Communication

Not peer-reviewed version

A Self-Organizing Multi-Layer Agent Computing System for Behavioral Clustering Recognition

Xingyu Qian , Aximu Yuemaier , Wen-Chi Yang , Xiaogang Chen * , Longfei Liang , Shunfen Li , Weibang Dai , Zhitang Song

Posted Date: 10 May 2023

doi: 10.20944/preprints202305.0746.v1

Keywords: field programmable gate array (FPGA); hardware implementation; real-time system; action clustering

Article

A Self-organizing Multi-Layer Agent Computing System for Behavioral Clustering Recognition

Xingyu Qian [1], Aximu Yuemaier [2], Wen-Chi Yang [3], Xiaogang Chen [1,*], Longfei liang [3], Shunfen Li [1], Weibang Dai [1], and Zhitang Song [1]

[1] Chinese Acad Sci, Shanghai Institute of Microsystem and Information Technology, Shanghai 200050, China
[2] Shanghaitech University, Shanghai 201210, China
[3] NeuHelium Co., Ltd., Shanghai 200050, China
* Author to whom correspondence should be addressed.

Abstract: Video behavior recognition often needs to focus on object motion processes. In this work, a self-organizing computational system oriented to behavioral clustering recognition is proposed, which achieves the extraction of motion change patterns by binary encoding and completes motion pattern summarization using a similarity comparison algorithm. And in the face of unknown behavioral video data, a self-organizing structure with layer-by-layer accuracy progression is used to achieve motion law summarization by using a multi-layer agent design approach. Finally, the real-time feasibility is verified in the prototype system using real scenes to provide a new feasible solution for unsupervised behavior recognition and space-time scenes.

Keywords: field programmable gate array (FPGA); hardware implementation; real-time system; action clustering

1. Introduction

Natural video has the power to be able to serve as unsupervised learning for static as well as dynamic vision tasks. The variation of video frame images contains a large amount of information about the scene and behavior. Video data consists of frames, and each frame is a collection of dotted data consisting of RGB pixels. Whether for the recognition of target behaviors or the detection of abnormal behaviors, most of the usual practices first classify the moving targets, such as the mainstream CNN+RNN [1], two-stream [2] or 3D CNN [3], and then combined with specific feature information for action behavior recognition.

For video processing, the extraction of key points in the original video information requires an intensive computational process, which will undoubtedly bring about many computational operations. And the complex and huge computation processes are bound to affect the processing speed. In an end-to-end real-world application, the too-slow single-frame processing speed cannot keep up with the video frame input rate, and it is difficult to meet the real-time processing. In fact, video information is always changing continuously, and this change situation is already strongly regular. If the act of classifying objects or features in motion is discarded in favor of focusing directly on the changing relationships between video frames, it is possible to achieve clustering and recognition of specific motion behaviors at the theoretical level. For example, in the case of a black-and-white silhouette performance, even without knowing the specific identity of the performing character or the background story, etc., it is still possible to understand the meaning of the performance action by watching it. This is obtained by focusing on the behavior-action change situation rather than classifying the target first.

In addition, for unlabeled video data, it is not possible to use a priori knowledge to know the detail density and objection distribution of the video to be processed, due to the computational processing of the original video. This situation needs to be done either by selecting the dataset or by completing the filtering manually. The algorithmic model needs to be designed in a targeted way to satisfy the extraction of feature values from unknown videos. For example, the algorithm model for absolute coordinate calculation applied to traffic information of a certain traffic road or an intersection for

 doi:10.20944/preprints202305.0746.v1

targeted processing is difficult to be directly applied to another intersection or even another viewpoint without modification, which will lead to difficulty in recognition under the original calculation model.

In this work, for the traffic road scenario, we propose a self-organizing multi-layer agent computing system (SMLACS) implemented using conventional algorithms. The computational effort is reduced by using a binary coding approach to filter the video information density and retain only motion information. Then the clustering and identification of motion processes are accomplished using a similarity comparison algorithm. Finally, a self-organizing multilayer structure is designed and used to verify the hardware feasibility using FPGAs combined with cameras through real scene data [4]. This work aims to provide a new and feasible design idea for motion analysis or even spatio-temporal scene applications, and thus achieves basic clustering and recognition capabilities in terms of functional effects only. However, the SMLACS focus on motion information scheme is equally applicable for otherwise blurred video data as opposed to the requirement of frame fineness for the frame calculation approach.

2. Relate work

Whether by traditional iDT [5] or artificial neural networks, optical flow methods play an important role in scene understanding and the pursuit of higher-level cognitive tasks such as motor behavior analysis. Most hardware implementations have focused on the H&S algorithm [6] and the L&K algorithm [7]. The advent of the H&S algorithm helps to retrieve the apparent motion of pixels from the video and image sequences. H&S algorithms require high speed to compute dense and accurate flow vectors with deterministic latency and low power consumption, and thus developed many applications. However, the sequential nature of the iterative solvers in the H&S algorithm leads to long computational processing times for sparse systems of equations until the desired accuracy is obtained. The L&K method in its essence solves the optical flow problem by least squares in an iterative-free way. The algorithm can be applied to sparse scenes because it relies only on local information derived from some small window around a point of interest, but the disadvantage is that the algorithm will not be able to find the pixel point if its motion is too large and moves outside the local window.

Optical flow methods require high-density computational requirements and are usually implemented based on high-performance, highly parallel processing architectures. For example, Lazcano et al. [8] processed 320 $\times$ 240 pixels video frames at 4 FPS by modifying the classical HSOF method on a GPU GeForce NVIDIA-GTX-980-Ti platform. Gokhan et al. [9] performed a targeted design based on HSOF on Cyclone$-$2 FPGA chip from Altera on a DE2-70 board and processed 256 $\times$ 257 pixels video frames at 256 FPS. Seong et al. [10] proposed an improved strategy for storing input images after Gaussian filtering operation using the L&K algorithm, and achieved processing 800 $\times$ 600 pixels HD video images at 30 FPS, reducing frame memory access by 61Barranco et al. [11] proposed the design of the L&K algorithm with the multi-scale extension, based on Xilinx Virtex4 XC4vfx100 chip to achieve 270 FPS for a 640 $\times$ 480 pixels, and 30 FPS with the multi-scale extension.

Compared to the mainstream algorithmic models, our proposed SMLACS focuses only on the moving process in binary encoding, and the algorithmic function is accomplished by using similarity comparison and part of logical judgments. SMLACS can achieve the clustering function for motion behavior with a small computational consumption, even for blurred video data.

3. Method

3.1. Binary Encoding(BE) and Discrete Temporal Sequence

In contrast to the optical flow method, where exact values are calculated to preserve the object feature information, we propose to care only about whether changes occur during the sampling phase of the video information as a basis for encoding and ignore the calculation of specific values. Therefore,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2023 doi:10.20944/preprints202305.0746.v1

based on the background modeling method, we choose to encode binary values (0 and 1) by comparing the difference values. The process of binary encoding using class optical flow is shown in Figure 1, where the example and subsequent legends are derived from the webcam capture video frame [4]. The video frame image divides its area into a 4 × 4 data grid, using the current video frame and the background frame to make the difference, and different grids count the difference information of the corresponding area. Two parameters are set as the evaluation criteria: the difference size threshold and the number threshold for exceeding the standard difference value, and when a difference data exceeds the difference criterion, its statistic is increased by 1. The other cases indicate no change (white areas in the figure).

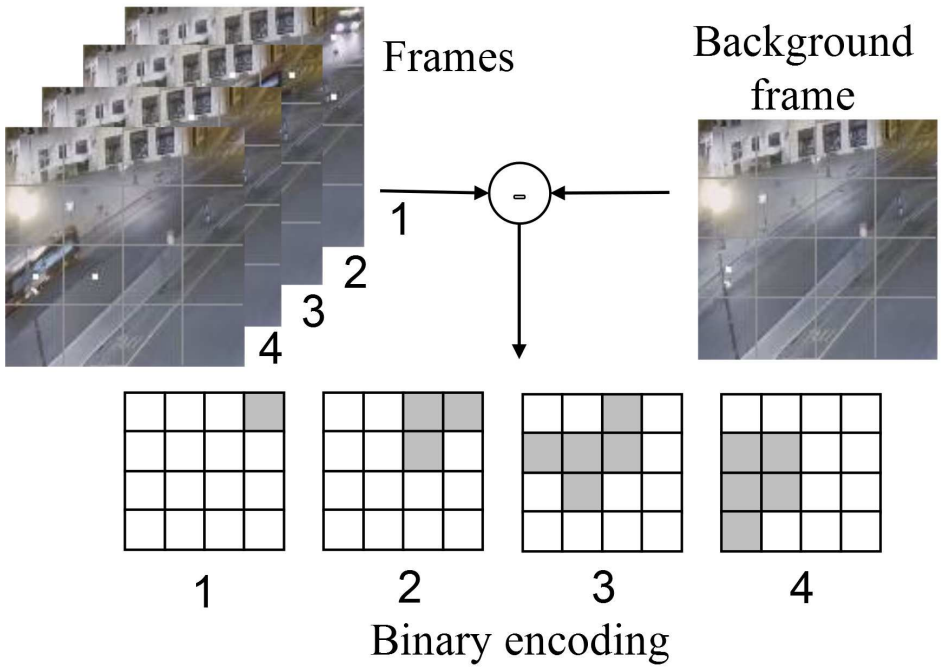


Figure 1. Class optical flow binary encoding.

In this way, the four video frames of the example in Figure 1 are encoded into the corresponding four binary arrays. The different binary arrays under this time slice can capture the motion of an object over time, while there is a logical sequence of changes in the binary data of each grid on the time scale. If there is motion behavior then the motion pattern can be understood by the change process of the video frame through the binary encoding method. The overall pixel profile of a video frame is not always stable, for example, the overall pixel size may change over time due to ambient lighting. If data is sampled and encoded for a short period of time, a fixed selection of background frames can always be maintained, while for complex environmental conditions or long runs, background frames need to be updated at regular intervals or when the background changes to avoid data failure due to large gaps with the current frame image. Binary encoding enables the conversion of high-density image information into low-density binary array parameters that capture the perception of change. The accuracy of the information data is determined by the number of divided grids, and a denser number of divided grids is set if high-accuracy coding is required.

The binary encoding method enables the transformation of the perceived changes in each grid into a simple encoded form of 0 and 1. Each individual grid can be combined on a time scale into a time series of varying lengths that can represent the duration of change. Only because of the binary encoding, each grid exhibits a continuous process of change that can only be represented as a continuous 1. Figure 2 shows a 4 × 4 uniform grid division and a partial video frame an encoded slice of a car driving after binary encoding. The four grids on the upper right of the image are named

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2023 doi:10.20944/preprints202305.0746.v1

A, B, C, and D. The encoded data of different frames of these four grids are collected to form discrete temporal sequences, and four sets of binary discrete sequences with successive changes are obtained. From the time scale, it is easy to see that there is an obvious correlation between the data changes of the four grids, which is closely related to the vehicle movement process, so it will show the sequential changes shown in the figure.

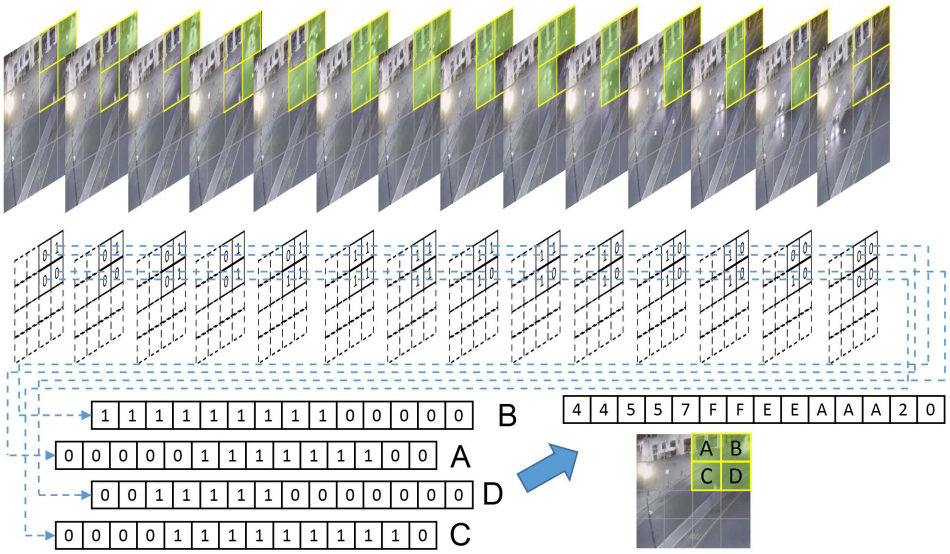


Figure 2. Video frame slicing and discrete temporal sequence coding.

If only binary encoding is used for the combination of discrete temporal sequence, not only more agents are required; but also the binary can show very limited variation and can only indicate the presence or absence of object motion. And if we only look at the data from a single grid, all the regions where changes exist will only turn out to be no longer stationary when the picture starts to appear at a certain time, and then return to a stationary state again after a period, and the overall object motion process cannot be summarized by a series of individual grid data. Therefore, by designing the use of four adjacent grids of data exactly can use hexadecimal data to superimpose a combination of its encoded values for each frame, which not only improves the data richness that is difficult to express in a binary sequence but also reduces the information density by a factor of four. For example, the fourteen frames of the grid in Figure 2 are combined into "44557FFEEAAA20", which can be reduced to "457FEA20" again if we ignore the repeated encoding segments and keep only the changed parts. ", greatly reduces the information density without losing the effective data.

3.2. Similarity Comparison

Binary coding represents the motion process of an object as a discrete temporal sequence. The discrete temporal sequence of this type is stored as a model, and the clustering and identification of the motion behavior of the region can be achieved by comparing the new motion sequence with the similarity of the model. We design to use the generic longest common subsequence (LCS) algorithm as the similarity calculation algorithm to obtain the optimal match. The specific calculation process of similarity comparison using the model and the input sequence is performed using the longest common subsequence algorithm scheme. The computation process is a bit-by-bit comparison to find out whether the two sequences are identical and what each corresponds to, to evaluate whether the two sequences are identical. The structure of the LCS algorithm is shown in Figure 3, where the length of the sequence is i. In fact, the last term is not necessarily the end marker 0. Here, we just take the complete sequence model as an example, the length of the model is j, and the result of the bitwise comparison is counted by iterating through the loop twice. The data between different comparison

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2023 doi:10.20944/preprints202305.0746.v1

positions become an inheritance relationship, and finally, the dynamic planning table of $(i+1) \star (j+1)$ two-dimensional array data is obtained.

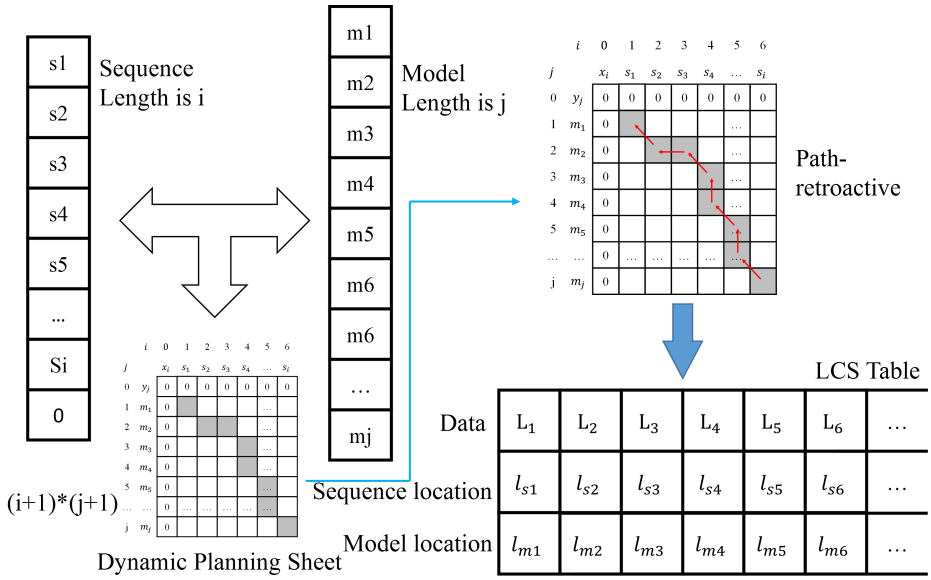


Figure 3. LCS algorithm operation process.

For example, the bit a of the input sequence of length i and the bit b of the model of length j are compared. If they do not agree, the current (a,b) data takes the maximum of the (a-1,b) and (a,b-1) values; and when their data are the same, the result of the (a,b) position is the value of its diagonal position (a-1,b-1) + 1. The dynamic planning table of the comparison process is obtained statistically in this way. Subsequently, backtracking from the maximum value is used, i.e., backtracking along the path from decreasing values. If the values of (a-1,b) and (a,b-1) adjacent to (a,b) are both smaller than 1, the next path position is (a-1,b-1), while if one of (a-1,b) and (a,b-1) has the same value as (a,b), the next path position is the position with the same value. Accordingly, the correspondence table of the LCS in the figure is obtained, where the first row represents the specific characters present in both sequences in the LCS, and the second and third rows correspond to the specific position of each column of the first row corresponding to its data in the input sequence and in the model. For example, if l_{s2} is 3 and l_{m2} is 4, it means that L_2 is the 3rd in the input sequence and the 4th in the model.

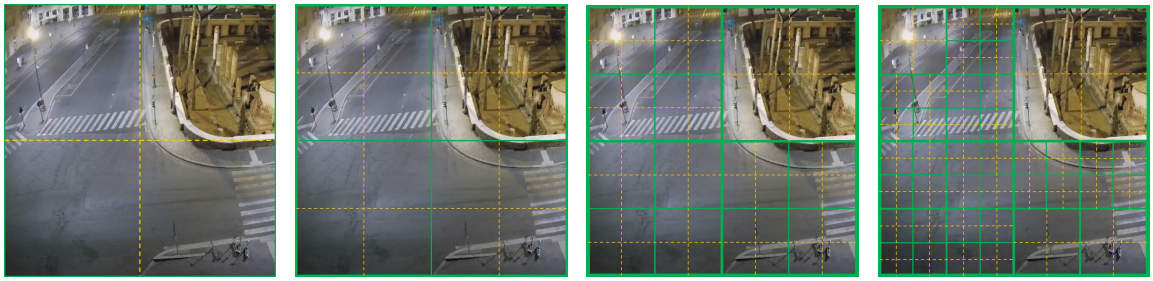
3.3. Multi-layer Self-organizing Structure

BE of the raw video data enables the perception of whether the relevant grid has changed. If the range of the grids is large, which means that the number of grids is small, less information can be obtained by binary coding. For example, if the whole image is not divided, only one grid is stored, so when the overall representation is 1, it only indicates that there is motion in the current frame, but no more detailed information is available. Assuming a pixel-by-pixel division, the detailed shape of the moving object can be obtained by binary coding alone. However, this approach is computationally intensive and loses the advantage of binary coding, so the amount of information available is determined by the size of the divided grid.

However, a video cannot have all its frames changed, and there will always be some areas where no moving objects always pass by. And if the set grid range is small, there may be quite a lot of grids that are always producing invalid data. We propose to use a progressive subdivision layer structure to capture the motion process of the video. The idea of the scheme is to provide more computational resources to the existence of more dense change regions for more detailed data processing. The specific agent system building structure of the scheme is expressed as a layer-by-layer input of agent resources from coarse precision regular summaries to refined regions. According to this dynamic allocation idea,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2023 doi:10.20944/preprints202305.0746.v1

we design a multi-layer structure(MLS) for coarse-to-fine precision progression, whose structure is shown in Figure 4. The green box in the figure indicates the region monitored by an agent, and the yellow divided dashed line in the green box indicates the corresponding four binary encoded regions, which receive and process the hexadecimal encoded temporal sequence of the four binary data in the green box by the computing agent.



(a) (b) (c) (d)

Figure 4. The case of dynamic segmentation layer by layer.

When an agent keeps a model with multiple temporal sequences, it indicates that there are multiple variations in the region that the agent is responsible for monitoring. This agent then seeks four agents in an idle state to be responsible for monitoring each of the four dashed regions in the figure with a co-working model like this agent. For example, in the initial operation of the system, the design is shown in Figure 4a, which divides the whole screen into only four grids and uses only one agent as the first layer, which can be responsible for the whole screen for motion monitoring. The data received by the agent of the first layer is very coarse only to get whether there is a regular change in the four areas of the division but not to get more detailed information. Subsequently, the agent of the first layer looks for four agents as the second layer to refine the four binary encoding regions received by the agents of the first layer and divide the four smaller regions for binary encoding respectively. Similarly, the third layer is subdivided within the size of the agent management area of the second layer, and the fourth layer is subdivided within the agent management area of the third layer. The agent's working state switching as well as data reception are achieved by filtering and filtering the received data for point-to-point transmission. At the same time, the execution process of the agent does not participate in any data exchange behavior except for receiving the input data of each frame at the beginning, which avoids data dependency and improves the processing speed, which can be easily realized to increase the processing speed to the video frame rate.

4. Prototype System

To verify the feasibility of SMLACS, we designed a prototype verification system. As shown in Figure 5, a rich storage resources platform UltraScale+ EG board, MIPI camera, and an LCD screen. The UltraScale+EG board is based on Xilinx Zynq−UltraScale all programmable SOC. It has four ARM Cortex A53 processors (PS) and 600 K series−7 programmable logic (PL) cells. The PS and PL are connected internally by the AXI bus. The camera's sampling rate is up to 60 FPS. And, the board is equipped with 912 36 Kb on−chip block RAM (BRAM), part of which is delivered to each agent to save and query the sequence model. The computational data of the sampling module of the video images and the subsequent processing of the model output results are stored in the DRAM.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2023 doi:10.20944/preprints202305.0746.v1

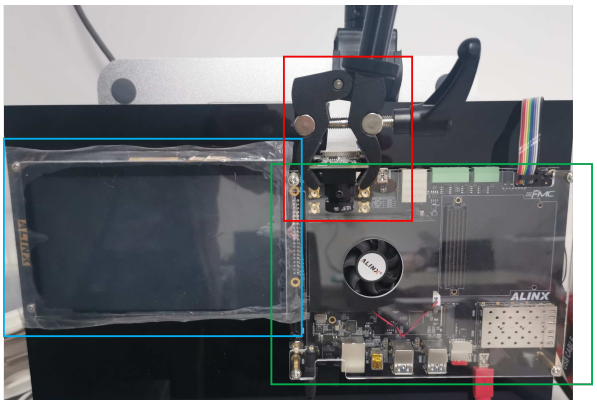


Figure 5. Prototype system.

The software interfaces of the prototype system are shown in Figure 6. In the hardware architecture MIPI controller outputs the video frames captured by the camera to the system and performs operations such as frame decoding and background differencing on the video frames. After the frame difference calculation, the original image and differencing results are quickly transferred to PS through AXI VMDA. PS binary encodes the differencing results and then transmits them to Agent Network to perform clustering recognition. The application receives and aggregates the clustering model recognition results from the agent network, and performs discriminative clustering of all the results into several different types, and image labeling and modification of the original image combined with the clustering results.

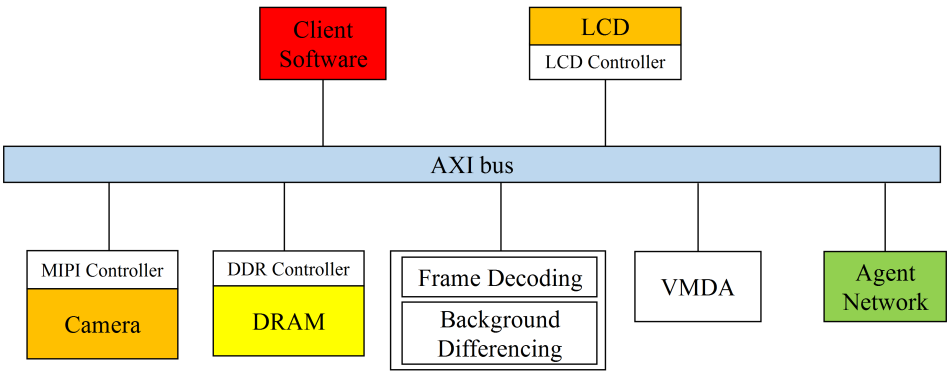


Figure 6. Prototype system working framework.

The internal processing logic of each agent in the prototype system is shown in Figure 7. The Agent Network receives binary encoded data that flows through each agent's Connector module. If an agent is working, information is recorded in the Connector module of that agent about the working mode, the area where the data is received, the level it is in, etc. If the binary-encoded data flows through, it is recorded in the Connector module of that agent. If the flow through the binary encoded data matches, it is saved; otherwise, it is passed on with assistance. Of course, when the subdivision has been carried out in the screen monitored by that agent, the transmitted data is saved and passed on at the same time to ensure that higher-level agents can receive the data. In this way, on the one hand, the data flow can save the transmission overhead, on the other hand, can be effectively managed. Each agent is internally applied with a separately used BRAM to ensure the efficiency of model queries and to avoid I/O problems caused by frequent use of mass storage.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2023 doi:10.20944/preprints202305.0746.v1

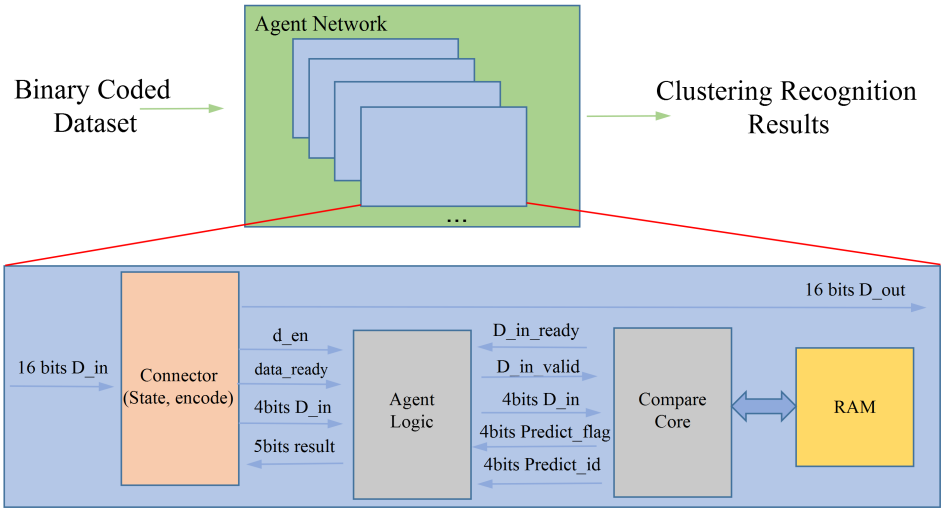


Figure 7. Structure diagram of agent network operations.

5. Experiment and Result

Our design uses a four-tier structure and requires a fully uniform partitioning case, the design applies 1 + 4 + 16 + 64, totaling 85 computational agents involved in the computational process. If the number of layers increases, the maximum number of computational nodes increases exponentially. We use the street view dataset [4] for validation, using a MIPI camera to capture the dataset frames and process them in real-time. Figure 8 shows the effect of self-organized dynamic segmentation of part of the upper left corner of the screen in Figure 4 (The area is already a proxy monitoring range for the second layer). It can be found that as the number of moving vehicles increases, the system gradually and dynamically performs a more fine-grained regular summary based on the changing clustering categories. The agents used for the computation are configured in regions where more variation exists, such as the upper right part of the region displayed in Figure 8, while its upper left part is largely unsegmented, as the region is mostly static.

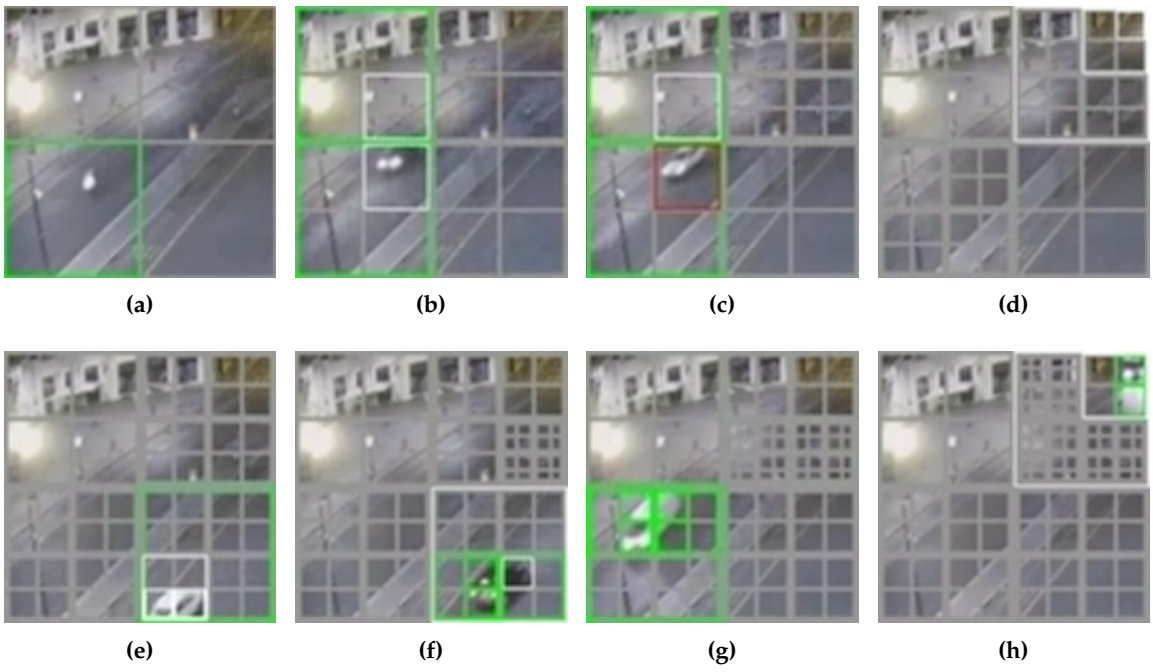


Figure 8. Dynamic division scenarios in operation.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2023 doi:10.20944/preprints202305.0746.v1

Through the clustering recognition behavior of SMLACS, the output results of the node network after each frame processing can spontaneously compose the form of a structure with the multi-layer classification effect. Depending on the clustering results, the classification is performed at different levels of fineness, and the higher the fineness, the more categories there are.

A cross-sectional comparison of this work with other results in terms of hardware resource utilization and performance is shown in Table 1. Since this work is designed with real-time output in addition to the implementation of the BE+LCS+MLS algorithm, the resource utilization of the full function and only the algorithm part are shown separately. The comparison shows that this work uses almost no DPS resources because it does not use floating point computation and matrix product, and the memory device BRAM, which is a key dependency, is not used more than other results. In terms of FFs and LUTs, the relative number of uses is higher than the other results, partly because a total of 85 agents were requested to participate, but the actual number of agents used is less than at the theoretical level because some of the stationary regions are not assigned agents to work on; on the other hand, it is because the FPGA platform provides an extremely rich of resources(FFs totaling 548,160, the utilization rate of 10.36 % and the system utilization rate of 25.49 %; LUTs totaling 274,080, the utilization rate of 32 % and the system utilization rate of 58.86 %). So, few optimization operations such as resource reuse are performed during the software compilation phase.

Table 1. Comparison with the state of the art in the field.

Work				Specs				
Implementation	Algorithm	Resolution	Frame Rate	Architecture	FFs	LUTs	DSPs	BRAMs
This Work	BE + LCS + MLS	1024 × 720	60+	UltraScale+ (200 MHz)	139699	161327	8	161
This Work (with output)	BE + LCS + MLS	1024 × 720	60+	UltraScale+ (200 MHz)	56910	87726	N/A	85
Seong [10]	Mono−Scale L&K	800 × 600	170	Virtex−6 (94 MHz)	17139	43228	54	132
Barranco [11]	Multi−scale L&K	640 × 480	32	Virtex−4 (83 MHz)	46122	51879	124	244
Wei [12]	Ridge Regression	640 × 480	30	Virtex−4 (94 MHz)	30368	45240	45	N/A
Tomasi [13]	Phase−based	512 × 512	40	Virtex−4 (42 MHz)	49762	76828	160	132
Seyid [14]	HBM + Refinement	640 × 480	39	Virtex−7 (200 MHz)	72864	36432	48	65

And the operating power consumption of SMLACS is shown in Table 2. The total power consumption when the platform is running is about 5.059 W, of which the static power consumption is about 0.762 W, accounting for about 15 % of the total power consumption. In terms of dynamic power consumption, counting from the type division, the consumption of clocks is about 0.504 W accounts for 12 %; the logic part is 411 mW, which is about 10 %; there is about 144 mW consumption for BRAM access, which only accounts for 3 %, while the PS side consumes 2.594 W, which accounts for 60 % of the overall. This is because while the algorithm part is processing the analysis, it is also processing and displaying the effect based on the sampled video and clustering results, and this part occupies a large amount of consumption. The power consumption in terms of filtering out the video output for BE+LCS+MLS is about 1.162 W, where the consumption of a single agent is only 14 mW, while the access consumption of all agents to BRAM is only 76 mW.

Table 2. Total dynamic and static power dissipation.

		Power	Rate	Power	Rate
Device Static	PS	0.1 W	13%	0.762 W	15%
	PL	0.662 W	87%		
Dynamic Power	Clocks	0.504 W	12%	4.297W	85%
	Signals	0.425 W	10%		
	Logic	0.411 W	10%		
	BRAM	0.144 W	3%		
	DSP	0.006 W	< 1%		
	PS	2.594 W	60%		

6. Conclusions

Behavior recognition and anomaly detection require analysis and processing of video information. Accurate computational approaches are well suited for object and feature classification, which places demands on computational volume, processing speed, and power consumption. The focus on the

motion process is different from the static information-based classification approach, where the motion process inherently represents a specific behavior pattern. Based on background modeling and optical flow method, our proposed SMLACS uses binary encoding to reduce the computational demand, and the direct way of extracting the motion change pattern between frames can be well implemented to summarize the motion information. We designed a prototype verification system based on FPGA platform and tested SMLACS on it by real-time processing. The experimental results show that SMLACS can achieve the behavior clustering recognition capability of unsupervised video and can use fewer resources to significantly reduce the computational effort, provide real-time processing capability, and further reduce power consumption.

In addition to the functional improvements needed for this work, there are also improvements to be made regarding self-organization. Also taking traffic intersections as an example, if an intersection is suddenly opened after a long period of closure, then the corresponding way of forming multi-layer agent connections should be dynamically changed accordingly. The scheme presented so far only involves the self-organization of the multi-layer agent structure in the face of unknown movements, but does not add dynamic changes, such as the agent of a layer choosing to return to the idle state to wait for the opportunity to re-engage with the multi-layer structure. The self-organized ground structure cannot be executed in a centrally controlled form, which requires point-to-point transfer communication involving change agents, and also requires consideration of the reality that hardware resources cannot be moved and programs are fixed when loaded.

Author Contributions: Conceptualization, X.C., X.Q., S.L.; methodology, X.C. X.Q., A.Y.; software, X.Q. and A.Y.; validation, X.Q. and X.C.; formal analysis, X.Q. and X.C.; investigation, W.D. and W.Y.; resources, A.Y., X.Q. and L.L.; data curation, X.Q., X.C. and W.D.; writing—original draft preparation, X.Q.; writing—review and editing, X.C. and W.Y.; visualization, X.C. and X.Q.; supervision, L.L and W.Y.; project administration, Z.S. and L.L.; funding acquisition, Z.S. And All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Strategic Priority Research Program of OF the Chinese Academy of Sciences grant number XDB44010200.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

Abbreviations

The following abbreviations are used in this manuscript:

BE	Binary Encoding
MLS	Multi-Layer Structure
LCS	Longest Common Sequence

References

1. McLaughlin, N.; Martinez del Rincon, J.; Miller, P. Recurrent Convolutional Network for Video-Based Person Re-identification. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1325–1334.
2. Feichtenhofer, C.; Pinz, A.; Zisserman, A. Convolutional Two-Stream Network Fusion for Video Action Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1933–1941.
3. Tran, D.e.a. Learning Spatiotemporal Features with 3D Convolutional Networks. 2015 IEEE ICCV, 2015, pp. 4489–4497.
4. skylinewebcams, 2021.
5. Wang, H.; Schmid, C. Action Recognition with Improved Trajectories. 2013 IEEE ICCV, 2013, pp. 3551–3558.

6. Horn, B.K.P.; Schunck, B.G. Determining Optical Flow. *Artif. Intell.* **1981**, *17*, 185–203. doi:10.1016/0004-3702(81)90024-2.
7. Lucas, B.D.; Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1981; IJCAI'81, p. 674–679.
8. Lazcano, V.; Rivera, F., GPU Based Horn-Schunck Method to Estimate Optical Flow and Occlusion; 2019; pp. 424–437. doi:10.1007/978-3-030-14812-6_26.
9. Gultekin, G.K.; Saranli, A. An FPGA based high performance optical flow hardware design for computer vision applications. *Microprocess. Microsyst.* **2013**, *37*, 270–286. doi:https://doi.org/10.1016/j.micpro.2013.01.001.
10. Seong, H.S.; Rhee, C.E.; Lee, H.J. A Novel Hardware Architecture of the Lucas–Kanade Optical Flow for Reduced Frame Memory Access. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 1187–1199. doi:10.1109/TCSVT.2015.2437077.
11. Barranco, F.; Tomasi, M.; Diaz, J.; Vanegas, M.; Ros, E. Parallel Architecture for Hierarchical Optical Flow Estimation Based on FPGA. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2012**, *20*, 1058–1067. doi:10.1109/TVLSI.2011.2145423.
12. Wei, Z.; Lee, D.J.; Nelson, B.E.; Archibald, J.K. Hardware-Friendly Vision Algorithms for Embedded Obstacle Detection Applications. *IEEE Trans. Circuits Syst. Video Technol.* **2010**, *20*, 1577–1589. doi:10.1109/TCSVT.2010.2087451.
13. Tomasi, M.; Vanegas, M.; Barranco, F.; Daz, J.; Ros, E. Massive Parallel-Hardware Architecture for Multiscale Stereo, Optical Flow and Image-Structure Computation. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 282–294. doi:10.1109/TCSVT.2011.2162260.
14. Seyid, K.; Richaud, A.; Capoccia, R.; Leblebici, Y. FPGA-Based Hardware Implementation of Real-Time Optical Flow Calculation. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 206–216. doi:10.1109/TCSVT.2016.2598703.