A VCA Protocol Based Multi-level Flexible Architecture on Embedded PLCs for Visual Servo Control

***Abstract*—We propose a flexible multi-level architecture which is based on a VCA protocol to address the problem of integration of PLC system, motion control system and visual system in ePLCs. The multi-level includes flexible layer, control layer and algorithm layer. The VCA protocol is designed to data interaction between the layers. Correspondingly, a customized hardware, memory allocation, multithreading structure are described to support the proposed flexible software architecture.**

***Index Terms*—motion control, visual servo control, embedded PLC, multi-level architecture**

I. Introduction

Integration technologies is driving the industrial automation [1]. The continuously integrations of sensors, controllers, robots, tools, etc. bring the concepts of self-aware equipment, intelligent factory and CPS, etc. [2], [3]. Recent researches [4]–[6] introduce the development of integration technologies. In industrial automation, PLC system, motion control system and visual system become more and more important and inseparable [7]–[9]. On the other hand, the advances of edge computing, fog computing and edge artificial [10]–[12] put forward new challenges on edge equipment of these systems.

A. Motivations

Recent advances in image processing and pattern recognition contribute to the thriving of visual system which has been applied in various fields. By extracting features from the image, the visual system could obtain parameters to replace human visual system to address lots of tasks, specially, the ever-growing requirements in industrial applications, e.g. works should be finished in dangerous environment, human vision is difficult to satisfy or numerous visual systems are needed in some large scale industrial production.

After the visual processing, in most cases, the motion control system as the power of automation is constantly needed to drive some actuators to finished some severe tasks which remarkably benefit of the replacement of large labor force.

On the other hand, PLCs have become a base of automation owning to its high reliability and easy programming [13]. Furthermore, Lots of researches are focusing on it to extremely extend its applied field. For instance, [14], [15] guarantee the reliability by verifying the program of PLCs, [16], [17] improve the performance of PLCs using advanced algorithms, [18] alleviates the development complexity of PLCs with a special software structure, [19] pose methods to update PLC programs dynamically.

In addition, the visual system, motion control system and PLC system are becoming increasingly inseparable. [20] is a typical case that describes how the three parts collaborate. The visual system analyzes the context and get error put into the motion control system. Simultaneously the PLC is judging the information, such as the position limitation of the every axis, to make some logical judgments accordingly.

Hence, how to pose a flexible structure to the integration of visual system, motion control system and PLC system attracts us.

B. Related Works

Visual control system is combined of special motion control system and visual system which is applied in various field, such as transport [21], circuit detection [22], sorting system, welding [20], assembling [23], [24], robot [25], [26], unmanned aerial vehicles [27], [28] and sorting [29]. These works address their problems in relevant fields. However, all these solutions are based on special motion control system and visual system.

On the other hand. The integration of logic control and motion control has variously deep researches [30]–[33]. [30], [33] realize the motion control directly in PLC. [34]–[36] use motion control module collaborated with PLC to implement their applications. However, the development method in these papers is disordered. Therefore, in 2005, PLCopen organization has released a related standard [37] which standardizes the motion control in PLC. Based on this standardization, [38] provides an advanced implementation in distributed automation system and companies, such as 3S [39], provide some tools. [18] poses a customized real-time compilation method to reduce the development complexity.

Above works provide impressive integrations on visual servo control system and PLC with motion control functions, however there are few papers discussing the integration of visual system, motion control and PLC. Most of applications are focusing on their applications with three individual systems, such as [20]. Hence, an integration structure of PLC system, motion control system and visual system should be provided to reduce the complexity and expand the application fields.

C. Our Contributions

We propose a flexible multi-level architecture which is based on a VCA protocol to address the problem of integration of PLC system, motion control system and visual system in ePLCs. The multi-level includes flexible layer, control layer and algorithm layer. The VCA protocol is designed to data interaction between the layers. Correspondingly, a customized hardware, memory allocation, multithreading structure are

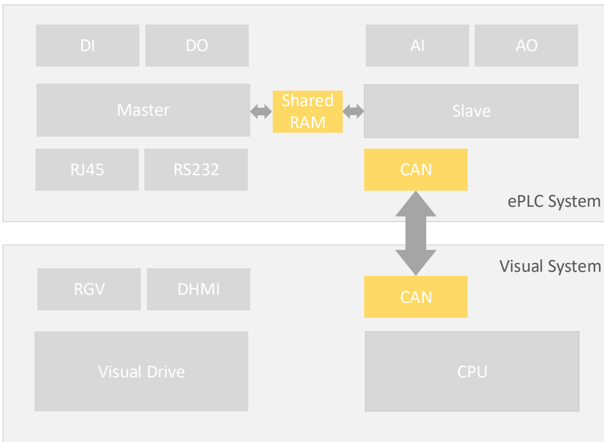

Fig. 1. A typical ES which adopts two processor architecture. The shared memory is used to transfer data between master and slave processors. The communication between ES and VS adopt the CAN Bus.

described to support the proposed flexible software architecture. In the remaining paper, in Section II, the hardware ans software structure, memory allocation and thread structure are introduced. In Section III, the mechanism of VCA protocol is addressed in detail. In Section IV, we illustrate the execution process of the proposed system. Then, we implement two cases which are binocular catching robot and winding machine with visual system in Section V. In the last section, we conclude our works.

II. System Architecture

A. Hardware Structure

The hardware is comprised of ePLC system (ES)and visual system (VS). The ES is a customized structure. The number of DI/DO, AI/AO and controlled servo system could be increased according to applications. Fig. 1 shows a typical ES which adopts two processor architecture. The shared memory is used to transfer data between master and slave processors. The communication between ES and VS could adopt multiple protocols, such as TCP, modbus, CAN, etc. In the shown Fig. 1, we adopt the CAN Bus.

B. Software Structure

In normal visual control system, the typical software structure is seen as in the left of Fig. 2. To our best knowledge, the VS contains several visual algorithms and every algorithm could extract some required parameters form pictures or videos. Meanwhile, the ES is comprised of modules which conclude logic part and algorithms and algorithms here are main motion control algorithms. Regarding the normal development method of visual servo control, the VS and ES is developed individually and using the communication protocol combines this two parts. However, this method should be always redesigned the programs in both systems because of the various visual algorithms and motion control algorithms. In addition, the logic and motion control are mixed together

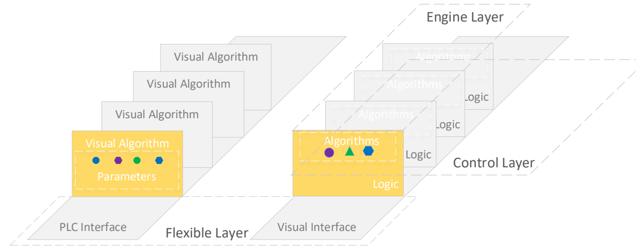

Fig. 2. Multi-level flexible architecture contains three layers: flexible layer, control layer and algorithm layer.

to develop in ES and there are also lots of communication protocols (EtherCat, Modbus, Can, etc.). Considering the high complexity of today's applications, it will be a cumbersome task.

Hence, we propose a multi-level flexible architecture which is based on VCA protocol. As shown in Fig. 2, the architecture contains three layers: flexible layer (FL), control layer (CL) and algorithm layer (AL).

Flexible layer: these layer is responsible for joint the VS and ES and consists of PLC interface and visual interface. The parameters are framed with VCA protocol frame (henceforth , abbreviated form of PF) interacted between VS and ES. Furthermore, a special protocol template (PT) is saved in both systems to illustrate the protocol.

Control layer: this is independent of algorithm layer to cope with the logic tasks. This make it possible to run the logic task and algorithm task in different processors.

Algorithm layer: it is mainly comprised of types of algorithms which could be built in individual processors for algorithms with high performance requirement.

C. Thread Structure

We adopt the analogous thread structure in [18]. Three special threads introduced below:

1) Visual Thread. This thread is responsible for interaction with VS. It will receive the protocol frame from the VS and save it to relevant address.
2) Control Thread. Control thread mainly execute logic program, deframe the protocol frame and interaction data with algorithm thread.
3) Algorithm Thread. Algorithm thread runs in slave processors. It is used to implement data interaction between processors and execution of algorithms.

D. Memory Allocation

The dedicated storage area of PLC in memory is made up of bit data area (M area) and byte data area (D area). Meanwhile, we regard M area and D area as set M of bit and set D of byte. Henceforth, we have three definitions below.

Definition 1 If $\exists$ set S, we define its subscripted lowercase letter s_i as an element of S and the subscripted i is used to distinguish the elements.

Definition 2 If $\exists S \subseteq M$ and $(\forall s_i \in S) \in \{0,1\}$. Meanwhile, each element s_i has four operators: $\mathcal{S}_0(s_i)$ assigns

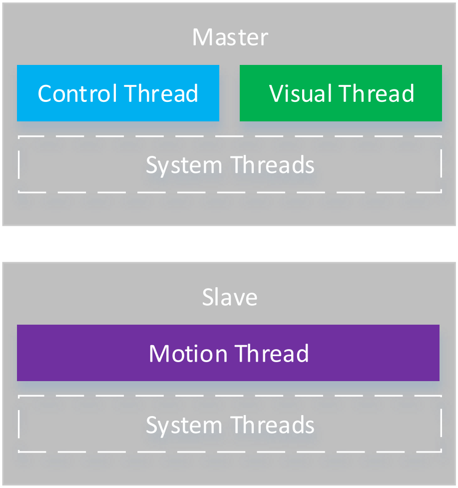

Fig. 3. Thread structureThe with three special threads: visual thread, Control thread, algorithm thread.

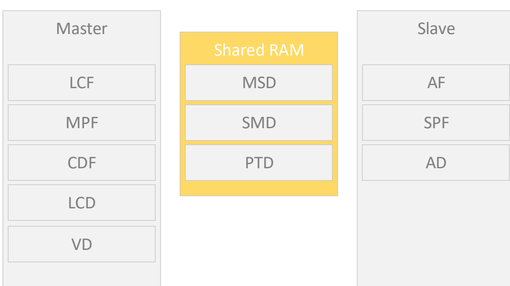

Fig. 4. Memory allocation of master processor, shared ram and slave processor.

0 to s_i, $\mathcal{S}_1(s_i)$ assigns 1 to s_i, $\mathcal{J}_0(s_i)$ represents that the value of s_i is 0, $\mathcal{J}_1(s_i)$ means that the value of s_i is 1. Then we define the set S has $\mathcal{B}$ attribute.

Definition 3 If $\exists S \subseteq D$ and $\forall s_i \in S$ has 4 bytes. We define the set S has $\mathcal{D}$ attribute.

Fig. 4 shows the memory allocation of master processor, shared RAM and slave processor. The shared RAM is a special structure to fast data interaction between master and slave processors. In more general cases, the $MtoS$ is located in master processor, the $StoM$ is located in slave processor and the PT is in both master and slave processors.

LCF (Logic Control Flag Area): the flag are used to start the modules. It has $\mathcal{B}$ attribute.

MPF (Master Processor Data Interaction Flag Area): it contains begin data transfer flag from master to slave (MSB), transfer state of master from master to slave (MSF), acknowledge flag of master from master to slave (MSA) and transfer state of master from slave to master (MSS). All of them have $\mathcal{B}$ attribute.

CDF (control frame saved flag area): this flags denote whether the control frame is saved. It has $\mathcal{B}$ attribute.

LCD (Logic Control Data Area): these data will be used to store the control frame. It has $\mathcal{D}$ attribute. Every element lcd_i is associated with a specified module.

VD (VCA protocol frame data area): all the frames received from the VS are stored in this area.

MSD (Master Processor Data Interaction Data Area): an area stores the data delivered from slave processors and it has $\mathcal{D}$ attribute.

SMD (Slave Processor Data Interaction Data Area): an area stores the data delivered from master processor and it has $\mathcal{D}$ attribute.

PTD (Protocol Template Data Area): the protocol template is stored in this area and could be read by both processors.

AF (Algorithm Flag Area): it includes algorithm flag of execution (AFE) and algorithm flag of state (AFS). Both of them have $\mathcal{B}$ attribute.

SPF (Slave Processor Data Interaction Flag Area): this area includes the begin data transfer flag from slave to master (SMB), transfer state of slave from slave to master (SMF), acknowledge flag of slave from slave to master (SMA) and transfer state of slave from master to slave (SMS). All of them have $\mathcal{B}$ attribute.

AD (Algorithm Data Area): these data help specified algorithm executing. It has $\mathcal{D}$ attribute.

We define the $\mathcal{P}$ to interact data between master processor and slave processor. It contains transferring data from master to slave ($\mathcal{P}_{mts}$) and transferring data from slave to master ($\mathcal{P}_{stm}$) which is defined below:

$$\begin{cases} \mathcal{P}_{mts} = \mathcal{U}(msb_i, msf_i, sma_i, sms_i, smd_i) \\ \mathcal{P}_{stm} = \mathcal{U}(smb_i, smf_i, msa_i, mss_i, msd_i) \end{cases} \tag{1}$$

Where $\mathcal{U}$ is the function to implement the process of data interaction between master and slave processors. $\mathcal{P}_{mts}$ and $\mathcal{P}_{mts}$ use the same function $\mathcal{U}$.

The process of $\mathcal{P}_{mts}$ is seen below: $\mathcal{S}_1(msb_i) \rightarrow \mathcal{S}_1(msf_i) \rightarrow send(smd_i) \rightarrow \mathcal{S}_0(msb_i) \rightarrow \mathcal{S}_1(sms_i) \rightarrow check(smd_i) \rightarrow \mathcal{S}_1(sma_i) \rightarrow \mathcal{S}_0(sms_i) \rightarrow \mathcal{S}_0(sma_i) \rightarrow \mathcal{S}_0(msf_i)$.

Where $send(msd_i)$ means sending data to msd_i in slave processor. $check(msd_i)$ means to check data of msd_i.

III. Integration of VS and ES

A. VCA Protocol Template

In cater to most applications, we propose a VCA-based multi-level flexible architecture. As shown in Fig. 5, a protocol template (PT) is adopted to support various types of implementations and a PT uniquely corresponds to a type of application. In the flexible architecture, users only need to redesign and reload the PT and then they can reuse the visual servo system again. The PT could be loaded into a stationary address of visual system and ePLC system. After restarting both systems, it will be put into a fixed area of RAM. The parsing modules form both systems will read it when parsing the protocol data.

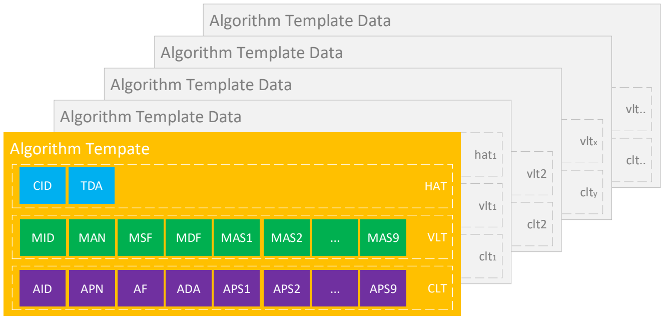

Fig. 5. A PT uniquely corresponds to a type of application. It contains four parts: head of protocol template, visual layer template , control layer template and algorithm layer template.

The PT is defined below:

$$\begin{cases} PT = \{HPT, VLT, CLT, ALT\} \\ HPT = \{CID, TDA\} \\ VLT = \{MID, MAN, MSF, MDA, MAS, CDATA\} \\ CLT = \{AID, APN, AF, ADA, APS, ADATA\} \\ ALT = \{PID, PDATA\} \end{cases} \tag{2}$$

The PT contains four parts: head of protocol template (HPT), visual layer template (VLT), control layer template (CLT) and algorithm layer template (ALT). Every part explains as follows:

1) HPT: this part includes communication unique ID (CID), template data storage address (TDA). Every PT only has one HPT.
2) VLT: it consists of module unique ID (MID), module contained algorithm number (MAN), module start flag (MSF), module data start address (MDA), module contained algorithm IDs (MAS) . VLT is not $\emptyset$. Every MAS include MAN algorithm IDs.
3) CLT: it is comprised of algorithm unique ID (AID), algorithm contained parameter number (APN), AF, algorithm data start address ADA, algorithm contained parameter IDs (APS). CLT is not $\emptyset$. Every APS include APN parameter IDs.
4) ALT: it contains parameter unique ID (PID). ALT is not $\emptyset$.

B. VCA Protocol Frame

The VCA protocol frame (PF) contains control frame and algorithm frame in its data field as shown in 6.

1) PF: it consists items of MID, visual frame length VFL, $CDATA$ and cyclic redundancy check CRC.The $CDATA$ contains several control frames. The MIDs both in PT and PF need one-to-one correspondence.
2) Control frame: it is comprised of AID, control frame length (CFL) and $ADATA$. The $ADATA$ includes several algorithm frames. The AIDs both in control frame and PF need one-to-one correspondence.
3) Algorithm frame: it contains PID, $PDATA$. PID is also the address of $PDATA$. The PIDs both in algorithm frame and PF need one-to-one correspondence.

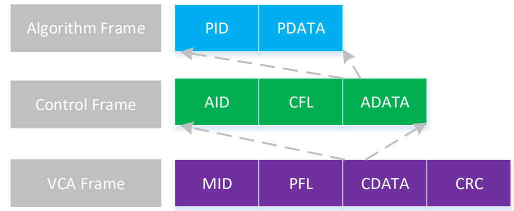

Fig. 6. VCA protocol frame contains control frame and algorithm frame in its data field.

C. Framing of PF

In VS, the PLC interface will frame the PF and transfer it to PLC. The framing contains three steps: VCA framing, CL framing, AL framing.

***VCA* Framing**: the process is realized as the Algorthm 1. It searches the PT with mid to find the relevant vlt_x. Through it, the VS can obtain the msf and mda. According to man, call the next step (VL Framing) to gain the $cdata$. Then calculate the length (vfl) and crc to finish the VL framing.

***CL* Framing**: Algorthm 2 illustrates the process. The VS seeks the FT to gain the clt_y which contains the af and ada. According to apn, call the next step (AL Framing) to obtain the $cdata$ and then calculate the length (cfl) to finish the CL framing.

***AL* Framing**: Algorthm 3 shows the process. The VS obtains the alt_z frome FT and then combines the pid and $pdata$.

Algorithm 1: $VCAFraming$

Input: mid, VED
Output: PF
SearchPT (mid, vlt_x);
$PF.mid \leftarrow mid$;
for $i = 0;i < vlt_x.man;i++$ **do**
  ALFraming ($vlt_x.mas$[i], VED,$cdata$);
end
$PF.cdata \leftarrow cdata$;
$PF.vfl \leftarrow$ Length ($cdata$) + 8;
$PF.crc \leftarrow$ CRC16 (PF);

Algorithm 2: $CLFraming$

Input: aid, VED, $cdata$
Output: $cdata$
SearchPT (aid, clt_y);
Create clf;
$clf.aid \leftarrow aid$;
for $i = 0;i < clt_y.apn;i++$ **do**
  ALFraming($clt_y.aps$[i], $visualData$, $adata$);
end
$clf.cfl \leftarrow$ Length($adata$) + 8;
$clf.adata \leftarrow adata$;
$cdata \leftarrow cdata$ + clf;

Algorithm 3: $ALFraming$

Input: pid, VED,$pdata$
Output: $pdata$
Search the PT with pid and get alt_z;
Create alf;
$alf.pid \leftarrow pid$;
$alf.pdata \leftarrow VED[pid]$;
$pdata \leftarrow pdata + alf$;

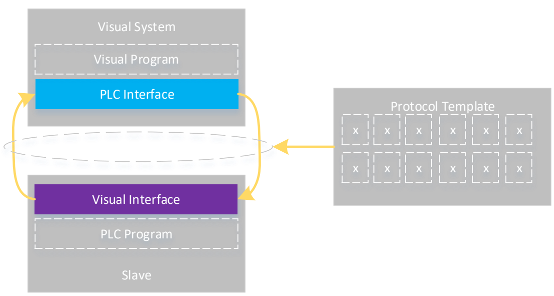

Fig. 7. Data Interaction Based on VP Protocol.

D. Deframing of PF

The process of deframing of PF is contained in ES which has three parts: VCA deframing, CL deframing and AL deframing.

***VCA* deframing**: as illustrated in Algorithm 4, at first, check the CRC. If the frame data is right, obtain the mid and msn form the start four and following four bytes data of PDF, respectively. Search the PT to gain the relevant vlt_x. Send the msn bytes start form the eighth byte of PDF to the address mda of vlt_x.

***CL* deframing**: according to mid, search PT to find the relevant vlt_x. Loop deal with the man times of the control frame. Use the aid to find clt_y and then send the $atata$ to correlative address of shared RAM. Algorithm 6 describes the process.

***AL* deframing**: use aid to get its clt_y. Loop cope with the algorithm frame apn times. Send the parameters to the algorithm. The process is shown in Algorithm 6.

Algorithm 4: $VLDeframing$

Input: PDF
$mid \leftarrow$ four bytes start form PDF;
searchPT (mid, vlt_x);
CRC16();
$msn \leftarrow$ four bytes start form $PDF + 4$;
$CassMemM[vlt_x.mda] \leftarrow msn - 8$ bytes start form $PDF + 8$;
$PDF \leftarrow PDF + msn$;

Algorithm 5: $CLDeframing$

Input: mid
searchPT(mid, vlt_x);
$mda \leftarrow vlt_x.mda$;
for $i = 0; i < vlt_x.man; i++$ **do**
  searchPT($vlt_x.mas[i]$, clt_y);
  $CassMemS[clt_y.apa] \leftarrow clt_y.vfl - 8$ bytes start form $mda + 8$;
  $mda \leftarrow mda + clt_y.vfl$;
end
clean;

Algorithm 6: $ALDeframing$

Input: aid
searchPT(aid, clt_y);
$ada \leftarrow vlt_x.ada$;
for $i = 0; i < clt_y.apn; i++$ **do**
  $CassMemA[CassMemA[ada]] \leftarrow CassMemA[ada + 4]$;
  $ada \leftarrow ada + 8$;
end
clean;

IV. System Operation Mechanism

A. Implementation of Flexible Program and Execution

The FL contains two parts: PLC interface and visual interface. They interact with each other using the agreed communication protocol which is defined as CID in HPT of PT.

The PLC interface is responsible for get parameters of motion control form VS and interact with ES. It includes three steps blew.

Step 1: after image processing, the VS extracts the useful data and stores into visual extracted data (VED) in which the parameters could be indexed by the PID.

Step 2: frame them into PF with the Algorithm 1, 2 and 3.

Step 3: transfer the VCA protocol frame to $ePLC$ with the relevant communication protocol in CID.

Visual interface is in the $ePLC$ designed to interact with VS and it contains two steps below.

Step 1: receive PFs from VS according the CID in HPT.

Step 2: save them in the ES. There are two pointers: PDF and PSP. PDF points to the address of deframing PF and PSP points to the address of saving PF. The PDF will point to the next address of PF if a PF is deframed. After saved the PF, the PSP will point to the next new address according the vfl of PF.

B. Implementation of Control Program and Execution

Control program CP is responsible for organizing the modules, executing the logic program, deframing the PF and interacting the data with algorithm program. $CP = \{VLDP, CLDP, IP, LP, PIP\}$, where CLP is the control

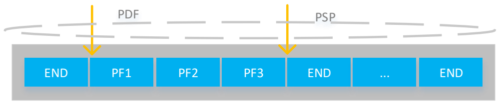

Fig. 8. PDF and PSP.

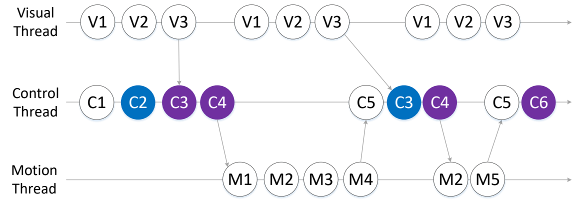

Fig. 9. Visual servo control system

layer deframe program. IP is the initial program which is used to initialize the data and state. LP is the logic program. PIP is the processor interaction program implemented to transfer and receive data between processors. The implementation of control program is described below.

Step 1: if $PDF! = PSP$, execute the $VLDP$ which means to call Algorithm 4

Step 2: traverse the LCF to check whether there is a module needed to execute. If $\exists\ \forall\ lcf_i \in LCF$ is 1, go to step 2.

Step 3: execute the IP to initialize the data and state and then execute the LP to find the called algorithm. If $\exists\ \forall\ af_i \in AF$ is 1, go to step 4.

Step 4: execute the $CLDP$ to call Algorithm 5.

Step 5: the $\mathcal{P}_{mts}$ is used in PIP to inform the slave processor to receive parameters and start algorithms and clear the relevant af_i.

Step 6: read and process the feedback data form the slave processor.

C. Implementation and Execution of Algorithm Program

The Algorithm Program (AP) is in EL. $AP = \{\mathcal{P}_{stm}, AS, ALDP\}$, where $\mathcal{P}_{stm}$ feeds back data to master processor, AS contains all algorithms and $ALDP$ deframe the AL frame. The execution of algorithm program is introduced below.

Step 1: traverse the LCF to check whether there is a algorithm needed to execute. If $\exists\ \forall\ af_i \in AF$ is 1, go to step 2.

Step 2: ALDP is executed to call Algorithm 5. The parameters are transfered to correlative as_i.

Step 3: as_i is executed and uses the $\mathcal{P}_{stm}$ to feed back data to the master processor.

D. Execution of Threads

Commonly, threads in master processor and in slave processors execute separately according to their priority and the interaction between control thread and algorithm threads occurs when using $\mathcal{P}_{mts}$ and $\mathcal{P}_{mts}$.

Visual thread consists of the following units.

V_1: receive PF from VS with the agreed communication protocol.

V_2: check PF and store it into VD.

V_3: call Algorithm 4 to deframe the PF and saved the $cdata$ into LCD.

Basic execution units of control thread are shown as follows:

C_1: start a module and initial it.

C_2: run logic program.

C_3: call Algorithm 5 to deframe control frame.

C_4: transfer data to slave processor by $\mathcal{P}_{mts}$.

C_5: deal with the feedback data.

C_6: end the module.

Motion thread contains the following basic execution units:

M_1: Start a algorithm.

M_2: Call Algorithm 6 to deframe the algorithm frame.

M_3: Execute the algorithm.

M_4: Feedback the data to master processor by $\mathcal{P}_{stm}$.

M_5: End the algorithm.

The threads execute as shown in Fig. 9. Visual thread (VT) continuously executes the unit $V1$ if there exits PF from VS, executes unit $V2$ to store the PF into VD at the address of PSP and execute $V3$ to deframe PF at the address of PDF and store its $cdata$ into LCD and $\mathcal{S}_1 cdf_i$. The control thread (CT) traverses LCF, finds $\exists \mathcal{J}_1 lcf_i$ and then executes the unit $C1$, executes unit C_2. If find a required execution algorithm and $\exists$ its $\mathcal{J}_1 mdf_j$ then CT starts unit $C3$ to deframe the control frame and then runs unit $C4$ and inform algorithm thread (AT) to execute M_1 unit. Next, AT executes unit $M2$ transferring data from SMD to AD. After that, the unit $M3$ is executed. During the running of algorithm, the unit $M4$ is continuously executed to feed data back to CT, meanwhile CT will execute unit $C5$ to response it until the execution of $M5$ and no other executed algorithms neither. During the running of the module, if the VT receives new PF and CT finds $\exists \mathcal{J}_1 cdf_i$, CT will execute unit C_3 and C_4 again. Next, the VT will run unit M_2 to update the parameters. If the MT finished all algorithms, then CT will run unit C_6 to finished the module and continue to traverse the LCF to find another required execution module.

V. Case Analysis

In this section, we introduce two scenarios of the proposed VCA protocol based integration method in ePLC. With the VCA protocol, users only need to code additional algorithms and change the PT when the scenario of visual servo system is changed.

A. Case 1 Binocular Catching Robot

As shown in Fig. 10, the binocular catching robot adopts two cameras to judge the position of the ball in the space. Through sending the continuously parameters to $ePLC$, the robot runs to the position to catch the ball. Finally, the robot will catch the ball. The cameras is xxx, the visual system is xxx, and the $ePLC$ is XXX. $ePLC$ uses tht TI F28M35 chip

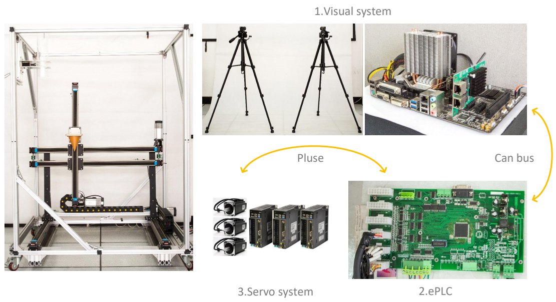

Fig. 10. Visual servo control system

TABLE I
PT OF BINOCULAR CATCHING ROBOT

CID	TDA	xxx	xxx	xxx	xxx	xxx
0X03000001	0x00					
MID	MAN	MSF	MDF	MAS1	MAS2	xxx
0X02000001	1	0X400	0X600	0X02000001		
AID	APN	AF	ADA	APS1	APS2	APS3
0X02000001	0X9	0X14A	0x1F4000	0X1	0X2	0X3
APS4	APS5	APS6	APS7	APS8	APS9	xxx
0X4	0X5	0X6	0x7	0X8	0X9	

with two cores: ARM Cortex M3 and TI C28x, which contains a shared RAM. The servo driver and motor are xxx and xxx respectively.

1) Design of PT: As

2) Design of Program:

3) Result:

B. Case 2 Winding Machine with Visual System

1) Design of PT:

2) Design of Program:

3) Result:

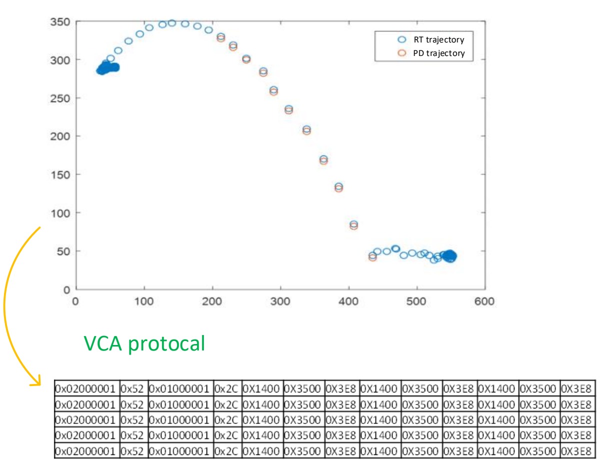

Fig. 11. The execution OF flexible program

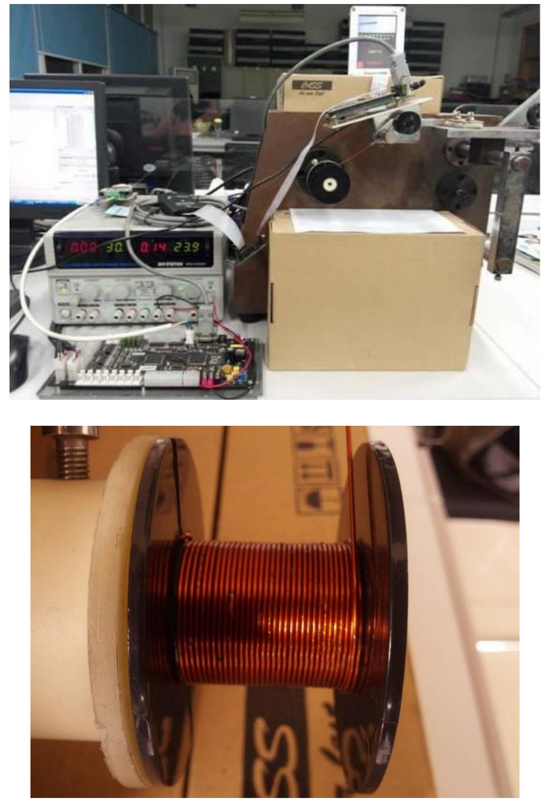

Fig. 12. Winding machine Based on Visual System

VI. Conclusion

We propose a flexible multi-level architecture which is based on a VCA protocol to address the problem of integration of PLC system, motion control system and visual system in ePLCs. The multi-level includes flexible layer, control layer and algorithm layer. The VCA protocol is designed to data interaction between the layers. Correspondingly, a customized hardware, memory allocation, multithreading structure are described to support the proposed flexible software architecture.

In the further research, we will implement a uniform development method of the visual servo control in ePLC.

References

[1] M. P. Kazmierkowski, “Integration technologies for industrial automated systems (zurawski, r., ed.; 2006) [book reviews],” *IEEE Industrial Electronics Magazine*, vol. 1, no. 1, pp. 51–52, 2007.

[2] J. Wan, B. Yin, D. Li, A. Celesti, F. Tao, and Q. Hua, “An ontology-based resource reconfiguration method for manufacturing cyber-physical systems,” *IEEE/ASME Transactions on Mechatronics*, vol. PP, no. 99, pp. 1–1, 2018.

[3] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, “Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory,” *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2018.

[4] A. W. Colombo, R. Schoop, and R. Neubert, “An agent-based intelligent control platform for industrial holonic manufacturing systems,” *IEEE Transactions on Industrial Electronics*, vol. 53, no. 1, pp. 322–337, 2006.

[5] A. Vaccaro, M. Popov, D. Villacci, and V. Terzija, “An integrated framework for smart microgrids modeling, monitoring, control, communication, and verification,” *Proceedings of the IEEE*, vol. 99, no. 1, pp. 119–132, 2010.

[6] E. Dean, K. R. Amaro, F. Bergner, I. Dianov, and G. Cheng, "Integration of robotic technologies for rapidly deployable robots," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2017.

[7] X. Feng, S. A. Velinsky, and D. Hong, "Integrating embedded pc and internet technologies for real-time control and imaging," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 1, pp. 52–60, 2002.

[8] T. N. Chang, B. Cheng, and P. Sriwilaijaroen, "Motion control firmware for high-speed robotic systems," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 5, pp. 1713–1722, 2006.

[9] X. Feng, R. Mathurin, and S. A. Velinsky, "Practical, interactive, and object-oriented machine vision for highway crack sealing," *Journal of Transportation Engineering*, vol. 131, no. 6, pp. 451–459, 2005.

[10] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1910–1920, 2017.

[11] W. Hou, Z. Ning, and L. Guo, "Green survivable collaborative edge computing in smart cities," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2018.

[12] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, "An edge-based architecture to support efficient applications for healthcare industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1.

[13] S. Hossain, M. A. Hussain, and R. B. Omar, "Advanced control software framework for process control applications," *International Journal of Computational Intelligence Systems*, vol. 7, no. 1, pp. 37–49, 2014.

[14] Y. Jiang, H. Zhang, H. Liu, X. Song, W. N. Hung, M. Gu, and J. Sun, "System reliability calculation based on the run-time analysis of ladder program," in *Proceedings of the 2013 9th joint meeting on foundations of software engineering*. ACM, 2013, pp. 695–698.

[15] Y. Jiang, H. Zhang, X. Song, X. Jiao, W. N. N. Hung, M. Gu, and J. Sun, "Bayesian-network-based reliability analysis of plc systems," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 11, pp. 5325–5336, 2013.

[16] S. Gerkšič, G. Dolanc, D. Vrančić, J. Kocijan, S. Strmčnik, S. Blažič, I. Škrjanc, Z. Marinšek, M. Božiček, and A. Stathaki, "Advanced control algorithms embedded in a programmable logic controller," *Control Engineering Practice*, vol. 14, no. 8, pp. 935–948, 2006.

[17] S. Dominic, Y. Lohr, A. Schwung, and S. X. Ding, "Plc-based real-time realization of flatness-based feedforward control for industrial compression systems," *IEEE Transactions on Industrial Electronics*, vol. PP, no. 99, pp. 1–1, 2016.

[18] H. Wu, Y. Yan, D. Sun, and S. Rene, "A customized real-time compilation for motion control in embedded plcs," *IEEE Transactions on Industrial Informatics*, 2018.

[19] D. Schütz, A. Wannagat, C. Legat, and B. Vogel-Heuser, "Development of plc-based software for increasing the dependability of production automation systems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2397–2406, 2013.

[20] H. Chen, K. Liu, G. Xing, Y. Dong, H. Sun, and W. Lin, "A robust visual servo control system for narrow seam double head welding robot," *International Journal of Advanced Manufacturing Technology*, vol. 71, no. 9-12, pp. 1849–1860, 2014.

[21] W. Xing, P. Lou, J. Yu, X. Qian, and D. Tang, "Intersection recognition and guide-path selection for a vision-based agv in a bidirectional flow network," *International Journal of Advanced Robotic Systems*, vol. 11, no. 1, p. 1, 2014.

[22] C. Y. Nian and Y. S. Tarng, "An auto-alignment vision system with three-axis motion control mechanism," *International Journal of Advanced Manufacturing Technology*, vol. 26, no. 9-10, pp. 1121–1131, 2005.

[23] Y. Wang, H. Lang, and C. W. D. Silva, "Visual servo control and parameter calibration for mobile multi-robot cooperative assembly tasks," in *IEEE International Conference on Automation and Logistics*, 2008, pp. 635–639.

[24] S. Xiao and Y. Li, "Visual servo feedback control of a novel large working range micro manipulation system for microassembly," *Journal of Microelectromechanical Systems*, vol. 23, no. 1, pp. 181–190, 2014.

[25] H. Wu, L. Lou, C. C. Chen, S. Hirche, and K. Kuhnlenz, "Cloud-based networked visual servo control," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 2, pp. 554–566, 2013.

[26] C. Y. Tsai, C. C. Wong, C. J. Yu, C. C. Liu, and T. Y. Liu, "A hybrid switched reactive-based visual servo control of 5-dof robot manipulators for pick-and-place tasks," *IEEE Systems Journal*, vol. 9, no. 1, pp. 119–130, 2017.

[27] N. Guenard, T. Hamel, and R. Mahony, "A practical visual servo control for an unmanned aerial vehicle," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 331–340, 2010.

[28] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre, "Landing of a quadrotor on a moving target using dynamic image-based visual servo control," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1524–1535, 2016.

[29] L. Y. Sun, G. M. Yuan, J. Zhang, Z. Liu, and L. X. Sun, "Automatic button cell sorting manipulator system research based on visual servo control," *Advanced Materials Research*, vol. 712-715, pp. 2285–2289, 2013.

[30] M. G. Ioannides, "Design and implementation of plc-based monitoring control system for induction motor," *IEEE Transactions on Energy Conversion*, vol. 19, no. 3, pp. 469–476, 2004.

[31] X. M. Shi, W. J. Fei, and S. P. Deng, "The research of circular interpolation motion control based on rectangular coordinate robot," *Key Engineering Materials*, vol. 693, pp. 1792–1798, 2016.

[32] N. Fang, "Design and research of multi axis motion control system based on plc," *Academic Journal of Manufacturing Engineering*, vol. 15, no. 1, pp. 17–23, 2017.

[33] A. Syaichu-Rohman and R. Sirius, "Model predictive control implementation on a programmable logic controller for dc motor speed control," in *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*. IEEE, 2011, pp. 1–4.

[34] W. F. Peng, G. H. Li, P. Wu, and G. Y. Tan, "Linear motor velocity and acceleration motion control study based on pid+velocity and acceleration feedforward parameters adjustment," *Materials Science Forum*, vol. 697-698, pp. 239–243, 2011.

[35] J. Qian, H. B. Zhu, S. W. Wang, and Y. S. Zeng, "A 5-dof combined robot platform for automatic 3d measurement," *Key Engineering Materials*, vol. 579-580, pp. 641–644, 2014.

[36] O. Co.Ltd, "Cs1w-mc221(-v1)/mc421(-v1) motion control units." *Operation Mannual*, 2004.

[37] P. T. C. 2., *Function blocks for motion control version 1.1*, 2005.

[38] C. Sünder, A. Zoitl, F. Mehofer, and B. Favre-Bulle, "Advanced use of plcopen motion control library for autonomous servo drives in iec 61499 based automation and control systems," *E & I Elektrotechnik Und Informationstechnik*, vol. 123, no. 5, pp. 191–196, 2006.

[39] S. S. S. GmbH, "Logic and motion control integrated in one iec 61131-3 system:development kit for convenient engineering of motion, cnc and robot applications." 2017.