reduces data quantity to $O(log_2 N)$. For a 1280×960 image, the maximum target size is about 300×200 pixels. The original buffer size is around 60Kbits. With run-length encoding, eight 8-bits lengths are enough to represent a row in the



(a)



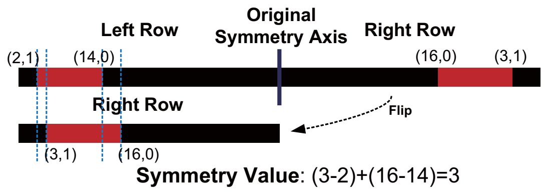
(b)

Fig. 3. Tracking results (red bounding boxes) at different time points. (a) Single-target tracking for a near right-turning vehicle. (b) Multi-target tracking for both near and far vehicles at the same time.

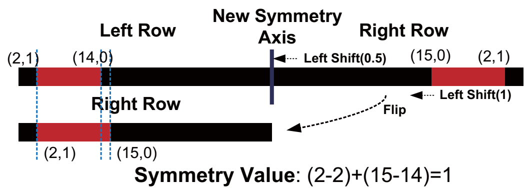
maximum target region. Hence, the buffer size can be reduce to 12.8Kbits, which results in a 78.6% reduction. As for 4096×2160 resolution, around 92.5% memory reduction can be achieved.

The current encoded run-length set is saved to memory if it exceeds a length threshold. Otherwise, it is combined with the previous run-length set. The isolated blob removal is operated via such mechanism. The length threshold is adjusted dynamically depending on the size of the tracked ROI. The larger the ROI is, the higher the threshold is.

The symmetry value is defined as the sum of difference between left-row's and right-row's length values. Therefore, the symmetry difference can be simply calculated by summing the subtractions of the first and the second, the third and the forth, ..., and the $N-1_{th}$ and the N_{th} smallest length numbers in a row. Fig. 5 shows an example for the process. For an original symmetry axis, the codes of its left-row and right-row sides are $\{(2,1), (14,0)\}$ and $\{(16,0), (3,1)\}$, respectively. Before substraction, a flip operation is applied to the right row. Therefore, the symmetry value is thus (3-2)+(16-14) = 3 pixels. Now if it applies an 1-pixel left-shift operator and the same flip operation for the right row, the symmetry value becomes (2-2)+(15-14) = 1 pixel. The 1-pixel left-shift for right row is the same as shifting the original symmetry axis to the left for 0.5 pixels. Finally, the symmetry value of every row in the ROI will be accumulated to generate a final score. The symmetry axis with the minimum score is determined as the final symmetry axis. Moreover, the shrinking operators in Eq. 7 and Eq. 8 can be treated as left-shift or right-shift operators mentioned here.



(a)



(b)

Fig. 5. Examples for symmetry value calculation in run-length domain. (a) Original. (b) After a left-shift operation.

6. ARCHITECTURE DESIGN

We develop the Knowledge-based Vehicle Tracking Processor (KVTP) for the entire framework (Fig. 6). The architecture contains a system controller, 8-parallel single-port on-chip memory banks, a temporal filtering module, and three processing engines including Color Segmentation Engine (CSE), Run-length Encoding Engine (RLEE), Symmetry Engine (SE). The frame data is stored in an off-chip DRAM memory and is accessed with a vertical raster scan. KVTP receives initial ROI information as a start signal and accesses RGB data from the off-chip DRAM. CSE converts pixel value from RGB to L*a*b* and binarizes the converted value. RLEE encodes the 0-1 sequences to run-length segments and dispatchs them to the parallel banks. SE loads run-length segments from the memory banks and calculates the symmetry value for each ROI. The final refined ROI information is generated and adjusted via a 3-tap FIR temporal filter for both width and position parameters. The output are fed back as the initial ROI information for the next frame.

6.1. Input Data Flow

In a front-faced captured image, target objects arrange horizontally (Fig. 7). For a line-based horizontal raster scan, it should buffer unprocessed partial object data if there are multiple targets (Fig. 7(a)). This increases on-chip memory (Buffer1, Buffer2 and Buffer3). Therefore, a region-based vertical raster scan is preferred (Fig. 7(b)). Only one buffer for all objects is adequate (Buffer1). The maximum memory equals to the size of the largest object. Once a current target is processed, the memory is released for the next one. The maximum target size is 300×200 pixels for 1280×960 resolution, the

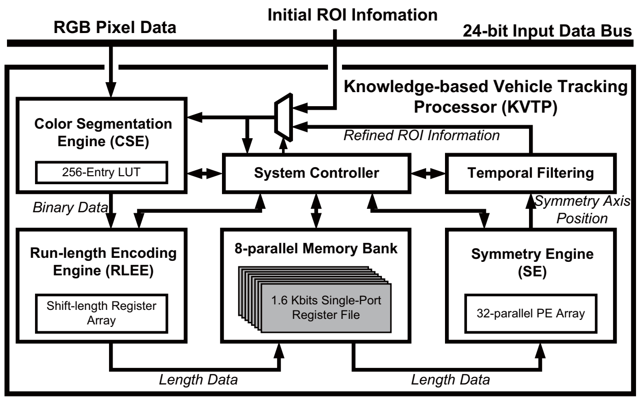


Fig. 6. Proposed knowledge-based vehicle tracking processor.

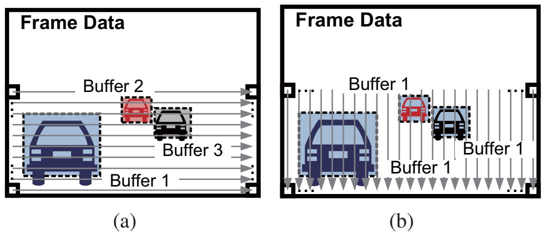


Fig. 7. Input data flow. (a) Horizontal scan. (b) Vertical scan.

vertical raster scan can save up to 77% on-chip memory (reduced from 256Kbits to 60Kbits). As for input bandwidth, 24-bit data bus supports 2.4Gbps bandwidth with 100MHz clock frequency. This is sufficient for a 4096×2160/10FPS video.

6.2. Color Segmentation Engine

Since RGB to L*a*b* conversion is nonlinear, which involves division and cubic root calculation. Instead of directly implementing dividers and cubic rooters, look-up table units (LUT) are adopted. CSE has three 256-entry LUT units for 8-bit input data.

6.3. Run-length Encoding Engine

The RLEE encodes data along each row horizontally. However, CSE sends binaries along columns because of the vertical input raster scan. The mismatch introduces latency and increases memory buffer size. To minimize latency, data access and encoding procedure should occur in the same time. Thus, RLEE selects a row in ROI for running length each cycle. The direct implementation adopts an enormous multiplexer for row selection and is quite area-consuming. We propose a shift-length register 1D array instead. As shown in Fig. 8, the lengths of all rows are shifted cycle by cycle in a circular FIFO structure. The length value of a row where the current input binary locates is shifted to the head of the 1D register array. Only the data at the head register is encoded. By using shift-length register array, 63% area reduction can be achieved (reduced from 224.8K to 83.4K). A 1.6Kbit local buffer is utilized for storing previous encoded length segments for isolated blob removal. If the current encoded length is smaller than a threshold, it is united with previous encoded segments. In brief, through run-length encoding, it further saves 78.6% (reduced from 60Kbits to 12.8Kbits)

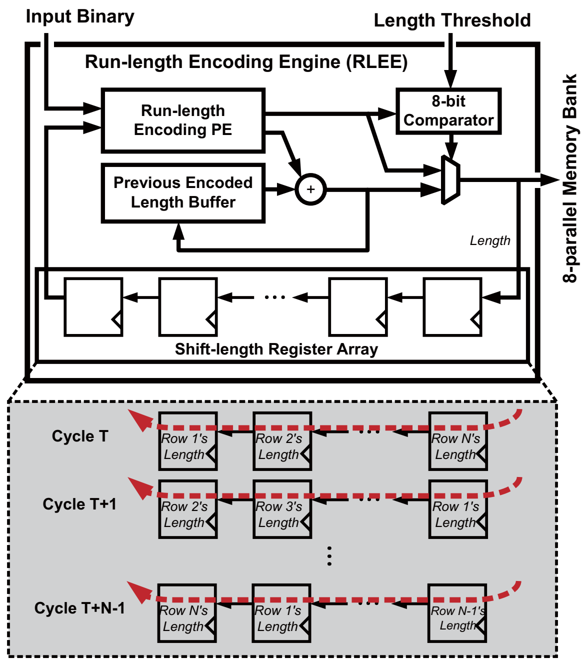


Fig. 8. Run-length encoding engine and the length shifting schedule in a shift-length register array for each clock cycle.

and 92.5% (reduced from 432Kbits to 32.4Kbits) on-chip memory for 1280×960 and 4096×2160 resolution, respectively.

6.4. Symmetry Engine

The SE consists of a 32-parallel processing element (PE) array that increases throughput for supporting multi-target tracking (Fig. 9). Symmetry value is the sum of subtractions of length. The encoded length data are loaded from the on-chip memory and then simultaneously transferred to all PEs. A 8 to 32 data dispatcher solves bandwidth mismatch between memory banks and the SE. Each PE calculates the symmetry value with respect to different symmetry axes. A comparator tree firstly sorts the loaded length according to their position in a row. Two neighboring length values are subtracted afterwards. The subtraction results are summed up and accumulated for all rows. The accumulated symmetry values of all PEs are then compared and the smallest one decides the refined symmetry axis.

6.5. N-parallel Memory Bank

The on-chip register file is used to store length codes. To solve throughput and bandwidth mismatch between RLEE and SE, N-parallel memory banking technique is utilized. In our experiments, 8-parallel is sufficient to reduce the mismatch. Besides, 8 lengths are enough to describe a row in the ROI. Therefore, the memory is separated into 8 banks. The N_{th} encoded length in a row is saved to the N_{th} bank. While encoding, only one bank is enabled to reduce power consumption. While symmetry value calculation, the words of all banks that represent a row are loaded to SE at a time for increasing throughput. After a target is processed, the 8-parallel memory banks are reset.

Table 2. Synthesis results and the simulated hardware profile

Frequency (MHz)	200	167	125	100
Gate count (K)	341.9	252.5	230.3	219.4
On-Chip Memory ($Kbits$)	12.8	12.8	12.8	12.8
Throughput (FPS)	1280×960@162 FPS	1280×960@135 FPS	1280×960@101 FPS	1280×960@81 FPS
	4096×2160@22 FPS	4096×2160@18 FPS	4096×2160@14 FPS	4096×2160@11 FPS

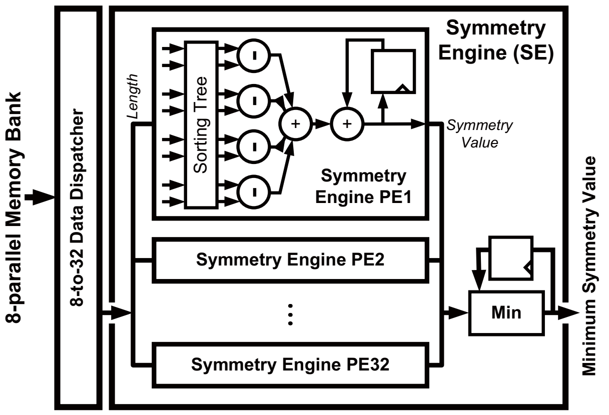
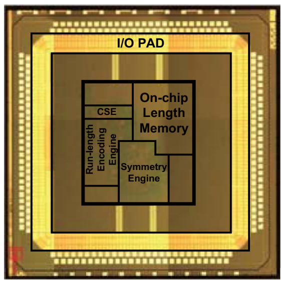


Fig. 9. Symmetry engine and the processing element array.



Technology	90nm CMOS
Chip Size	4.84 mm^2
Core Size	1.23 mm^2
Core V$_{dd}$	2.5 V
On-chip Memory	12.8 Kbits
Operating Frequency	100 MHz
Supported Frame Rate	81.4 FPS@1280x960 11.3 FPS@ 4096x2160
Maximum Target #	5

Fig. 10. The die micrograph and performance summarization.

7. CHIP IMPLEMENTATION RESULTS

The knowledge-based vehicle tracking processor is implemented in an UMC 90nm Low-K SP CMOS process. Table 2 lists logic synthesis results with different clock frequency for multi-target tracking. The chip size is 2.2×2.2mm^2 including IO pads and bonding pads. The throughput is 5× to 10× compared to software implementation. Chip power dissipation has relations with number of targets. It consumes about 645mW for maximum 5 tracking tasks and 23mW for single-target tracking. A total 12.8Kbits 8-parallel memory buffer supports an object with maximum height 200 pixels. For 4096×2160 specification, two memory banks are combined to support an object with maximum height 400 pixels. The chip operates at 100MHz frequency with 2.5V core voltage. The die micrograph and detailed chip performance are shown in Fig. 10.

8. CONCLUSION

We establish an efficient and reliable vehicle tracking framework. The contribution is three-fold. (1) An intelligent vision-based cruise control system is constructed. We exploit relationship between application requirements and the processing capability. (2) An application-specific knowledge-based tracking algorithm is proposed considering objects' appearance. (3) A circuit is designed to support high specification. We also bring up a run-length domain algorithm optimization. In the future, the system can be integrated to develop an intelligent vehicle or other automotive applications.

9. REFERENCES

[1] O. Servin, K. Boriboonsomsin, and M. Barth, "An energy and emissions impact evaluation of intelligent speed adaptation," in *Proc. IEEE Intelligent Transportation Systems*, Sept. 2006, pp. 1257–1262.

[2] S. Tsugawa, "An overview on energy conservation in automobile traffic and transportation with ITS," in *Proc. IEEE Vehicle Electronics*, Sept. 2001, pp. 137–142.

[3] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 4, pp. 143–153, Sept. 2003.

[4] S. Sivaraman and M.M. Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, pp. 267 –276, June 2010.

[5] Y.-M. Tsai, K.-Y. Huang, C.-C.Tsai, and L.-G. Chen, "Learning-based vehicle detection using up-scaling schemes and predictive frame pipeline structures," in *Proc. IEEE ICPR*, Aug. 2010, pp. 3101–3104.

[6] Y. Changjiang, R. Duraiswami, and L. Davis, "Efficient mean-shift tracking via a new similarity measure," in *Proc. IEEE CVPR*, June 2005, vol. 1, pp. 176–183.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *PAMI, IEEE Transactions on*, vol. 25, pp. 564–577, May 2003.

[8] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *European Conference on Computer Vision*, 2002, pp. 661–675.

[9] Y. Changjiang, R. Duraiswami, and L. Davis, "Fast multiple object tracking via a hierarchical particle filter," in *Proc. IEEE ICCV*, Oct. 2005, vol. 1, pp. 212–219.

[10] Qiu Tu, Yiping Xu, and Manli Zhou, "Robust vehicle tracking based on scale invariant feature transform," in *Proc. IEEE International Conference on Information and Automation*, June 2008, pp. 86–90.

[11] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Jan. 2004.

[12] G.P. Stein, O. Mano, and A. Shashua, "Vision-based ACC with a single camera: bounds on range and range rate accuracy," in *Proc. IEEE Intelligent Vehicles Symposium*, June 2003, pp. 120–125.