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

Abstract—**The visual system, motion control system and programmable logic controller (PLC) system are becoming increasingly inseparable, however there are few papers discussing the integration of these three systems. Most of papers are focusing on their applications individually. We propose a flexible multi-level architecture which is based on a VCA protocol to address the integration problem. The multi-level includes flexible layer, control layer and algorithm layer. The flexible layer is adopted to seamlessly integrate visual system and embedded PLC ($ePLC$) system. 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. At last, we implement two cases using the proposed VCA protocol based flexible architecture in which the generality of the proposed flexible architecture is verified. In first case, the winding machine with visual system implements regular winding effect by the correction of θ. In second case, the binocular catching robot uses the camera to track the trajectory and send to $ePLC$. By adjusting the speed and position, the robot can catch the ball finally.**

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

I. Introduction

Integration technologies are 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 intelligence [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 scenarios, 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

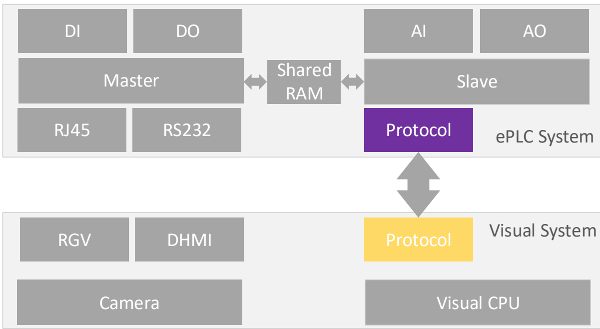

Fig. 1. A typical ES which adopts two processor architecture. The shared memory is used to transfer data between master and slave processors. The communication between ES and VS adopt the CAN Bus.

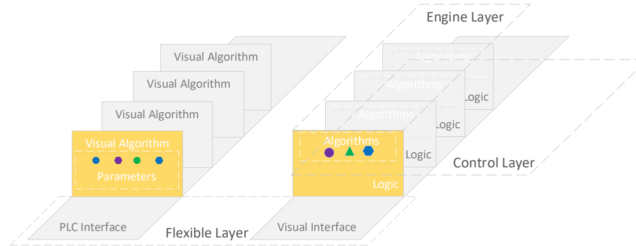

Fig. 2. Multi-level flexible architecture contains three layers: flexible layer, control layer and algorithm layer.

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 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

The typical software structure is seen as in Fig. 2. To our best knowledge, the VS commonly 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. The 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 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 to address the problem of lack of generic method to integrate the VS and ES. 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.

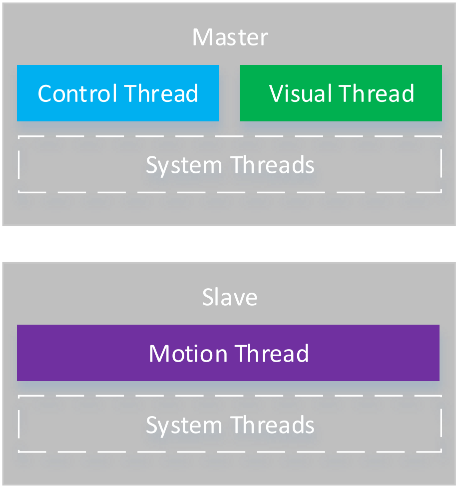

Fig. 3. Thread structureThe with three special threads: visual thread, Control thread, algorithm thread.

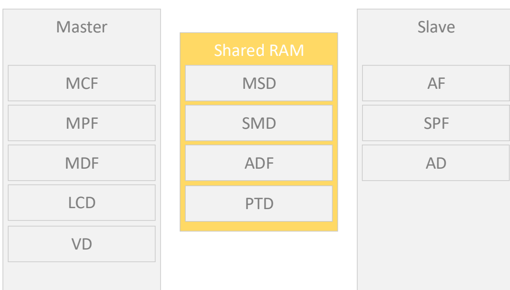

Fig. 4. Memory allocation of master processor, shared ram and slave processor.

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 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.

MCF (Module 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.

MDF (Module data saved flag area): this flags denote whether the module data are 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.

ADF (Algorithm data saved flag Area): the flags denote whether the algorithm data are saved. It has $\mathcal{B}$ 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 PT

In cater to most applications, we propose a VCA protocol based multi-level flexible architecture. As shown in Fig. 5, a

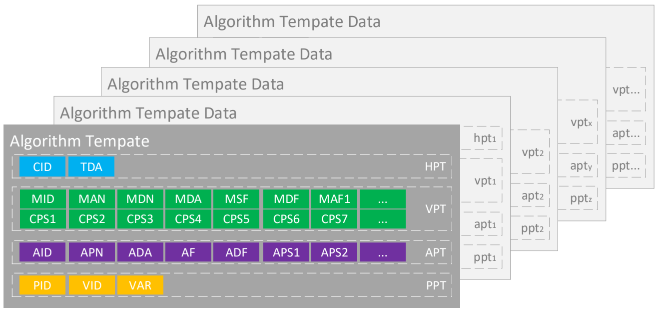

Fig. 5. A PT uniquely corresponds to a type of application. It contains four parts: head of protocol template, visual layer template , control layer template and algorithm layer template.

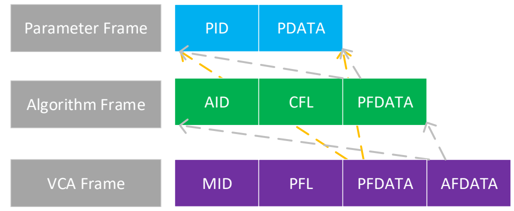

Fig. 6. VCA protocol frame contains parameter frames and algorithm frames in its data field and the algorithm protocol frame contains several parameter frames.

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 PF. The VCA protocol could be used to bidirectionally transfer PF using the same PT and algorithms of framing and deframing.

The PT is defined below:

$$\begin{cases} PT = \{HPT, VPT, APT, PPT\} \\ HPT = \{CID, TDA\} \\ VPT = \{MID, MAN, MDN, MDA, MSF, MDF, \\ \qquad\qquad \bigcup_{x=1}^{i} MAF_x, \bigcup_{x=1}^{j} CPS_x\} \\ APT = \{AID, APN, ADA, AF, ADF, \bigcup_{y=1}^{i} APS_y\} \\ PPT = \{PID, VID, VAR\} \end{cases} \tag{2}$$

The PT contains four parts: head of protocol template (HPT), VCA protocol template (VPT), algorithm protocol template (APT) and parameter protocol template (PPT). 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) VPT: it consists of module unique ID (MID), module contained algorithm number (MAN), , module contained data number (MDN), module data start address (MDA), module start flag (MSF), module data saved flag (MDF), module contained algorithm IDs (MAS_x) . VLT is not $\emptyset$. Every VPT includes MAN algorithm IDs.
3) APT: it is comprised of algorithm unique ID (AID), algorithm contained parameter number (APN), algorithm data start address ADA, algorithm data saved flag (ADF), algorithm contained parameter IDs (APS_y). CLT is not $\emptyset$. Every APT includes APN parameter IDs.
4) PPT: it contains parameter unique ID (PID), its relevant visual algorithm ID (VID) and the conversion ratio of visual algorithm parameters and motion algorithm parameters (VAR).

B. VCA PF

The VCA protocol frame (PF) contains parameter frames and algorithm frames in its data field and the algorithm protocol frame contains several parameter frames as shown in 6.

1) PF: it consists items of MID, visual frame length (VFL), data from parameter frames ($PFDATA$) and data form algorithm frames($AFDATA$).The $PFDATA$ and $AFDATA$ contains several parameter frames and control frames, respectively. The MIDs both in PT and PF need one-to-one correspondence.
2) Algorithm frame: it is comprised of AID, control frame length (CFL) and $PFDATA$. The AIDs both in algorithm frame and PF need one-to-one correspondence.
3) Parameter frame: it contains PID, $PDATA$. PID is also the address of $PDATA$. The PIDs both in parameter frame and PF need one-to-one correspondence.

C. Framing of PF

The framing contains three steps: VCA framing, CL framing, AL framing. Transfer data (TD) save the data to be transferred and it is indexed by the PID. In the VS, it represents the parameters obtained form the visual algorithms. In the ES, the data come from the RAM.

***VCA* Framing**: the process is realized as the Algorthm 1. It searches the PT with mid to find the relevant vlt_x. Through it, we can obtain the man and mdn. According to man and mdn, call the Algorithm 3 and Algorithm 2 to gain the $pfdata$ and $afdata$. Then calculate the length (vfl) to finish the VCA framing.

***CL* Framing**: Algorthm 2 illustrates the process. It seeks the PT to gain the alt_y which contains the apn. According to apn, call the Algorithm 3 to obtain the $pfdata$ and then calculate the length (cfl) to finish the AL framing.

***P* 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, TD
Output: PF
SearchPT (mid, vpt_x);
$PF.mid \leftarrow mid$;
for $i = 0;i < vpt_x.mdn;i++$ **do**
  ALFraming ($vpt_x.cps$[i], TD,$pfdata$);
end
for $i = 0;i < vpt_x.man;i++$ **do**
  CLFraming ($vpt_x.mas$[i], TD,$afdata$);
end
$PF.pfdata \leftarrow pfdata$;
$PF.afdata \leftarrow afdata$;
$PF.vfl \leftarrow (vpt_x.mdn + vpt_x.man) \times 4 + 8$;

Algorithm 2: $CLFraming$

Input: aid, TD, $afdata$
Output: $afdata$
SearchPT (aid, apt_y);
Create clf;
$af.aid \leftarrow aid$;
for $i = 0;i < apt_y.apn;i++$ **do**
  ALFraming($apt_y.aps$[i], TD, $pfdata$);
end
$af.cfl \leftarrow apt_y.apn \times 4 + 8$;
$af.pfdata \leftarrow pfdata$;
$afdata \leftarrow afdata + clf$;

D. Deframing of PF

The process of deframing of PF has three parts: VCA deframing, CL deframing and AL deframing. Received Data (RD) are used to save the deframing parameters. the $RD1$, $RD2$ and $RD3$ are different temporarily storing data area. In the VS, $RD3$ are the data fed back to visual algorithms. In the ES, the $RD1$, $RD2$ and $RD3$ donate LCD, MSD and AD, respectively.

***VCA* deframing**: as illustrated in Algorithm 4, obtain the mid and pfl form the start four and following four bytes data of PDF, respectively. Search the PT to gain the relevant vpt_x. Send the $pfl - 8$ bytes start form the eighth byte of $PDF + 8$ to the address mda of vlt_x.

***CL* deframing**: according to mid, search PT to find the relevant vpt_x. Loop deal with the mda times of the parameter frame. In each loop, the data are sent to $RD3$ with the relevant pid. Loop deal with the man times of the algorithm frame. In each loop, use the aid to find apt_y and then send the $afdata$ to correlative address of $RD2$. Algorithm 5 is described the process.

***AL* deframing**: use aid to get its apt_y. Loop cope with the algorithm frame apn times. Send the parameters to $RD3$ with the correlative pid. The process is shown in Algorithm 6.

Algorithm 3: $ALFraming$

Input: pid, TD,$pfdata$
Output: $pfdata$
SearchPT (pid, ppt_z);
Create pf;
$pf.pid \leftarrow pid$;
$pf.pdata \leftarrow TD[pid] \times ppt_z.var$;
$pfdata \leftarrow pfdata + alf$;

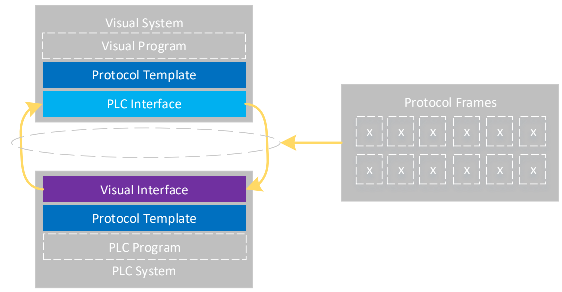

Fig. 7. PF interaction between PLC interface and visual interface.

Algorithm 4: $VLDeframing$

Input: PDF
$mid \leftarrow$ four bytes start form PDF;
searchPT (mid, vpt_x);
$pfl \leftarrow$ four bytes start form $PDF + 4$;
$RD1[vpt_x.mda] \leftarrow pfl - 8$ bytes start form $PDF + 8$;
$PDF \leftarrow PDF + pfl$;

Algorithm 5: $CLDeframing$

Input: mid
searchPT(mid, vpt_x);
$mda \leftarrow vpt_x.mda$;
for $i = 0;i < vpt_x.mdn;i++$ **do**
  $RD3[RD1[mda]] \leftarrow 4$ bytes start form $mda + 4$;
  $mda \leftarrow mda + 8$;
end
for $i = 0;i < vpt_x.man;i++$ **do**
  searchPT($vpt_x.mas[i]$, apt_y);
  $RD2[apt_y.apa] \leftarrow apt_y.vfl - 8$ bytes start form $mda + 8$;
  $mda \leftarrow mda + apt_y.vfl$;
end

IV. System Operation Mechanism

A. Implementation of Flexible Program and Execution

The FL contains two parts: PLC interface and visual interface, shown in Fig. 7. They interact with each other using the agreed communication protocol which is defined as CID in HPT of PT. The execution from PLC interface to visual interface is comprised of five steps.

Algorithm 6: $ALDeframing$

Input: aid
searchPT(aid, apt_y);
$ada \leftarrow apt_y.ada$;
for $i = 0; i < apt_y.apn; i + +$ **do**
  $RD3[RD2[ada]] \leftarrow$ 4 bytes start form $ada + 4$;
  $ada \leftarrow ada + 8$;
end

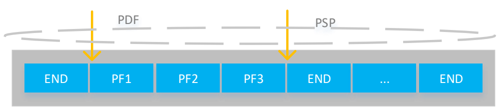

Fig. 8. PDF and PSP. PDF points to the address of deframing PF and PSP points to the address of saving PF. The PDF will point to the next address of PF if a PF is deframed. After saved the PF, the PSP will point to the next new address according the vfl of PF.

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 data required to be transfered (TD) in which the parameters could be indexed by the PID.

Step 2: frame TD into PF with the Algorithm 1, 2 and 3. Here, the TD is an array in VS.

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

Step 4: visual interface receives PFs from VS according the CID in HPT.

Step 5: visual interface saves them in the ES. There are two pointers: PDF and PSP, shown in Fig. 8. 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.

The execution from visual interface to PLC interface consists of the following steps. **Step 1**: visual interface frames TD into PF with the Algorithm 1, 2 and 3. Here, the TD is in RAM of $ePLC$.

Step 2: visual interface transfers the PFs to VS with the relevant communication protocol in CID.

Step 3: PLC interface deframes the PFs with the Algorithm 4, 5 and 6.

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 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 the pointer $PDF! = PSP$, 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 3.

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: if $\exists\ \forall\ adf_i \in ADF$ is 1, execute the $CLDP$ to call Algorithm 5, go to step 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: if $\exists\ \forall\ adf_i \in ADF$ is 1, $ALDP$ is executed to call Algorithm 5, go to step 3.

Step 3: execute the motion algorithm.

Step 4: the $\mathcal{P}_{stm}$ is used 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.

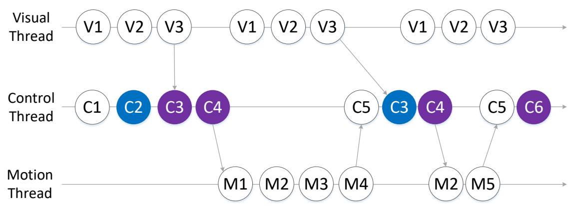

Fig. 9. Process of thread execution.

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: Winding Machine with Visual System

As shown in Fig. 10, (a) is the visual system, whose CUP is ARM9 based S3C2440 and which uses the CANIF interface to communicate with OV9650 CMOS Camera; (b) is the $ePLC$, CASS-PLCA149B; (c) is the winding machine with visual system which contains two axes: the U-axis and Q-axis; (d) is shown the Winding effect, the angle of the wire, Q-axis and U-axis. In winding process, the tension of the copper wire will change irregularly, especially for the thick ones. However, we use the same speed ratio of U-axis and Q-axis which always leads to irregularity of each layer of the coil. Hence, we can use the visual system to decrease or increase the speed of Q-axis to control the angle of wire.

Fig. 11 is the structure of the winding machine system. The ePLC has two chips: a master chip and a slave chip. It uses the RS232 to receive PFs and could control six servo systems at most. The winding machine has two axes: the U-axis and Q-axis. The VS use the CMOS camera to gain winding images

TABLE I

PT OF WINDING MACHINE

cid_1	tda_1	xxx	xxx	xxx	xxx	xxx
0x01	0x00					
mid_1	man_1	mdn_1	mda_1	msf_1	mdf_1	mas_1
0x01	1	0	0X200	0X400	0X600	0x01
aid_1	apn_1	af_1	ada_1	$aps1_1$	xxx	xxx
0x01	0X1	0X14A	0x1F4000	0X0		
pid_1	vid_1	var_1	xxx	xxx	xxx	xxx
0X0	0X1	1000				

TABLE II

PF OF WINDING MACHINE

mid_1	pfl_1	aid_1	cfl_1	pid_1	$pdata_1$
0x01	0x14	0x01	0xE	0x01	0x400
0x01	0x14	0x01	0xE	0x01	0x200
0x01	0x14	0x01	0xE	0x01	0x000

and transfer the digital information to ARM CUP. The ARM CUP will extract parameters from the digital information, frame PFs and transfer PFs to $ePLC$ continuously.

1) Design of PT: In [18], we propose the customized winding machine language which only contains 13 instructions. Here, we can use the second and third instructions to control U-axis and Q-axis, respectively.The process of winding is mainly to control the speed ratio of U-axis and Q-axis, hence we use the VS to influence the speed of Q-axis only. The PT of winding machine is illustrated in Table I in which the pid_1 is the speed of Q-axis and the 1000 of var_1 is used to multiply by the θ.

2) Process of PF: Table II is shown the interacted PF between the VS and ES. At first, the VS detects that the θ is 2 degrees. Then, it sends the PF to inform the ES to increase the Q-axis speed of 2000 pulses per millisecond (p/ms). When the θ decreases to 1 degree, the increment of the Q-axis speed is changed to 1000 p/ms. Since the θ has recovered to 0, the ES is informed to recover the initial speed if the Q-axis.

B. Case 2: Binocular Catching Robot

As shown in Fig. 12, 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. $ePLC$ uses the TI F28M35 chip with two cores: ARM Cortex M3 and TI C28x, which contains a shared RAM. The servo system is adopted ASDA-B2.

1) Design of PT: The $ePLC$ is designed to control six axes parallel. We adopt the same algorithm of Q-axis and U-axis in case one while we name three axes of the Cartesian Robot as X-axis, Y-axis and Z-axis. In this case, we have the similar module with case one however it contains three motion algorithms. pid_1, pid_2 are the position and speed of X-axis; pid_3, pid_4 are the position and speed of Y-axis; pid_5, pid_6

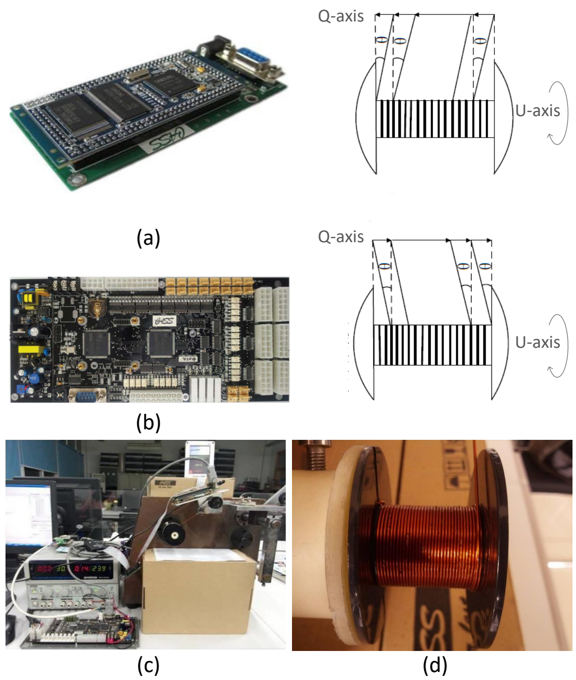

Fig. 10. (a) is the visual system, whose CUP is ARM9 based S3C2440 and which uses the CANIF interface to communicate with OV9650 CMOS Camera; (b) is the $ePLC$, CASS-PLCA149B; (c) is the winding machine with visual system which contains two axes: the U-axis and Q-axis; (d) is shown the Winding effect, the angle of the wire, Q-axis and U-axis.

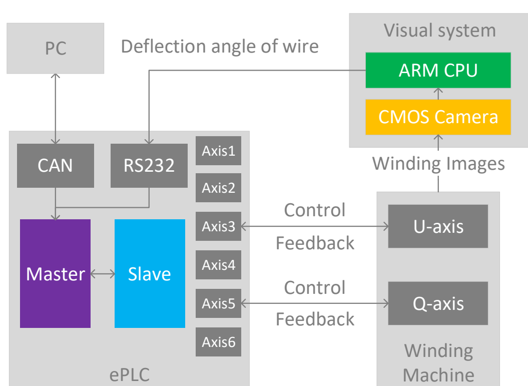

Fig. 11. Structure of the winding machine system. The ePLC has two chips: a master chip and a slave chip. It uses the RS232 to receive PFs and could control six servo systems at most. The winding machine has two axes: the U-axis and Q-axis. The VS use the CMOS camera to gain winding images and transfer the digital information to ARM CUP. The ARM CUP will extract parameters, frame PFs and transfer them to $ePLC$ continuously.

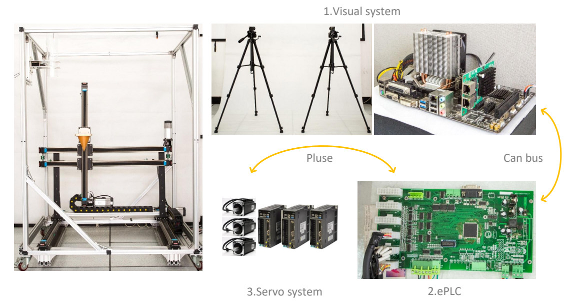

Fig. 12. Binocular Catching Robot consists of visual system, $ePLC$ and servo system.

TABLE III
PT OF BINOCULAR CATCHING ROBOT

cid_1	tda_1	xxx	xxx	xxx	xxx	xxx	xxx	xxx
0x01	0x00							
mid_1	man_1	mdn_1	mda_1	msf_1	mdf_1	mas_1	mas_1	mas_1
0x01	1	0	0X200	0X400	0X600	0x01	0x02	0x03
aid_1	apn_1	af_1	ada_1	$aps1_1$	$aps2_1$	xxx	xxx	xxx
0x01	0X2	0X14A	0x1F4000	0X0	0x4			
aid_2	apn_2	af_2	ada_2	$aps1_2$	$aps2_2$	xxx	xxx	xxx
0x02	0X2	0X14A	0x1F4000	0X8	0xA			
aid_3	apn_3	af_3	ada_3	$aps1_3$	$aps2_3$	xxx	xxx	xxx
0x03	0X2	0X14A	0x1F4000	0XE	0X10			
pid_1	vid_1	var_1	pid_2	vid_2	var_2	pid_3	vid_3	var_3
0X0	0X1	10000	0X4	0X1	10000	0X8	0X1	10000
pid_4	vid_4	var_4	pid_5	vid_5	var_5	pid_6	vid_6	var_6
0XA	0X1	10000	0XE	0X1	10000	0X10	0X1	10000

are the position and speed of Z-axis. The VS use meter and meter per second (m/s) to measure the distance and speed, respectively while in the ES, driving every 1 cm need to output 10000 pluses. Hence, the VAR of position and speed are both 10000.

2) Process of PF: The PFs of a time catching the ball is shown in Table IV and the trajectory of the ball is shown in Fig. 13 which is only shown the position of X-axis and Y-axis. The VS sends the PF to ES to adjust the destination and speed of every axis.

C. Result

Through the analysis of two cases, the proposed VCA based flexible structure provides a generic method to address the visual servo control problem in $ePLC$.

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.

TABLE IV
PF OF BINOCULAR CATCHING ROBOT

mid_1	pfl_1	aid_1	cfl_1	pid_1	$pdata_1$	pid_2	$pdata_2$	aid_2	cfl_2	pid_3	$pdata_3$	pid_4	$pdata_4$	aid_3	cfl_3	pid_5	$pdata_5$	pid_6	$pdata_6$
0x01	0x14	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400
0x01	0x14	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400
0x01	0x14	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400
0x01	0x14	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400
0x01	0x14	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400
0x01	0x14	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400	0x01	0xE	0x01	0x400	0x01	0x400

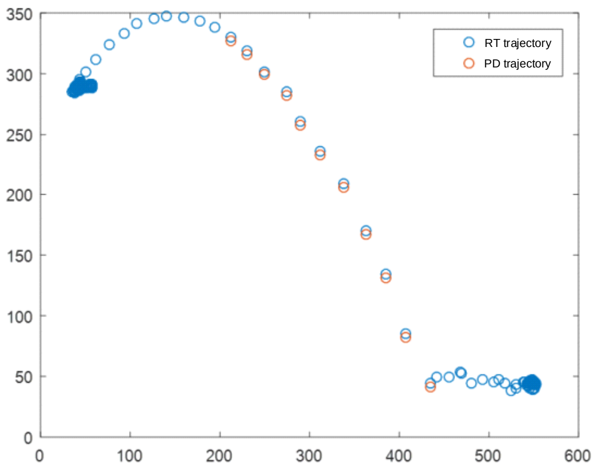

Fig. 13. Trajectory of the ball which is only shown the position of X-axis and Y-axis.

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.