Advanced Controller for Robot Manipulator

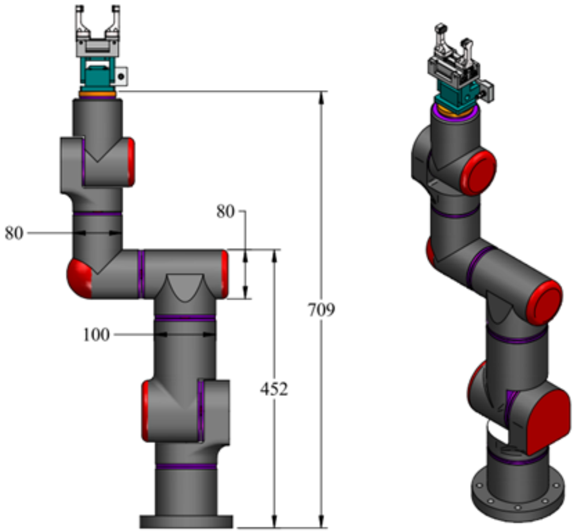
Jwu-Sheng Hu
Professor, Dept. of Electrical and Computer Engineering, NCTU, Taiwan
Technical Director, MSL, ITRI, Taiwan
Director, Intelligent Robotics Division, MSL, ITRI, Taiwan

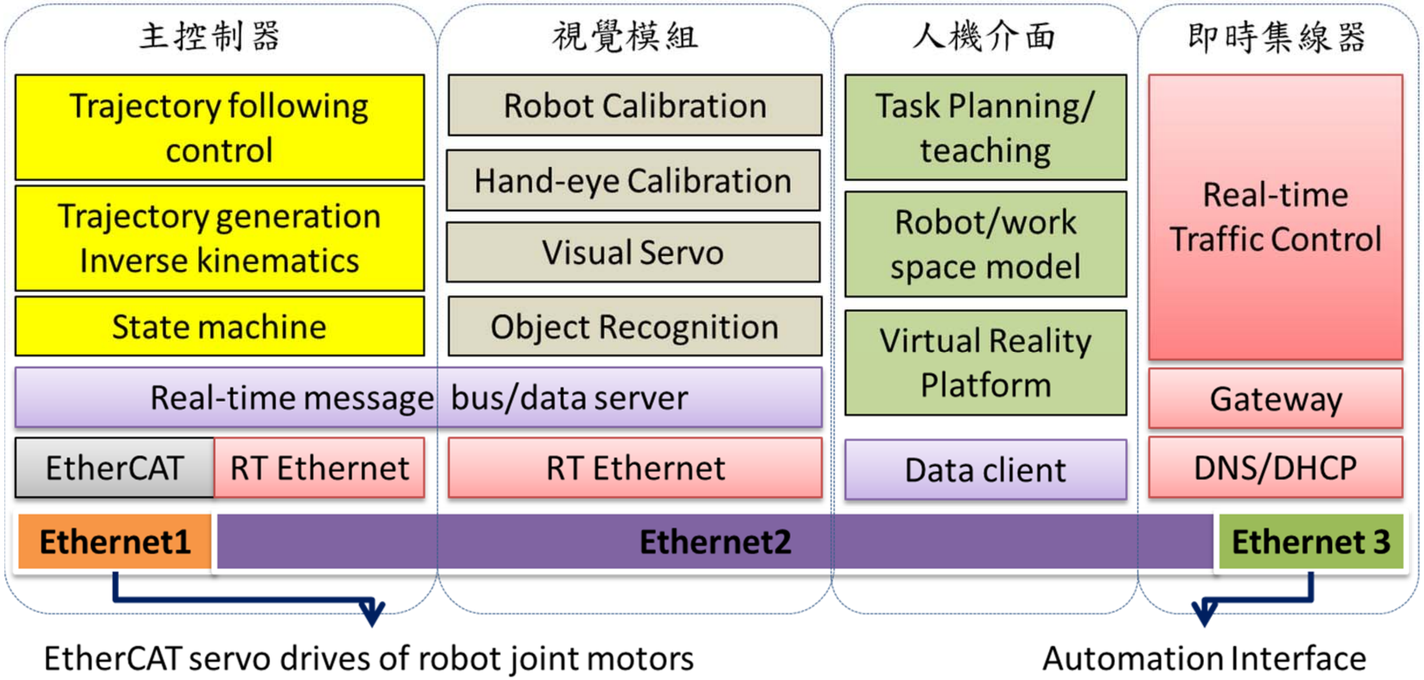
Outline

- Advanced Robot Arm
- Real-time OS
- EtherCat Protocol
- Robotic Arm Dynamic Control Technology
- 3D Objects Pose Estimation
- Intelligent Graphic User Interface System
- Advanced Hand-Eye-Workspace Calibration

Advanced Robot Arm

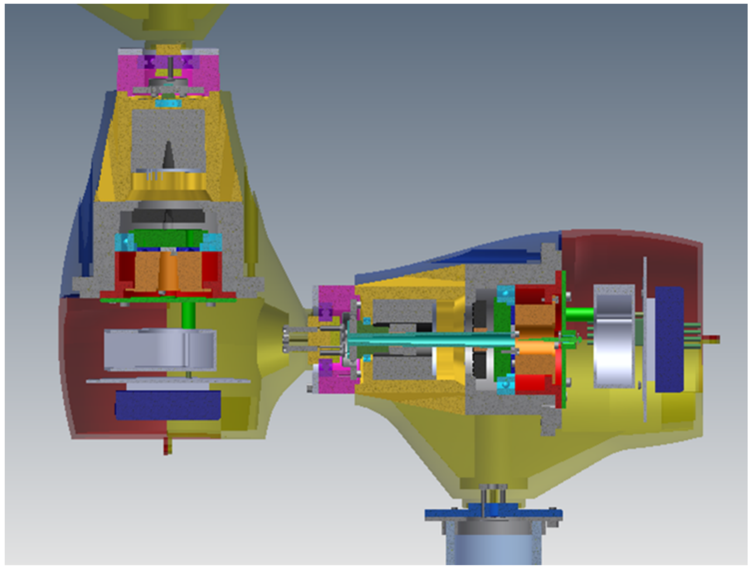
- Designed and assembled by Industrial Technology Research Institute.
- Small and compact size
- Seven degree of freedom

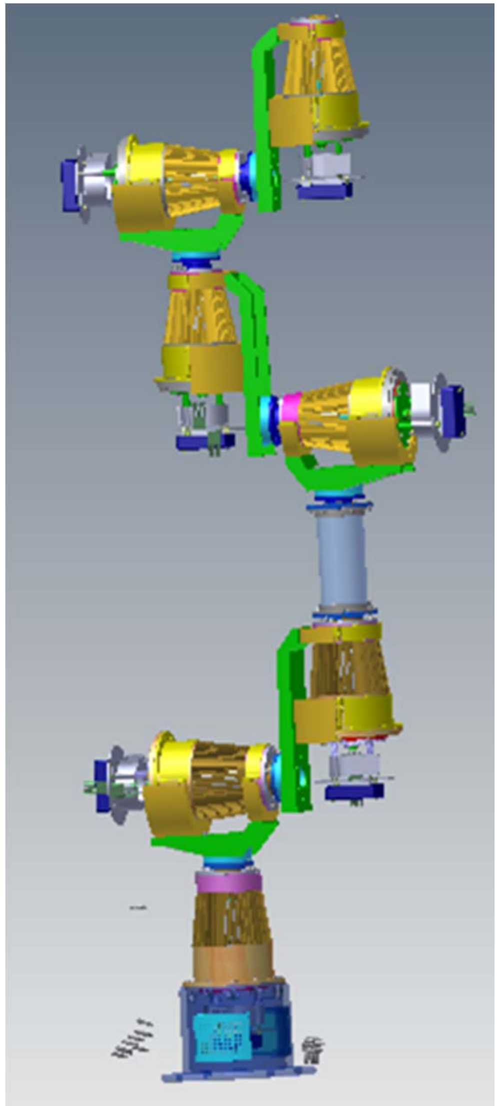




Advanced Robot Arm

- EtherCAT communication
- No control box, and the motor driver is embedded in each link.
- Hollow shaft design for routing





Outline

- Advanced Robot Arm
- **<u>Real-time OS</u>**
- EtherCat Protocol
- Robotic Arm Dynamic Control Technology
- 3D Objects Pose Estimation
- Intelligent Graphic User Interface System
- Advanced Hand-Eye-Workspace Calibration

Real-time OS
Introduction

- Growing penetration of Linux
- OSS EtherCAT has been developed
- Control applications in ***user space*** is possible
  - IgH version 1.5.0 or above

- However,
  - IgH can't guarantee RT capability
  - Even RTAI is adopted

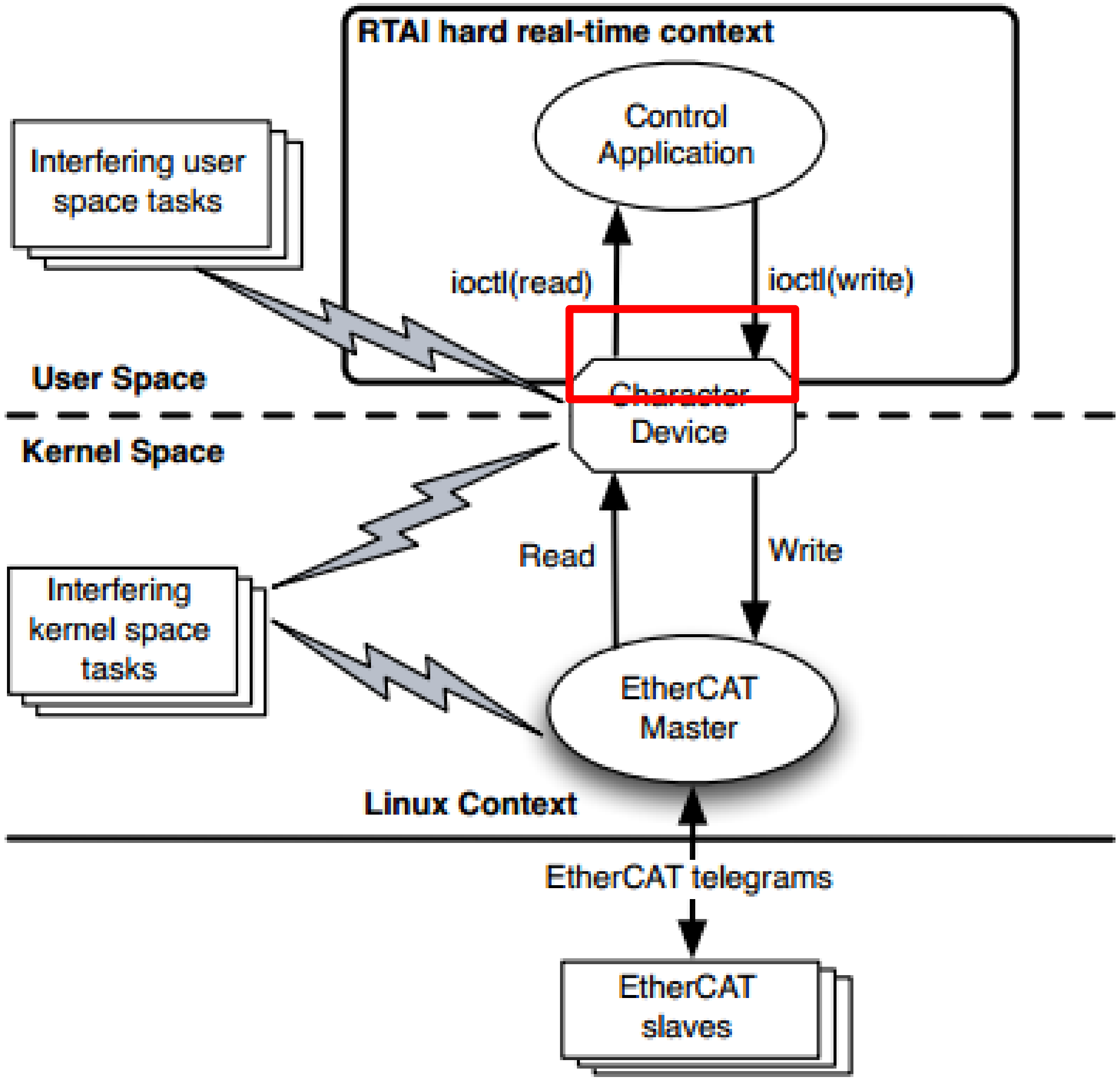
Real-time OS
IgH EtherCAT

- IgH EtherCAT is composed of
  - master module
    - Provide functions and data structures to allow applications accessing the master functionalities via API
  - control application
    - Interface with the master modules by means of API
    - For the cyclic exchange of process data with EC slaves
    - Can be implemented with IgH EC libraries

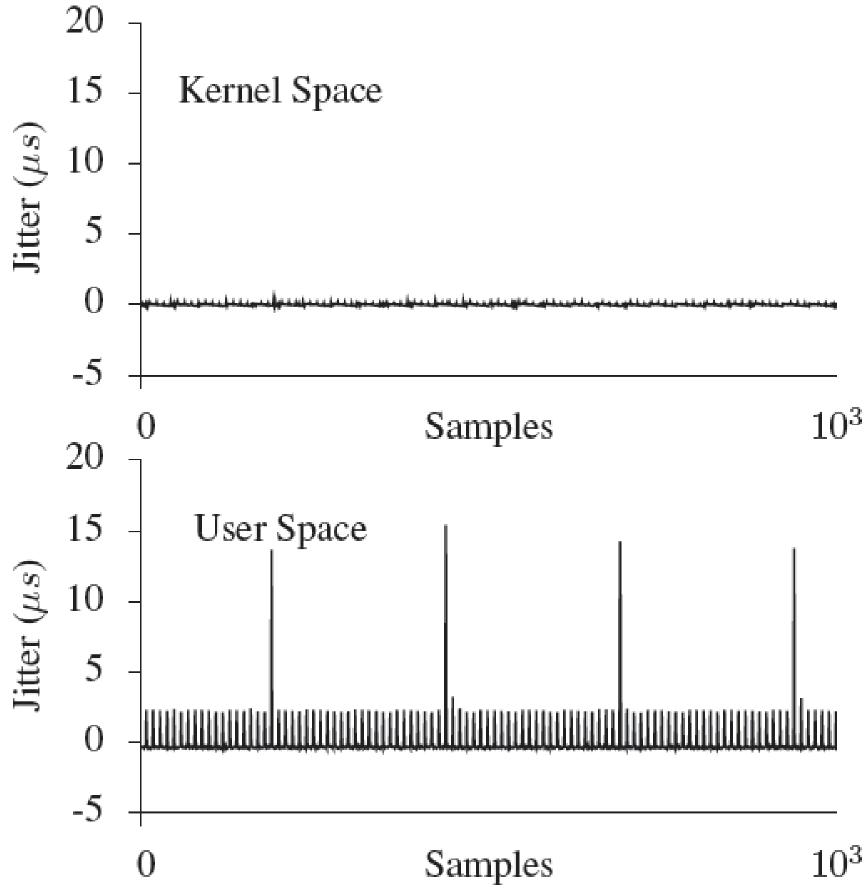
Real-time OS
Cereia’s Work

- “**A user space EtherCAT master architecture for hard real-time control systems** ”
  - 2012 IEEE ETFA Conference
- Implementation
  - Linux kernel 2.6.32.11 + RTAI 3.8.1
  - IgH 1.5.0
- Improve the user space control application
  - Modify the control application to be RTAI tasks
  - Make it to reach real-time capability

Real-time OS
Drive via Character Device



Real-time OS
Jitter: Kernel vs. User Space



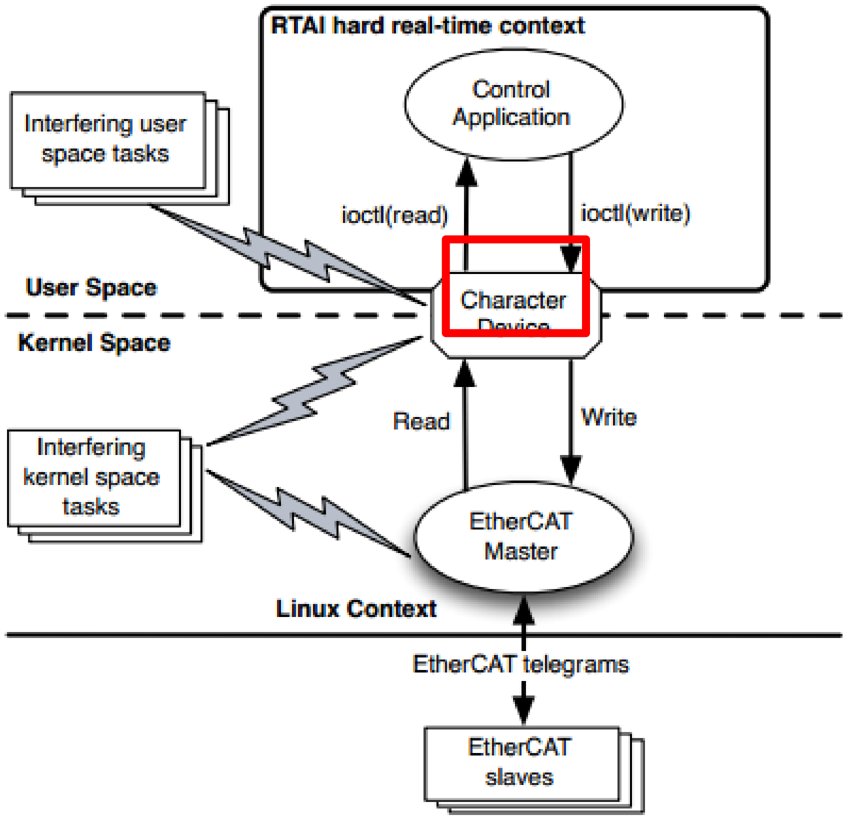
Kernel space timing

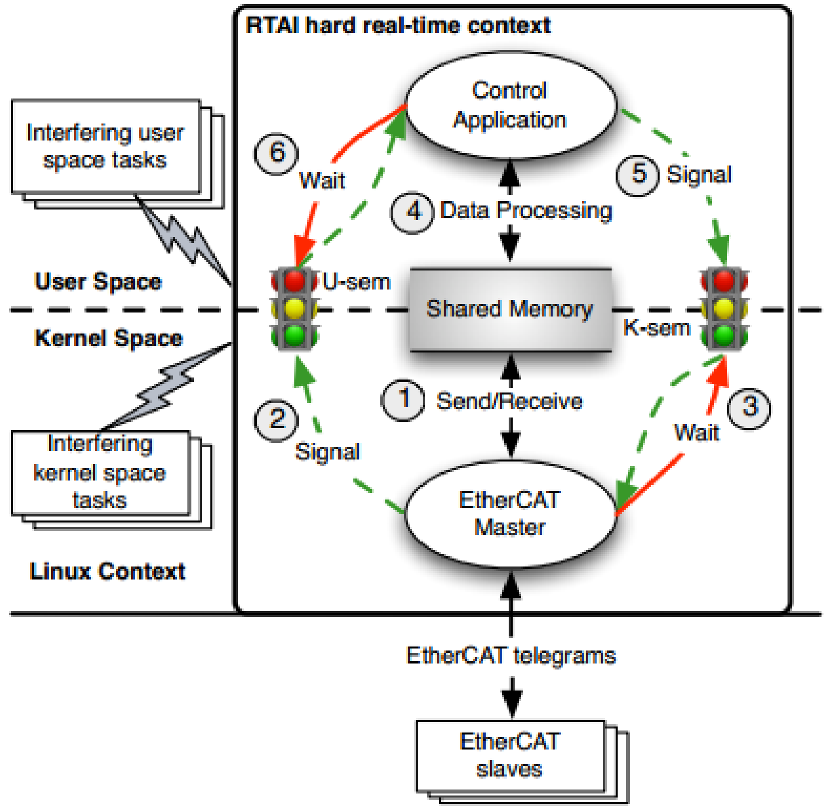
Kernel	*User*
while(1) { **rt_task_wait_period();** write_outputs(); read_inputs(); signal(U-sem); *wait(K-sem);* }	while(1) { wait(U-sem); /*data processing*/ *signal(K-sem);* }

User space timing

Kernel	*User*
while(1) { wait(K-sem); write_outputs(); read_inputs(); signal(U-sem); }	while(1) { **rt_task_wait_period();** signal(K-sem); wait(U-sem); /*data processing*/ }

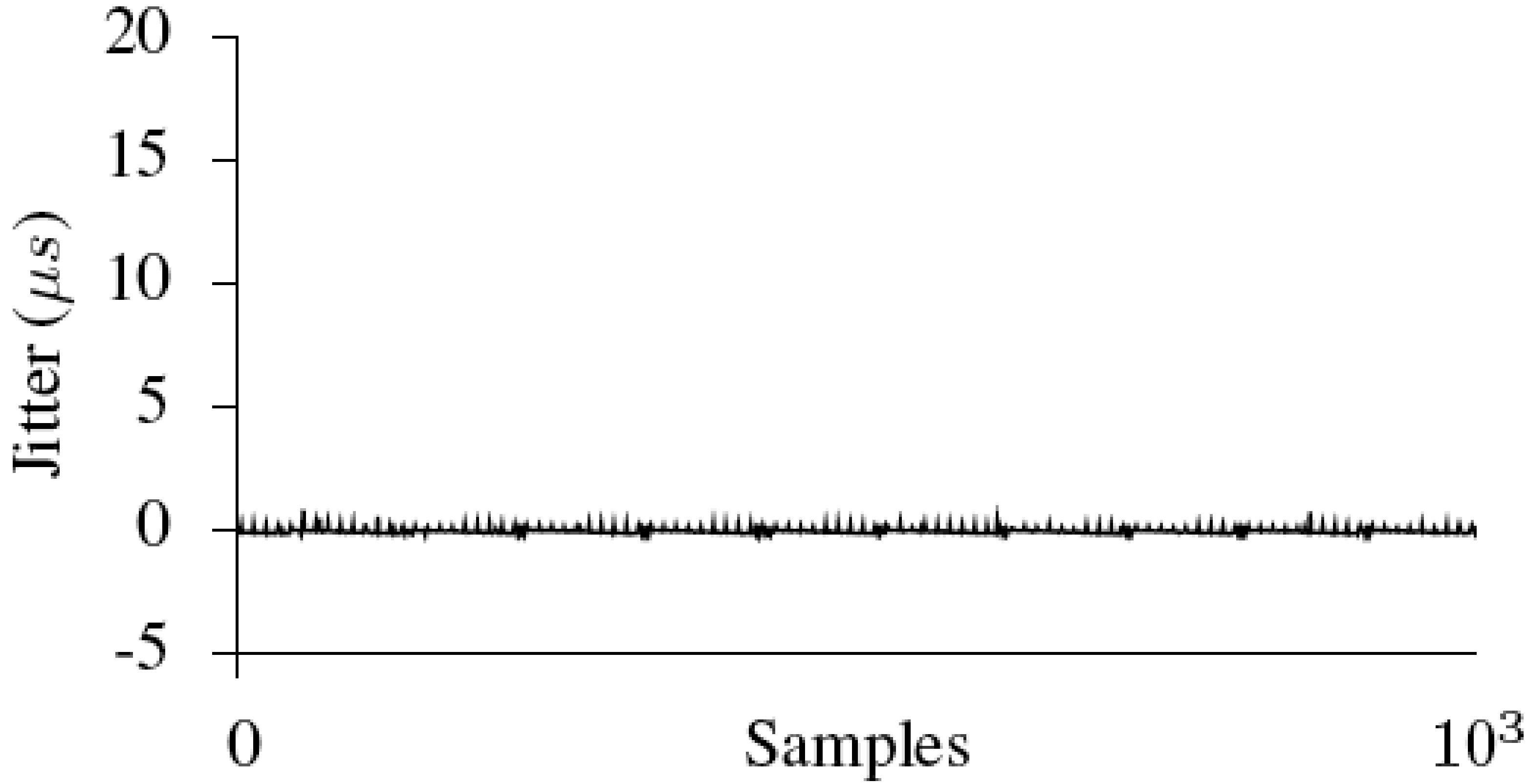
Real-time OS
Character Device vs. Shared Memory



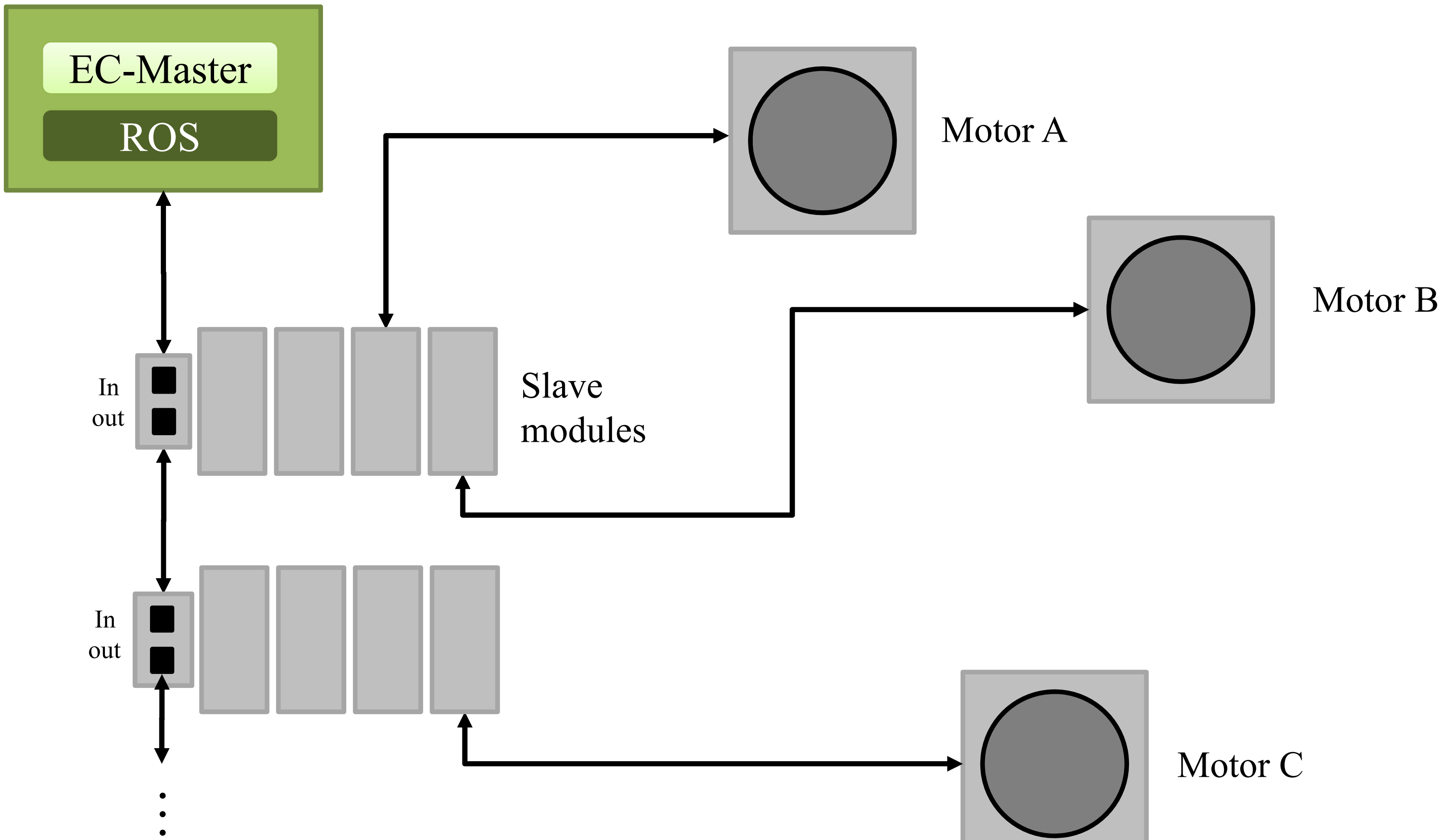


Real-time OS

Jitter: User Space/Cereia's Design



Real-time OS
EC-based Robot OS



Real-time OS
Conclusion

- Control application can be implemented at user space based on RTAI
  - Real time guaranteed

- Cereia's Design can be used to realize a Robot OS
  - For real-time control

Outline

- Advanced Robot Arm
- Real-time OS
- <u>**EtherCat Protocol**</u>
- Robotic Arm Dynamic Control Technology
- 3D Objects Pose Estimation
- Intelligent Graphic User Interface System
- Advanced Hand-Eye-Workspace Calibration

EtherCat Protocol
Outline

- Industrial Communication Technique Comparison
- COE Communication Structure
  - Cyclic Data Transmission
  - Acyclic Data Transmission
- EtherCAT Master Processing Procedure
- Device Description File
  - Device Description File Analysis
  - Package Analysis
  - Establish Motion Control Parameters
- EtherCAT Master User Interface Demo

EtherCat Protocol
Industrial Communication Technique Comparison

Real-time Level

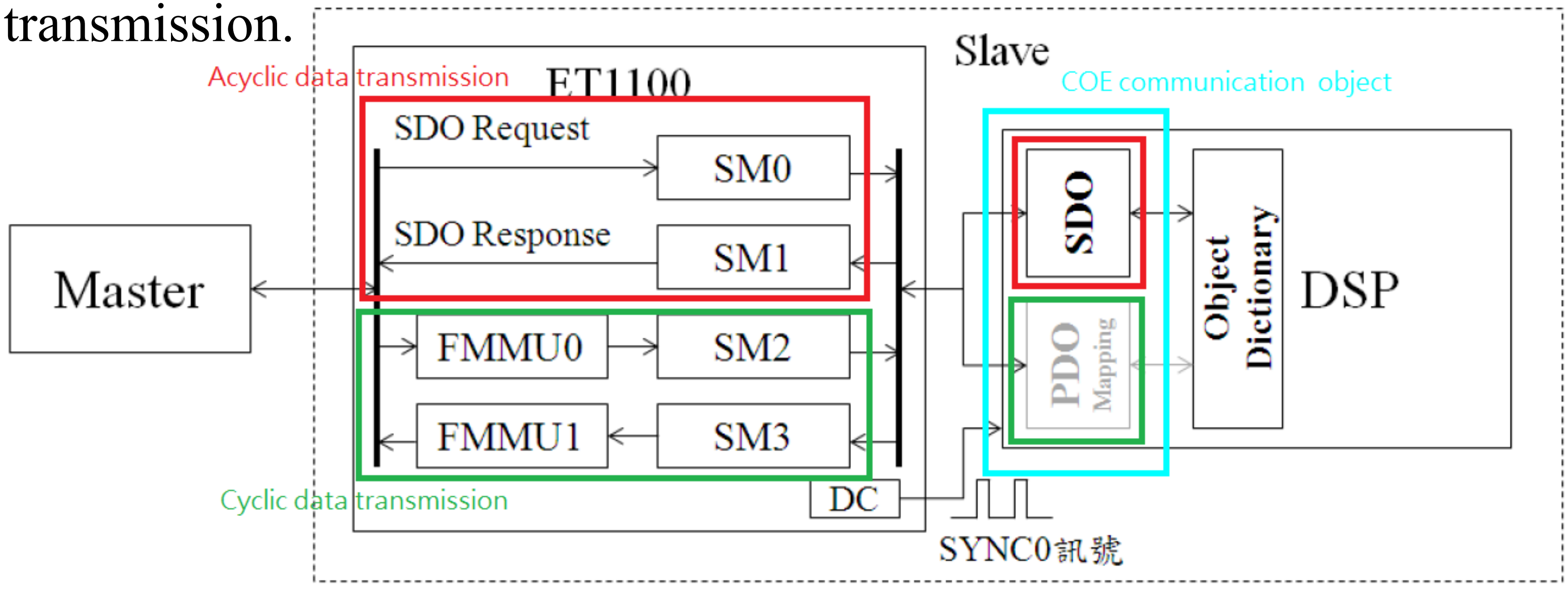
Protocols	Real-time class	Throughput	Max devices	Synchronisation
CAN	2	10 kb/s–1 Mb/s, 31.25 kb/s, 1 Mb/s, 2.5 Mb/s (5 Mb/s optical fiber)	32	Y
PROFIBUS PROFIBUS-DP PROFIBUS-PA	2	 9.6 kb/s-12 Mb/s 31.25 kb/s	 126 32/seg.	N
MODBUS	1	10,100 Mb/s, 1 Gb/s	247	N
Ethernet / IP	1	10,100 Mb/s, 1 Gb/s	Almost Unlimited	Y
PROFINET	三種皆有	100 Mb/s	Almost Unlimited	Y
PowerLink	3	100 Mb/s	240	Y
SERCOS	3	100 Mb/s	254	Y
EtherCAT	3	100 Mb/s	65535	Y

EtherCat Protocol
COE Communication Structure

COE(CANopen Over EtherCAT) includes the following communication mode:

(1) Mailbox mode – Use SDO (Service Data Object) for acyclic transmission.

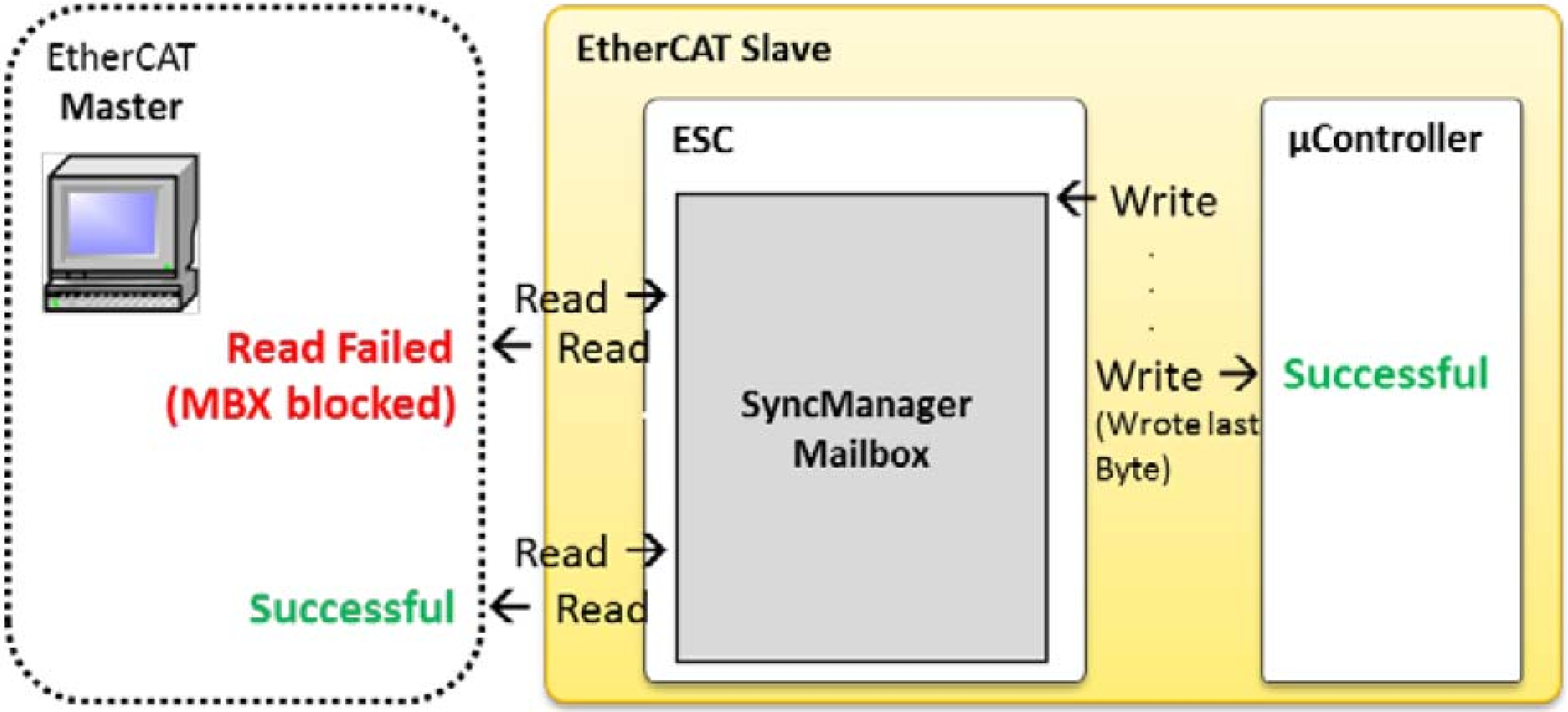
(2) Buffer mode – Use PDO (Process Data Object) for cyclic transmission.



EtherCat Protocol
COE Communication Structure

- Acyclic Data Transmission(Mailbox mode)

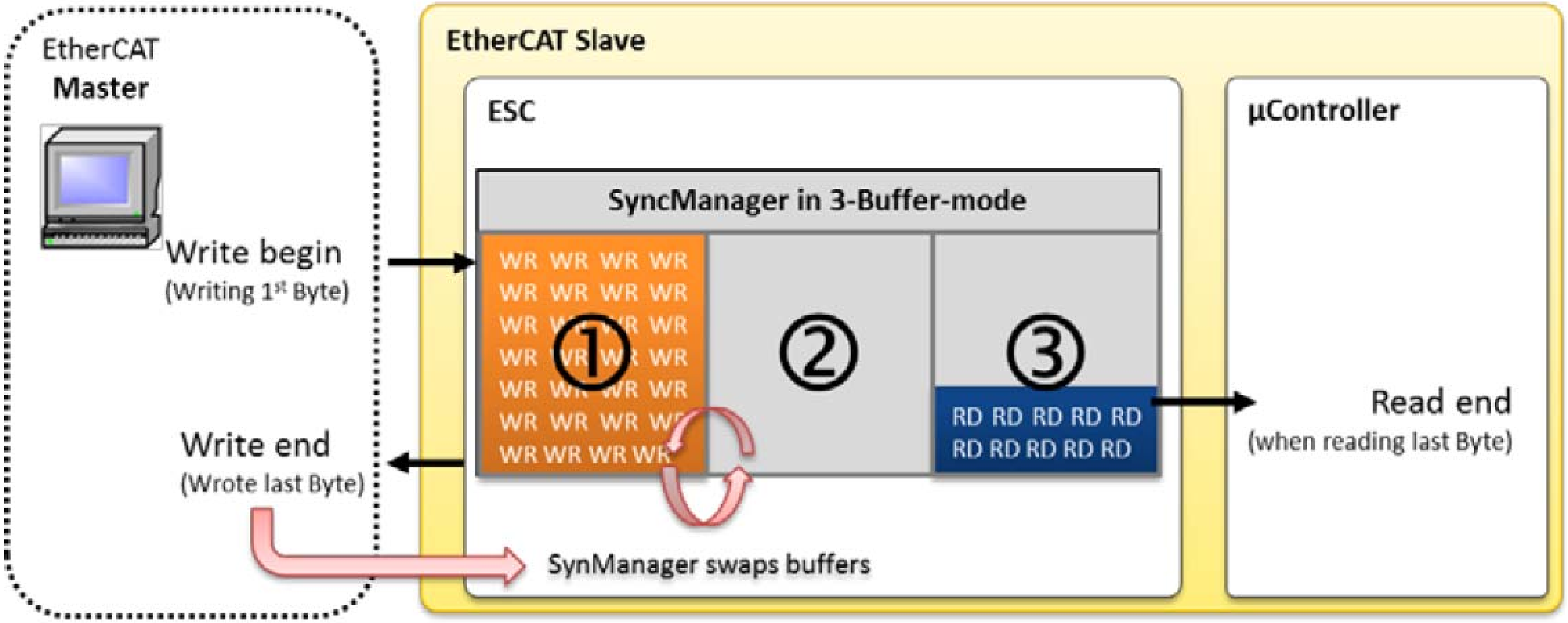
When the sender writes the buffer, the buffer is locked for writing until the receiver has read it out.



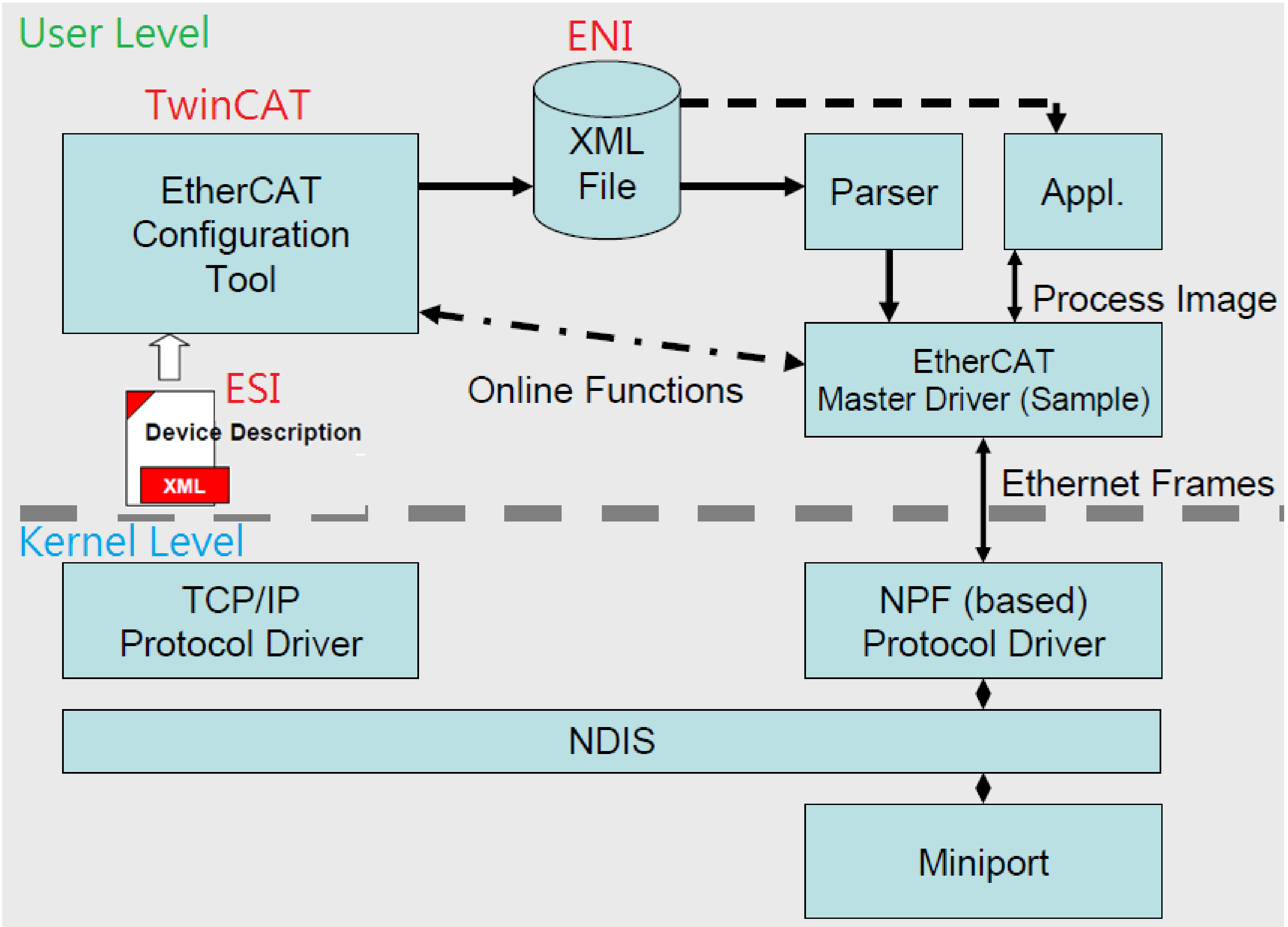
EtherCat Protocol
COE Communication Structure

- Cyclic Data Transmission(Buffer mode)

The sender can always update the content of the buffer. If the buffer is written faster than it is read out by the receiver, old data is dropped. Thus, the receiver always gets the latest consistent buffer content which was written by the sender.

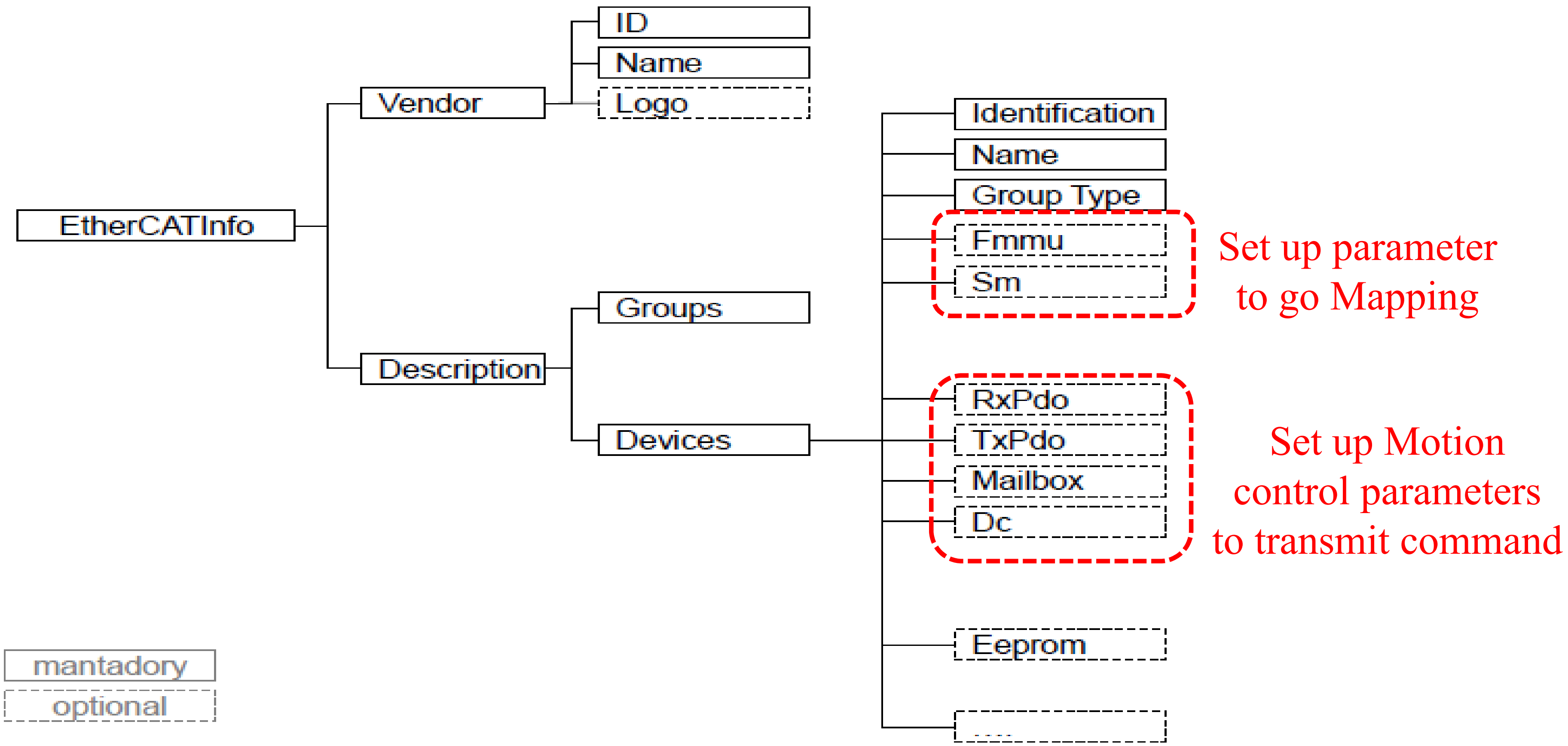


EtherCat Protocol
EtherCAT Master Processing Procedure



EtherCat Protocol
Device Description File

- EtherCAT slave device describe



EtherCat Protocol
Device Description File

- Device description file analysis

	DefaultSize	StartAddress	ControlByte	Enable	MinSize	MaxSize	Text
1	128	#x1000	#x26	1	34	192	MBoxOut
2	128	#x1400	#x22	1	34	192	MBoxIn
3	2	#x1800	#x64	1			Outputs
4	6	#x1C00	#x20	1			Inputs

Mapping physical address

RxPdo

Fixed	1
Sm	2
Index DependOnSlot=true	
Name	Outputs
Entry (2)	

	Index	SubIndex	BitLen	Name	Comment	DataType
1	Index DependOnSlot=true	0	16	Control Word	object 0x6040:0	UINT
2	Index DependOnSlot=true	0	32	TargetPosition	object 0x607A:0	DINT

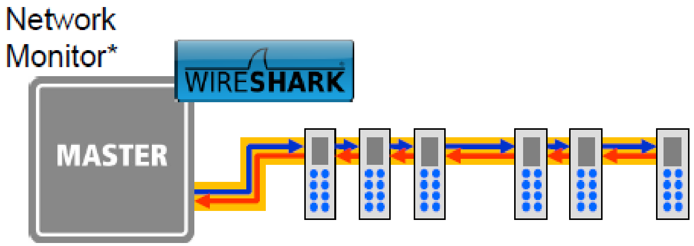
TxPdo

Fixed	1
Sm	3
Index DependOnSlot=true	
Name	Inputs
Entry (2)	

	Index	SubIndex	BitLen	Name	Comment	DataType
1	Index DependOn...	0	16	Status Word	object 0x6041:0	UINT
2	Index DependOn...	0	32	ActualPosition	object 0x6064:0	DINT

EtherCat Protocol
Device Description File

- Package analysis




```
EtherCAT datagram(s): 'FPWR': Len: 16, Adp 0x3e9, Ado 0x600, Wc 0
  EtherCAT datagram: Cmd: 'FPWR' (5), Len: 16, Adp 0x3e9, Ado 0x600, Cnt 0
    Header
    Fieldbus Memory Management Units (FMMU)
      Log Start: 0x01000000
      Log Length: 0x0002
      Log StartBit: 0x00
      Log EndBit: 0x07
      Phys Start: 0x1800
      Phys StartBit: 0x00
      FMMU Type: 0x02
        .... ...0 = Type: Read ignore
        .... ..1. = Type: Write in use
      FMMU Active: 0x01
        .... ...1 = Active: Enabled
    Working Cnt: 0
```

Master Logical Address

Slave Physical Address

No.	Time	Source	Destination	Protocol
21317	21.192077000	Beckhoff_10:1c:9c	Beckhoff_01:00:00	ECAT
21318	21.192086000	Beckhoff_01:00:00	MS-NLB-PhysServer-01_05:10:1c:9c	ECAT
21319	21.194077000	Beckhoff_10:1c:9c	Beckhoff_01:00:00	ECAT
21320	21.194085000	Beckhoff_01:00:00	MS-NLB-PhysServer-01_05:10:1c:9c	ECAT
21321	21.196077000	Beckhoff_10:1c:9c	Beckhoff_01:00:00	ECAT
21322	21.196086000	Beckhoff_01:00:00	MS-NLB-PhysServer-01_05:10:1c:9c	ECAT
21323	21.198075000	Beckhoff_10:1c:9c	Beckhoff_01:00:00	ECAT
21324	21.198083000	Beckhoff_01:00:00	MS-NLB-PhysServer-01_05:10:1c:9c	ECAT
21325	21.200077000	Beckhoff_10:1c:9c	Beckhoff_01:00:00	ECAT
21326	21.200085000	Beckhoff_01:00:00	MS-NLB-PhysServer-01_05:10:1c:9c	ECAT
21327	21.202077000	Beckhoff_10:1c:9c	Beckhoff_01:00:00	ECAT

9 µs

2.002ms

EtherCat Protocol
Device Description File

- Establish motion control parameters
  - When controller parameter is used, it musts set up.
  - Using handshake mechanism makes sure whether

	DefaultSize	StartAddress	ControlByte	Enable	MinSize	MaxSize	Text
1	128	#x1000	#x26	1	34	192	MBoxOut
2	128	#x1400	#x22	1	34	192	MBoxIn
3	2	#x1800	#x64	1			Outputs
4	6	#x1C00	#x20	1			Inputs

```
⊟ EtherCAT datagram: Cmd: 'FPRD' (4), Len: 128, Adp 0x3e9, Ado 0x1400, Cnt 1
  ⊟ Header
      Cmd        : 4 (Configured address Physical Read)
      Index: 0xb7
      Slave Addr: 0x03e9
      Offset Addr: 0x1400
    ⊞ Length     : 128 (0x80
      Interrupt: 0x0000
  ⊟ EtherCAT Mailbox Protocol:CoE
    ⊟ Header
        Length: 10
        Address: 0x0000
        Priority: 0
        Type     : CoE (CANopen over EtherCAT) (0x3)
        Counter  : 5
    ⊟ CoE
        Number   : 0
        Type     : SDO Res(3)
      ⊟ SDO Res  : Scs 2
        ⊞ Initiate Upload Response: 0x4f
        Index: 0x6000
        SubIndex: 0x00
        Data: 0x08
    Data: 454c393830305f3441202850494332342 95f56356930315f...
    Working Cnt: 1
```

Slave addr : Slave node ,
offset addr : Physical address

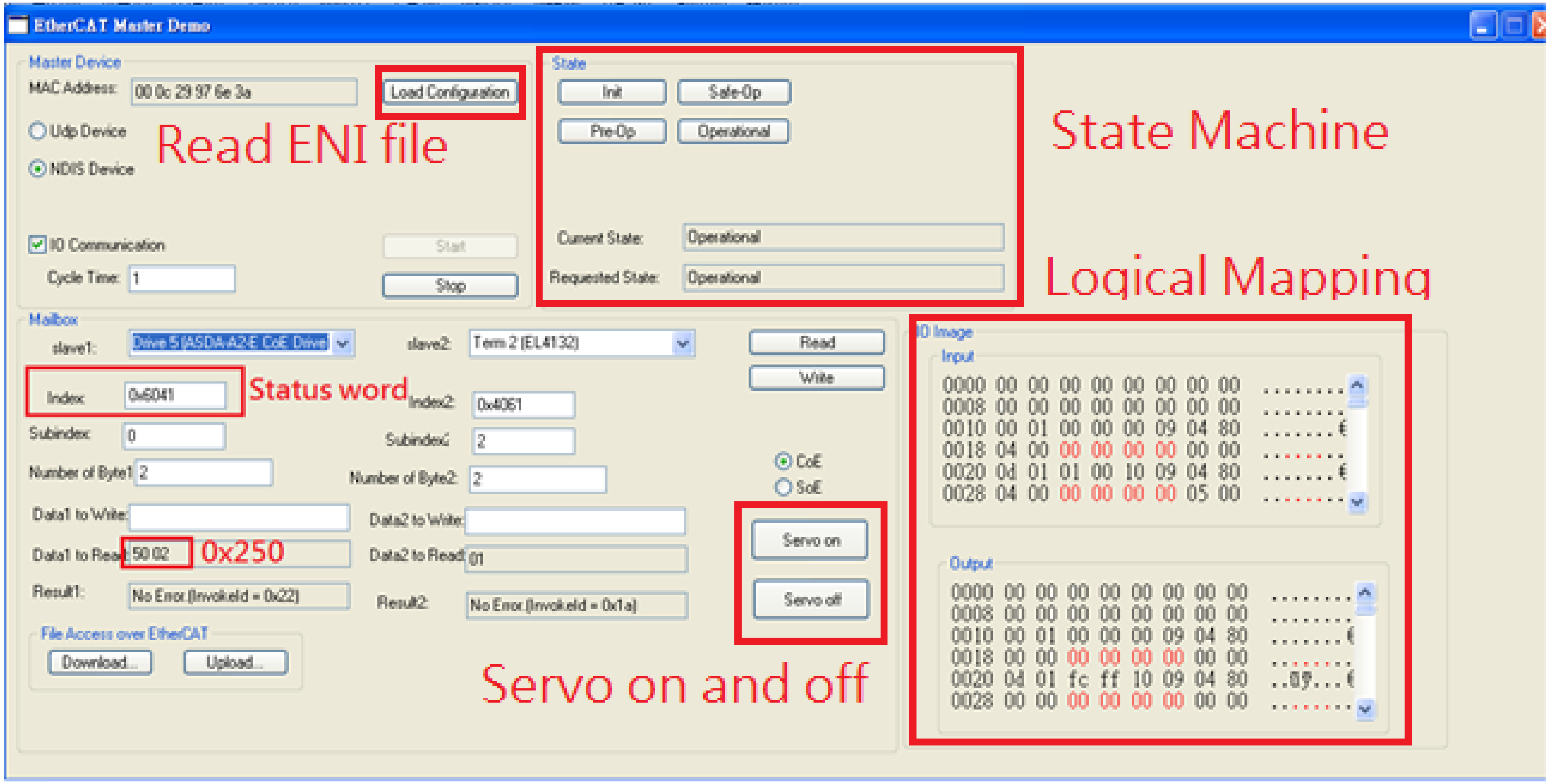
CANopen over EtherCAT

EtherCat Protocol
EtherCAT Master User Interface Demo

- EtherCAT master interface program develops in Window XP and uses Visual Studio 2010 C++ to develop.
- The master UI includes the following feature :
  1) Reading ENI file acquires network topology and slave station information
  2) Sending the initializing packet of ENI file initializes the slave station in different state machine transformation.
  3) Acyclic Data transmission. Ex : Set up parameter、Write into fixed command...
  4) Write into the acyclic data for two different slaves at the same time

EtherCat Protocol

EtherCAT Master User Interface Demo



Outline

- Advanced Robot Arm
- Real-time OS
- EtherCat Protocol
- **Robotic Arm Dynamic Control Technology**
- 3D Objects Pose Estimation
- Intelligent Graphic User Interface System
- Advanced Hand-Eye-Workspace Calibration

Robotic Arm Dynamic Control Technology

Introduction

- We develop high-precision, high-response-speed multi-axis control laws for 7-dof robotic arm. The control laws are robust, not affected by the change of the load arm. We regulate the control parameters with systematic rule.

Robotic Arm Dynamic Control Technology
Kinematics

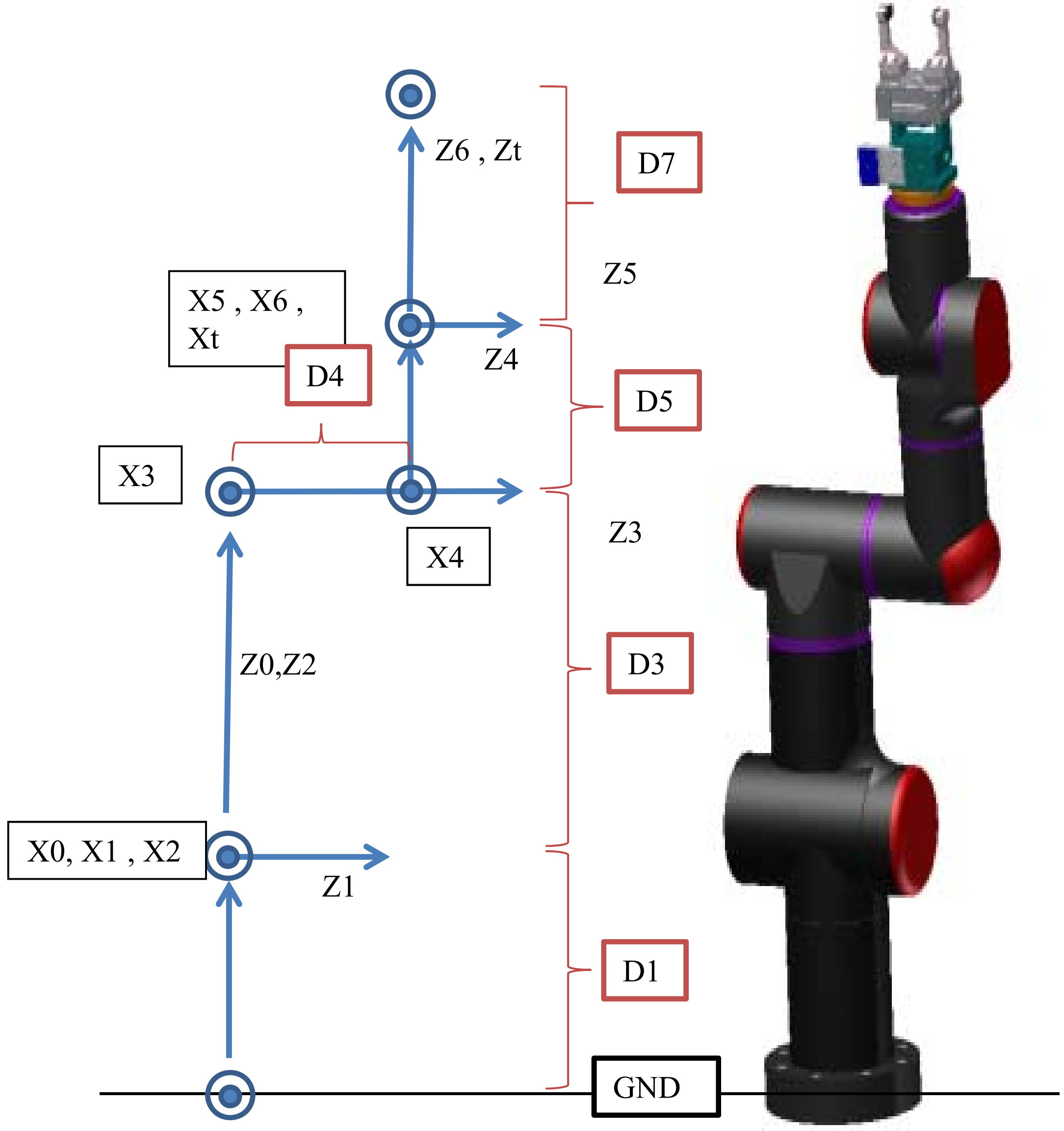
- Forward kinematics:

  We use D-H model to get the end-effector pose corresponding to angle of each axis. We transfer joint-space coordinate into Cartesian coordinate.

DH Parameters

Joint	θ	α (degree)	a	d
1	θ1	-90	0	D1
2	θ2	90	0	0
3	θ3	-90	0	D3
4	θ4	90	0	D4
5	θ5	-90	0	D5
6	θ6	90	0	0
7	θ7	0	0	D7

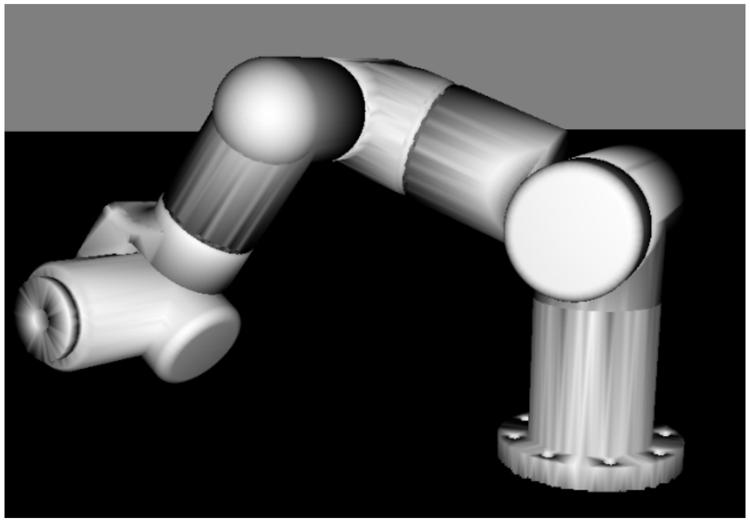
Robotic Arm Dynamic Control Technology
Kinematics

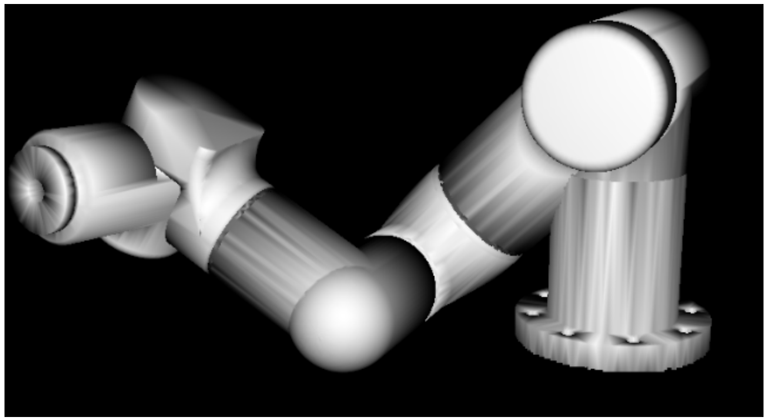


Robotic Arm Dynamic Control Technology
Kinematics

- Inverse kinematics:

  We calculate angle of each axis by the position of end-effector. We transfer Cartesian coordinate into joint-space coordinate. The verification through 3D simulation are shown as follow:





Robotic Arm Dynamic Control Technology

Dynamic Equations

Robotic arm robot arm dynamic equations describe the relationship between position, velocity, acceleration and torque, which can be expressed as the following formula:

$$\tau = \mathrm{M}(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta)$$

where θ is the rotation angle of each axis, $M(\theta)$ is the mass inertia matrix, $C(\theta,\dot{\theta})$ is the centripetal force and the Coriolis force matrix, the above two matrices are 7×7 matrix, $G(\theta)$ is the 7×1 gravity vector, τ is the torque of each axis from the actuator.

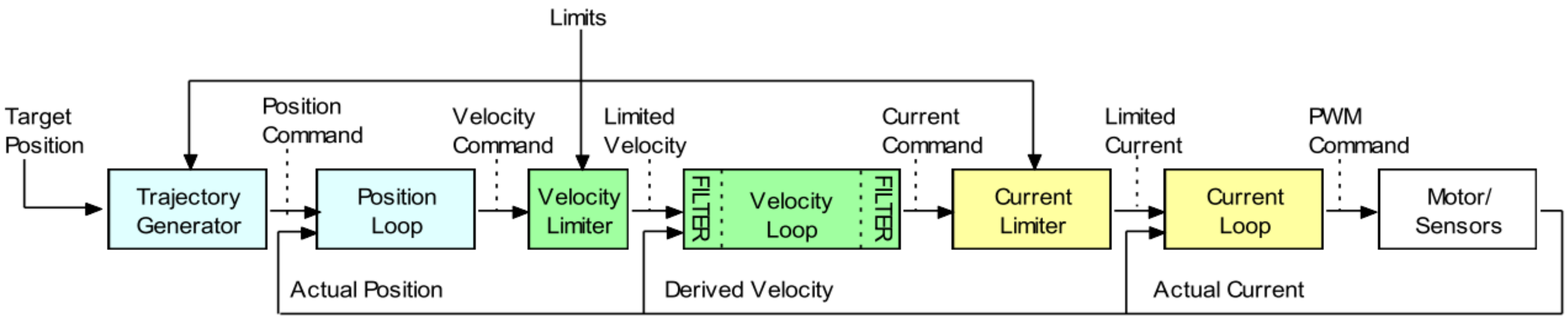
Robotic Arm Dynamic Control Technology

Dynamic Equations

- Newton-Euler equation and the Euler-Lagrange equation are common methods to derive dynamic equations. We represent length, mass, inertia and centroid location of robotic arm as symbols through MATLAB symbolic toolbox, and then derive the dynamic equations by the Newton-Euler equation and Euler-Lagrange equation, respectively, to get $M(\theta)$, $C(\theta, \dot{\theta})$, $G(\theta)$. After verification, the τ derived by the Newton-Euler equation and by the Euler-Lagrange equation are matched.

Robotic Arm Dynamic Control Technology
Current Loop System Identification

- Accelnet Plus driver provides position, velocity and current mode. Due to that current is proportional to torque, the dynamic control is implemented using the current mode. In order to determine whether we need to compensate or not, we have to get the frequency response of the current loop of the servo motor.



The driver control loop block diagram

Robotic Arm Dynamic Control Technology
Current Loop System Identification

- The process of the system identification is as follows:
  1) Set the parameters of current loop controller (PI controller's Kp, Ki)
  2) Input different frequency of sinusoidal current command to motor by CME2, respectively, and record the magnitude and phase of input and output current to plot bode plot.
  3) Combine the input current command and output current in step b, and identify the system by using MATLAB identification toolbox.
  4) Input the test sinusoidal command to the model from step 3, and then compare the model output with the actual model output to verify the result of identification.

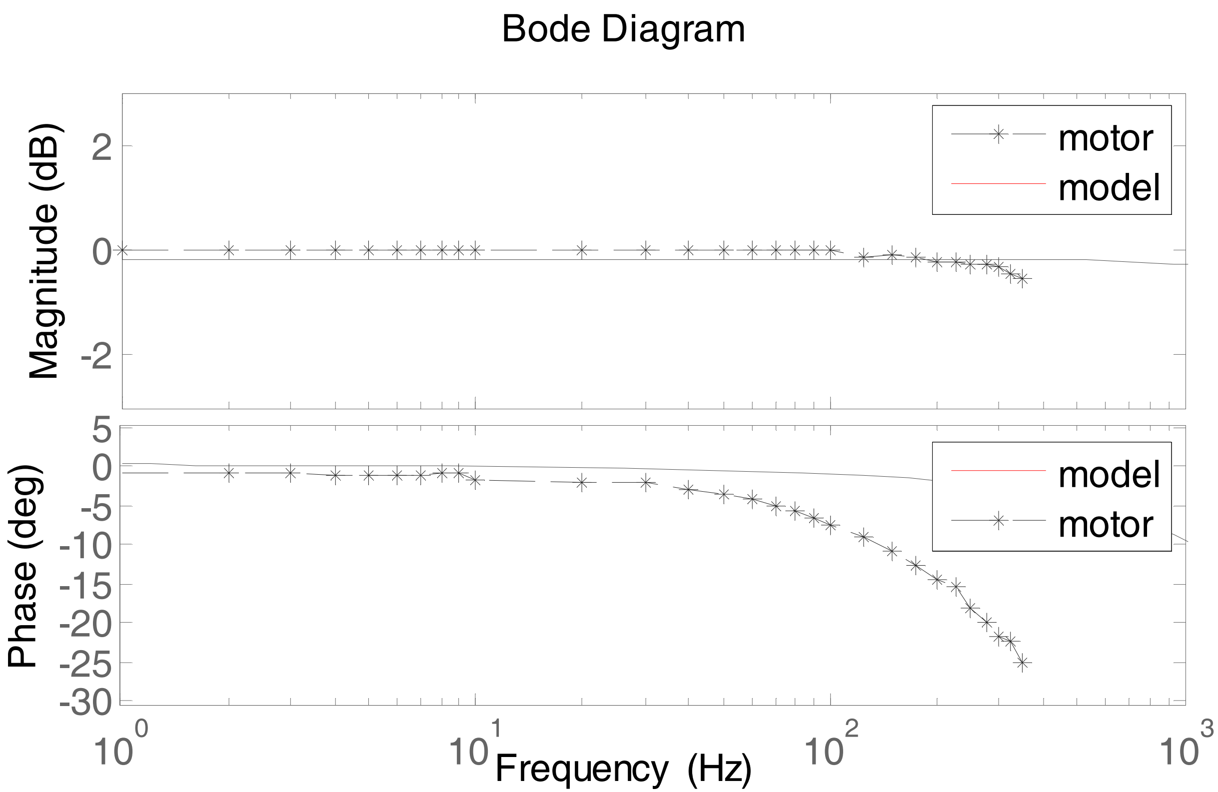
Robotic Arm Dynamic Control Technology Experimental Result

- DC brushless motor: KBM-10H01-C00
- Digital servo drive: Accelnet Plus Module EtherCAT AEM-090-14
- Absolute encoder: DS-58[20] (18 bits Angular resolution)

- Obtained model：$\frac{37760\,s + 18490}{s^2 + 38470\,s + 20080}$

Robotic Arm Dynamic Control Technology Experimental Result

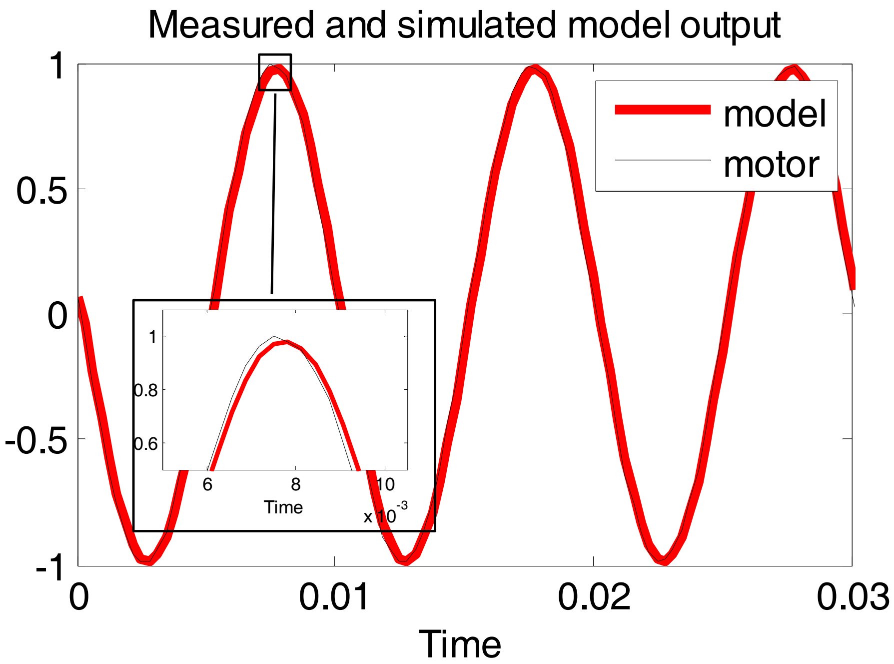
- Comparing the bode plot of models and servo motor as follows:



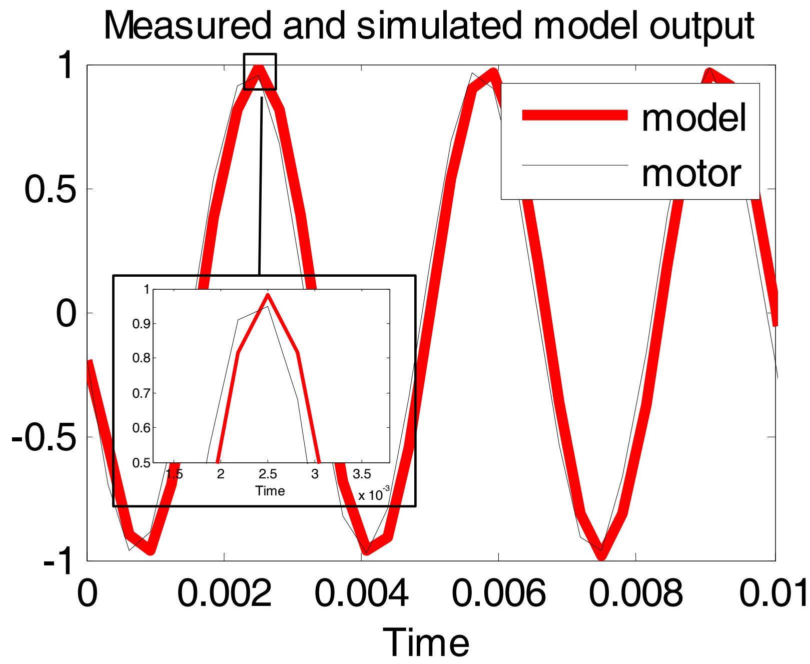
Bode plot of models and servo motor

Robotic Arm Dynamic Control Technology Experimental Result

- Input the sinusoidal test command and compare the outputs of the model and the motor.



100HZ



300HZ

Robotic Arm Dynamic Control Technology
Future Work

- Solve the singularity problem of 7-axis Jacobian Matrix.
- Build the process of robot arm system identification to identify the parameters of dynamic model.
- Develop high efficiency motion control law of robot arm.

Outline

- Advanced Robot Arm
- Real-time OS
- EtherCat Protocol
- Robotic Arm Dynamic Control Technology
- **3D Objects Pose Estimation**
- Intelligent Graphic User Interface System
- Advanced Hand-Eye-Workspace Calibration

3D Objects Pose Estimation

Outline

- Introduction
- Problem Statement
- CAD Database system
- Hand-Eye Calibration
- Experimental Result
- Conclusion and Feature Work

3D Objects Pose Estimation

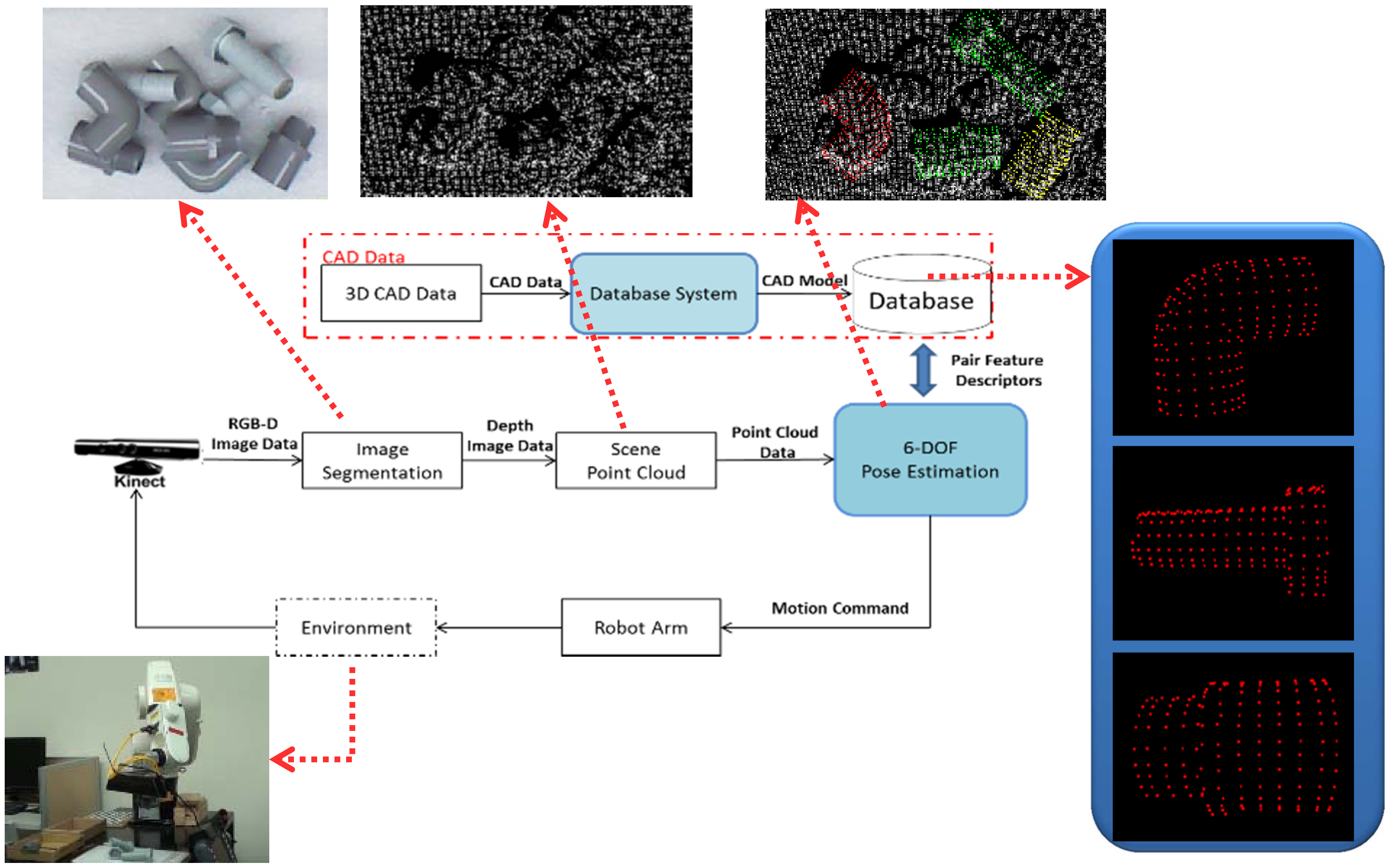
Introduction

- With the advent of new-generation depth sensors, using 3D data is becoming increasingly popular. As these sensors are commodity hardware and sold at low cost, a rapidly growing group of people can acquire 3-D data cheaply and in real time.
- For a texture-less object like a mechanical part, conventional visual feature matching usually fails due to the absence of rich texture features.
- The vision system is easily set up to recognize different objects by using CAD models database.

3D Objects Pose Estimation
Problem Statement

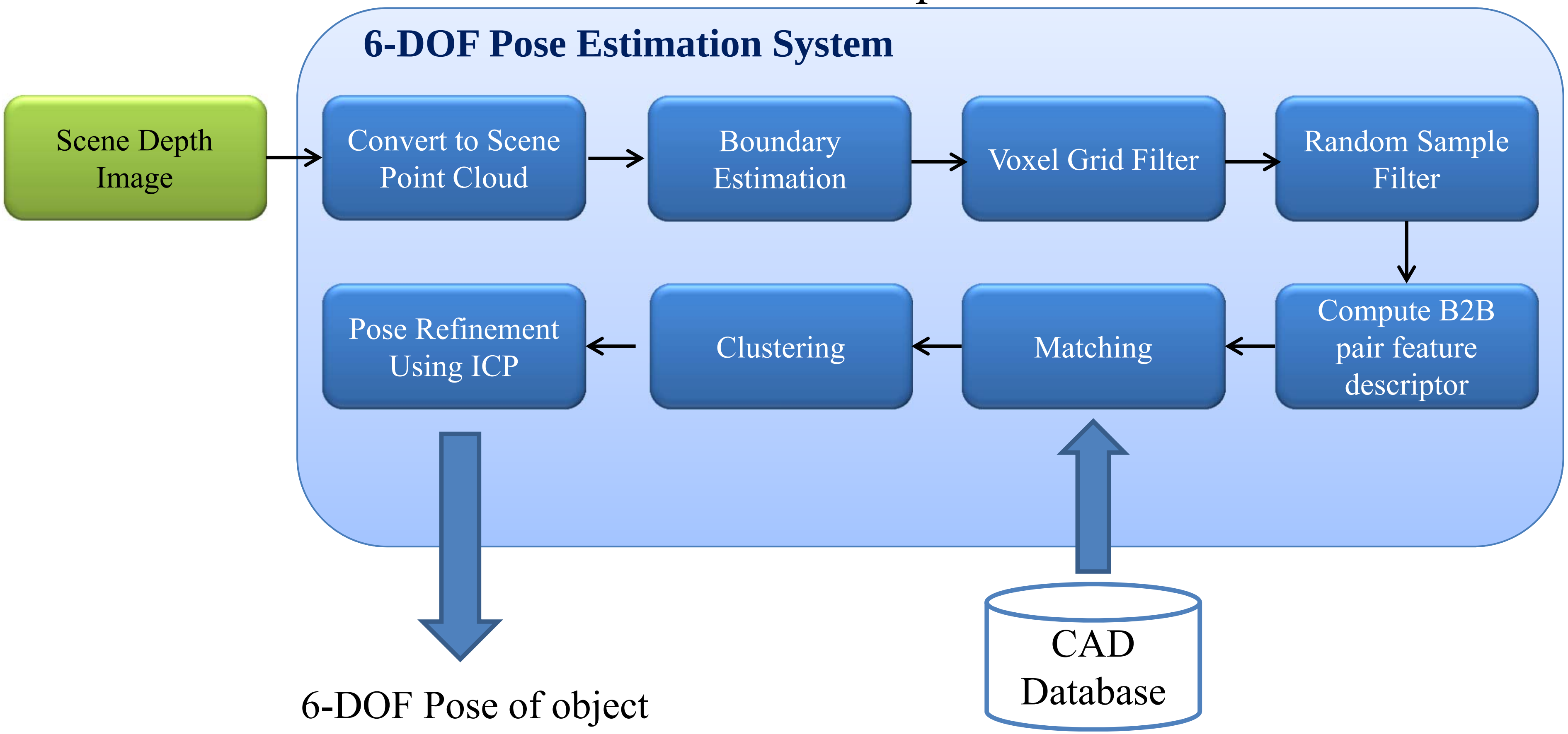
- How to carefully chosen feature descriptor encode more information compactly and thereby provide higher accuracy 6-DOF pose estimation and enable faster computation.
- How to use CAD data to establish the CAD-model database, so that the system not only work even in highly cluttered environment, but also to accurately estimate 6-DOF pose of the workpieces.

System Architecture



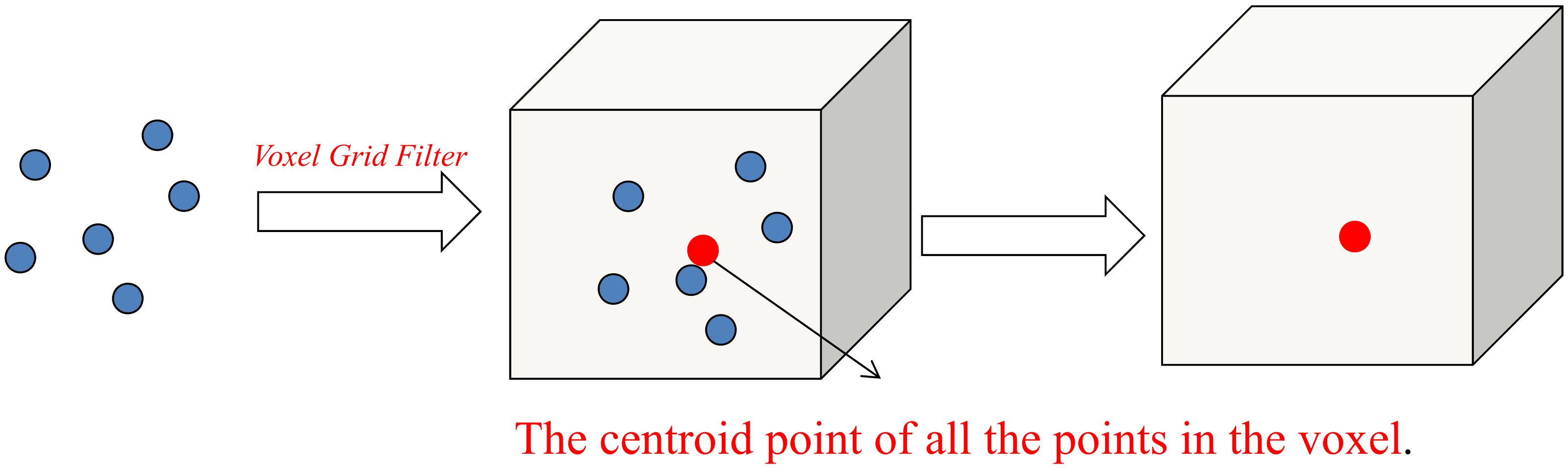
3D Objects Pose Estimation
6-DOF Pose Estimation System

- The system can convert scene depth image to point cloud format, and estimate the 6-DOF pose of object by using CAD Database which has been established in offline phase.

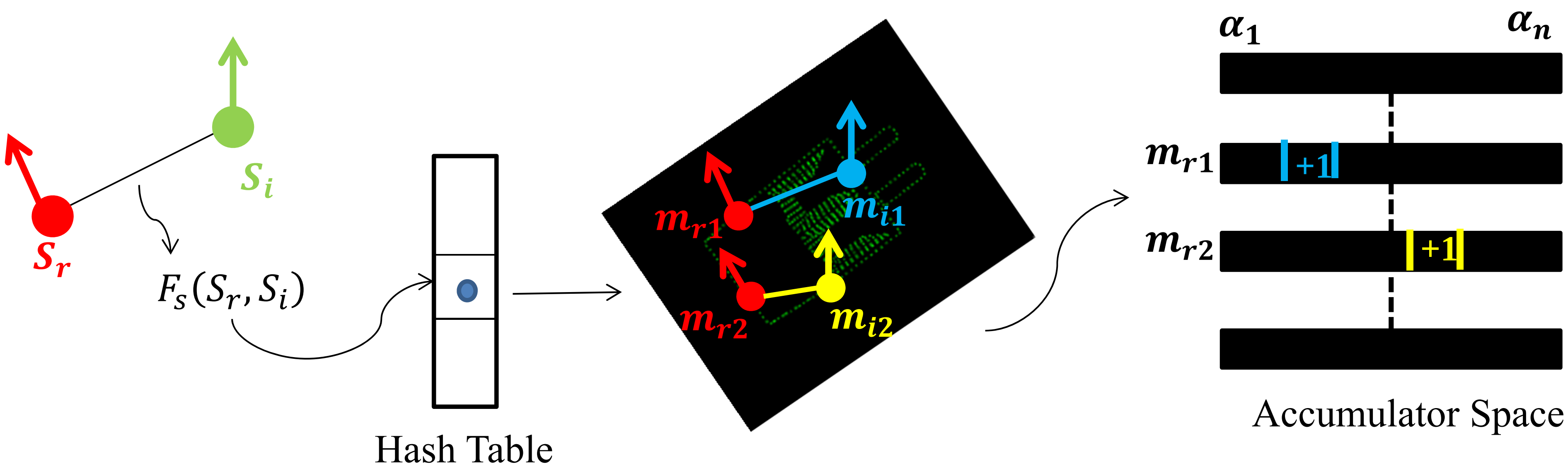


3D Objects Pose Estimation
6-DOF Pose Estimation System

- The Voxel Grid Filter created 3D voxel grids from the input point cloud data, then points in each voxel (i.e. 3D box) will be describe as their centroid.



3D Objects Pose Estimation
Voting-Scheme Matching



(1) Reference point Sr is paired with every other point Si, and their point pair feature F is calculated.

(2) Feature F is matched to the hash table, which returns a set of point pairs on the model that have similar distance and orientation.

(3) For each point pair on the model matched to the point pair in the scene, the local coordinate is calculated by solving $\mathbf{s}_i = T_{s\to g}^{-1} R_{\mathbf{x}}(\alpha) T_{s\to g} \mathbf{m}_i$.

(4) After is calculated, a vote is cast for the local coordinate $(\mathbf{m}_i, \alpha)$

3D Objects Pose Estimation

Pose Clustering

- The retrieved poses are clustered such that all poses in one cluster do not differ in translation and rotation for more than a predefined threshold.

$$if\ (\ trans_{pose\_n} < \ threshold_{trans\_n}\ )\ and\ \ (\ rotation_{pose\_n} < \ threshold_{rotation\_n}\ )$$

=> *pose_n is put in cluster_n*

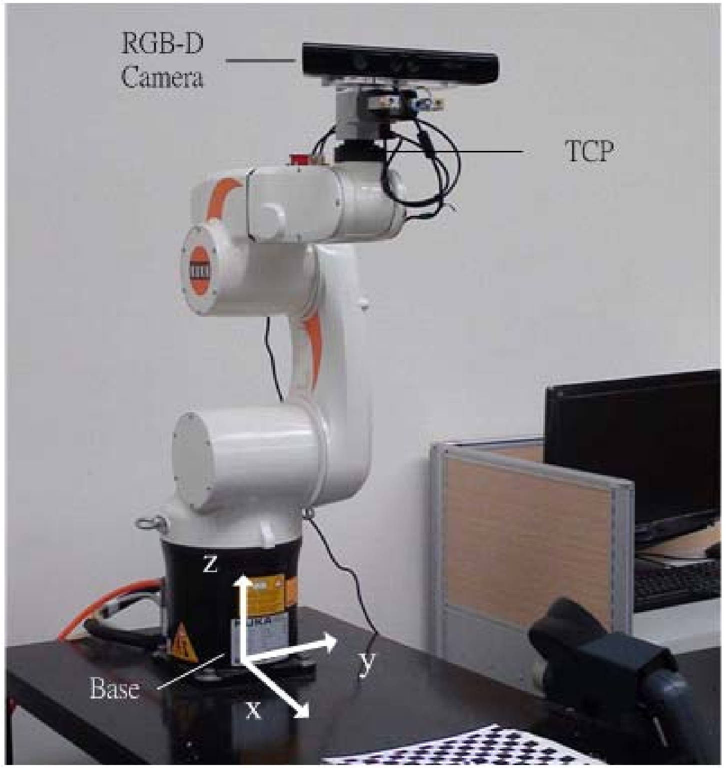
- The score of a cluster is the sum of the scores of the contained poses, after finding the cluster which vote is bigger than threshold, the resulting pose is calculated by averaging the poses contained in the cluster.

$$if\ vote_{cluster\_n} > threshold$$

=> *cluster_n is final pose estimation*

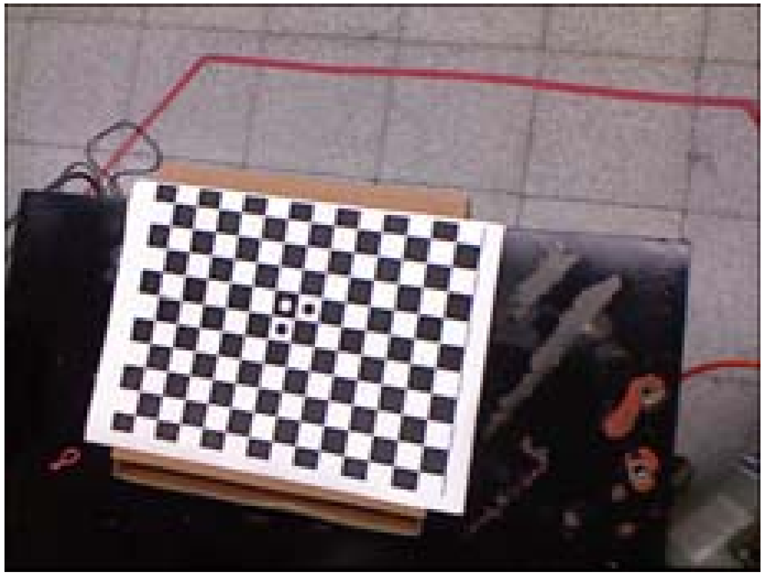
3D Objects Pose Estimation
Eye-in-Hand Calibration

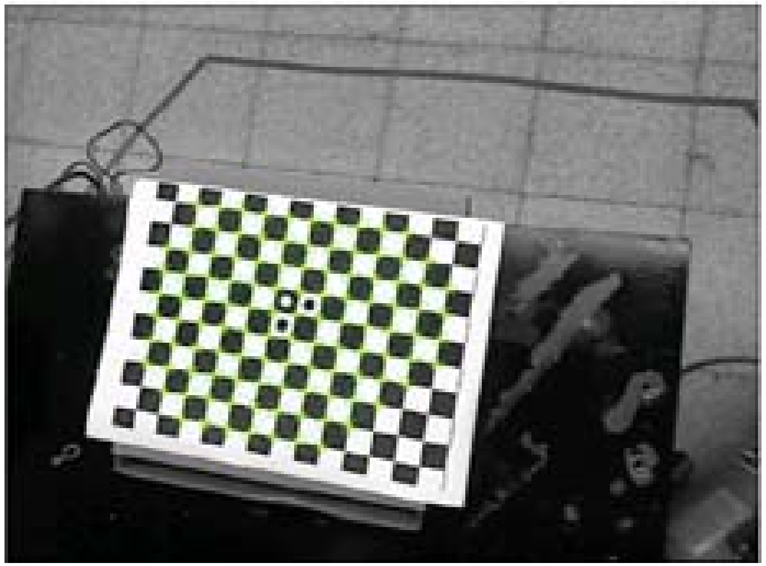
- We set up an eye-in-hand configuration using the Kuka 6-DOF robot arm installed with a Kinect camera.
- We first calibrated the eye-in-hand configuration by using the DLR Camera Calibration Toolbox.



3D Objects Pose Estimation
Calibration Result

- We took 15 images of the calibration pattern with varying orientations and translation. The error means are 2.54 mm, 7.99mm and 2.74mm in x-coordinate, y-coordinate and z-coordinate, respectively.





3D Objects Pose Estimation
Experimental Result

3D Objects Pose Estimation
Conclusion and Feature Work

- The 6-DOF object pose estimation based on CAD-model database has been proposed.
- We have implemented the 6-DOF pose estimation system in kuka robot. The experimental result shown that the proposed method can pick 3-4 bins in one minute.
- In the feature, we will speed up the cycle time of the method.

Outline

- Advanced Robot Arm
- Real-time OS
- EtherCat Protocol
- Robotic Arm Dynamic Control Technology
- 3D Objects Pose Estimation
- **Intelligent Graphic User Interface System**
- Advanced Hand-Eye-Workspace Calibration

Intelligent Graphic User Interface System

Goal

- The graphic user interface system is developed for operating an *industrial robotic arm*, with the implementation on the tablet of an Android based application.
- The graphic user interface system:
  - Providing a more ease-to-use virtual interface for the operator to execute the task.
  - Providing a platform to realize the remote control.

Intelligent Graphic User Interface System

Communication

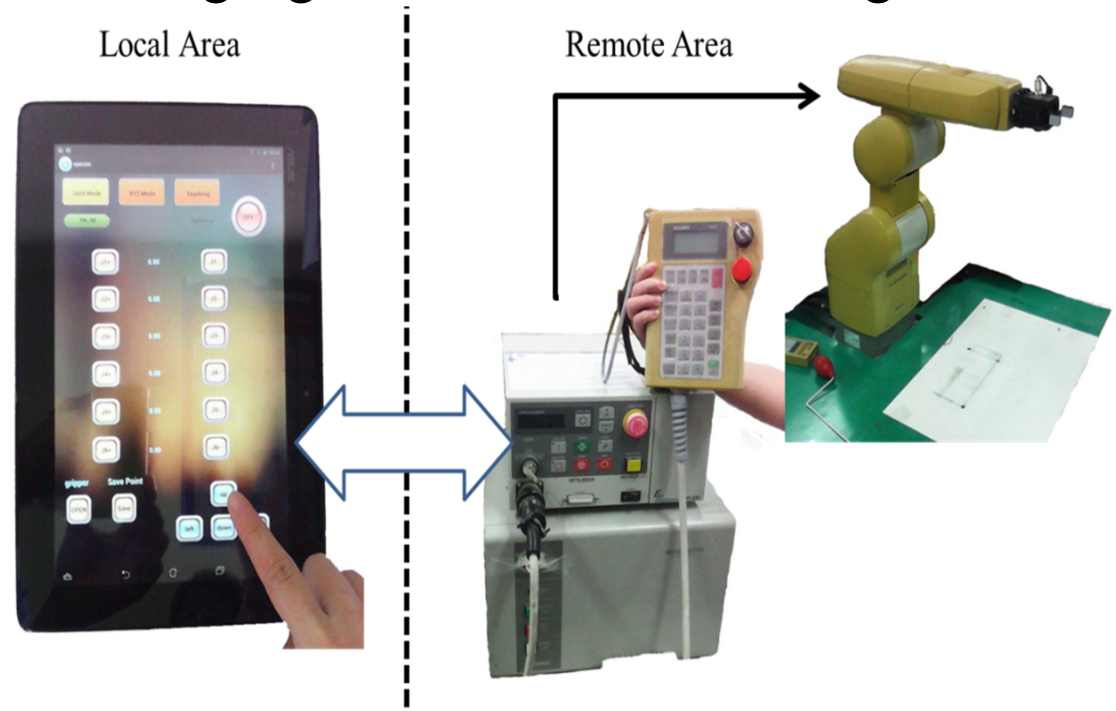
- The WiFi technology makes it possible for us to control robot *remotely*, while tablet makes it possible for us to control the robot in the visual interface.
- The programming based on TCP Windows Sockets is used to realize the communication.
  - TCP Windows Sockets:
    - A reliable transport protocol
    - Ordered
    - Streaming

Intelligent Graphic User Interface System

Communication

- It's a simple way to realize the *Client / Server* model.

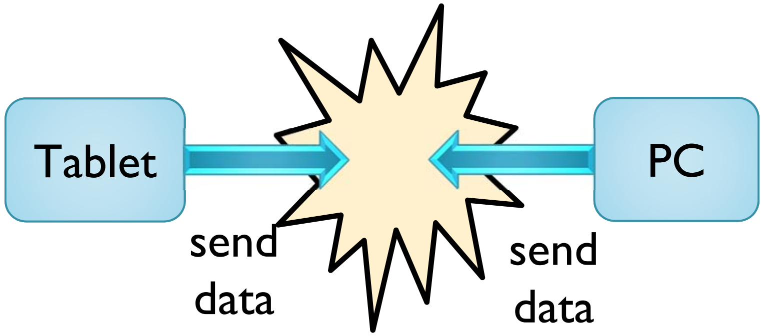
  - Server:
    - PC: Microsoft Visual Studio
    - Program: C language
  - Client:
    - Android tablet: eclipse
    - Program: Java language



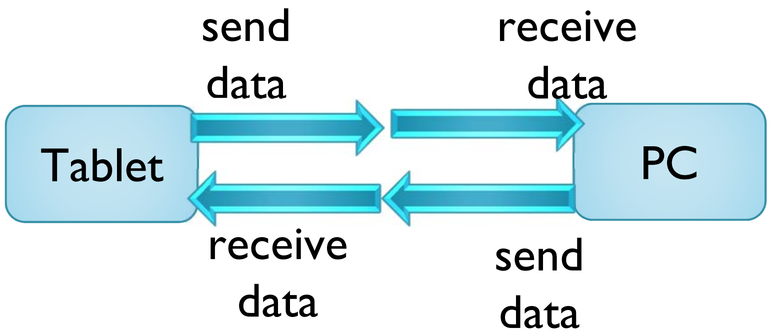
Intelligent Graphic User Interface System
Communication

- PC：The program uses multi-threads.
  - Multi-thread is the ability of a program to manage multiple requests at one time in the computer.

Single-Thread



Multi-Thread



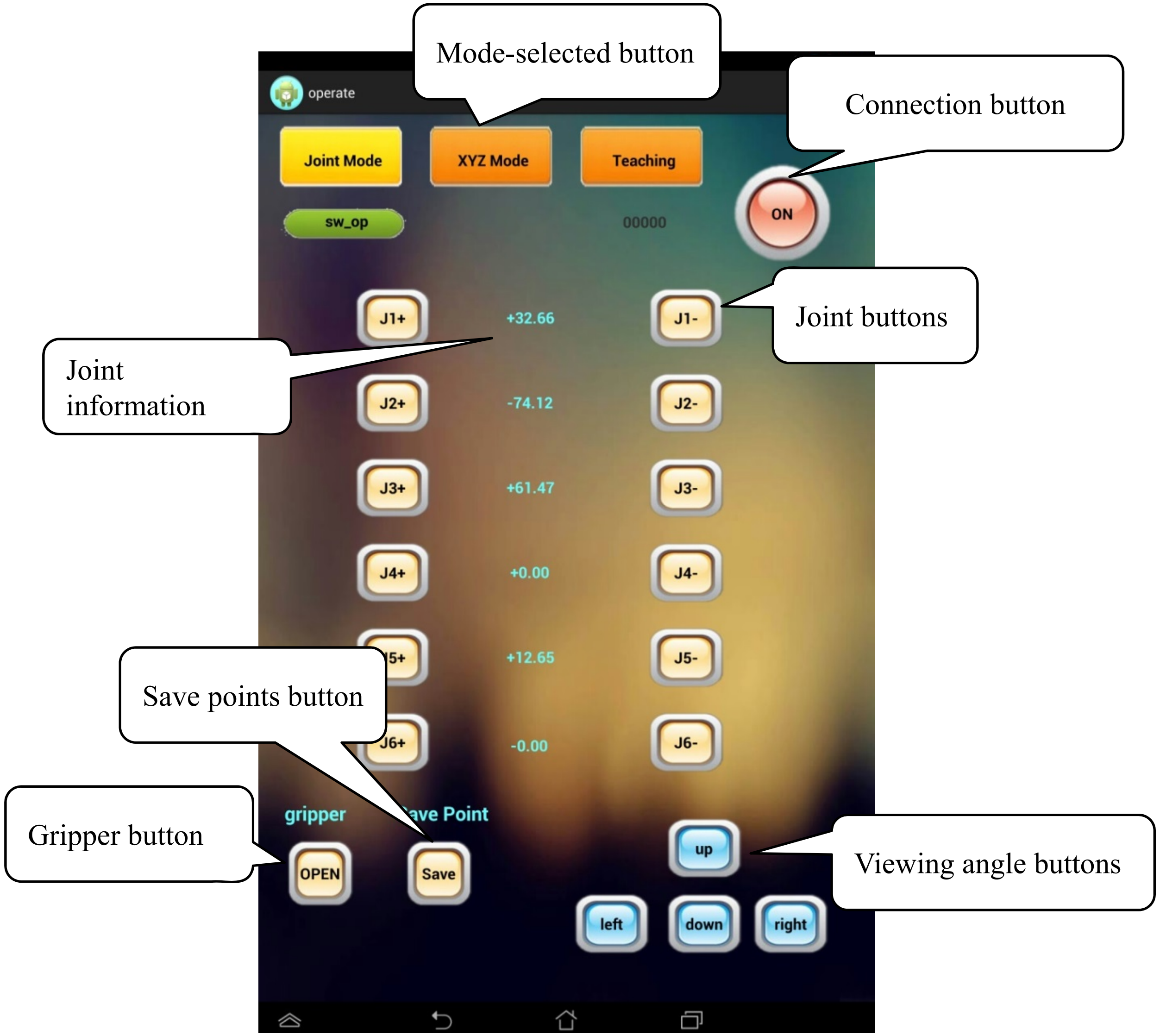
Intelligent Graphic User Interface System

Specifications

- Tablet
  - ASUS Memo Pad FHD ME302KL-1B024A
  - Qualcomm 8064 Quad-core 1.5GHz
  - OS：Android4.2
  - Size：264 x 182 x 9.5mm
  - LCD size：10.1-inch IPS
  - WiFi、Bluetooth

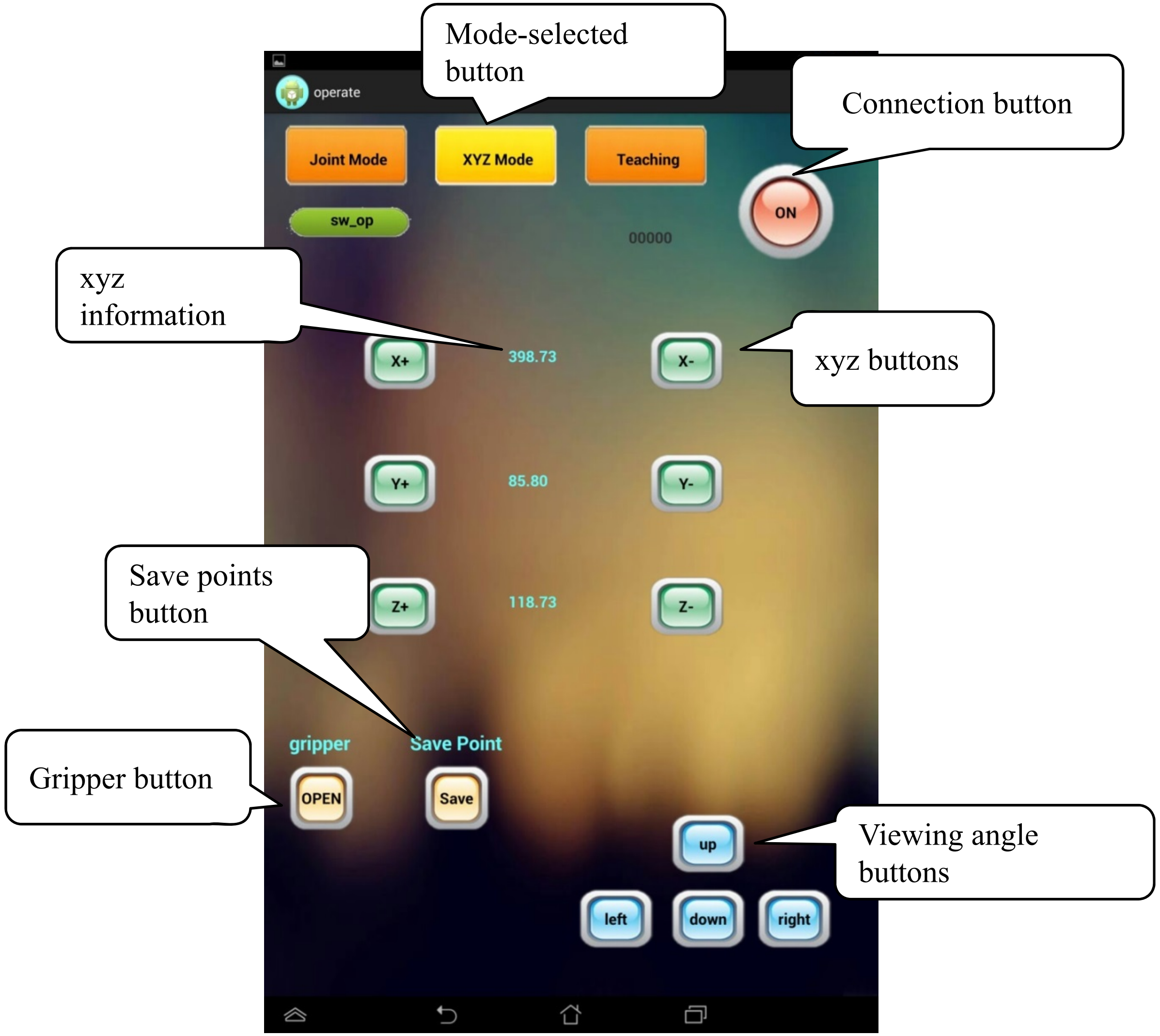
Intelligent Graphic User Interface System
Operation Interface

- Tablet
  - Joint Mode



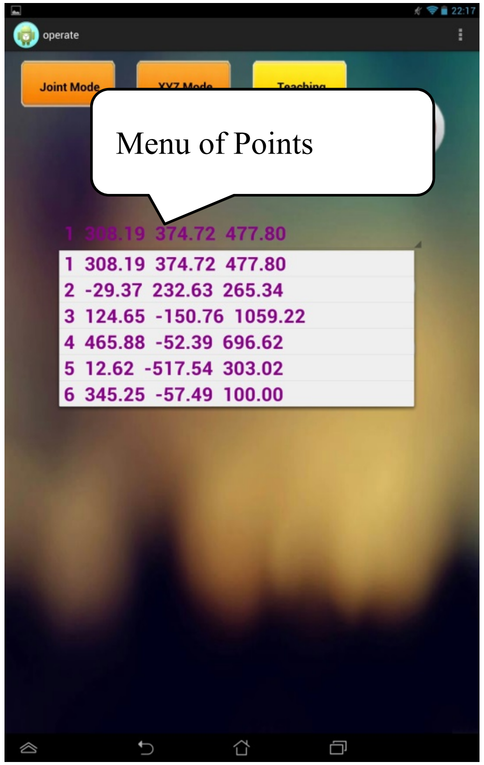
Intelligent Graphic User Interface System
Operation Interface

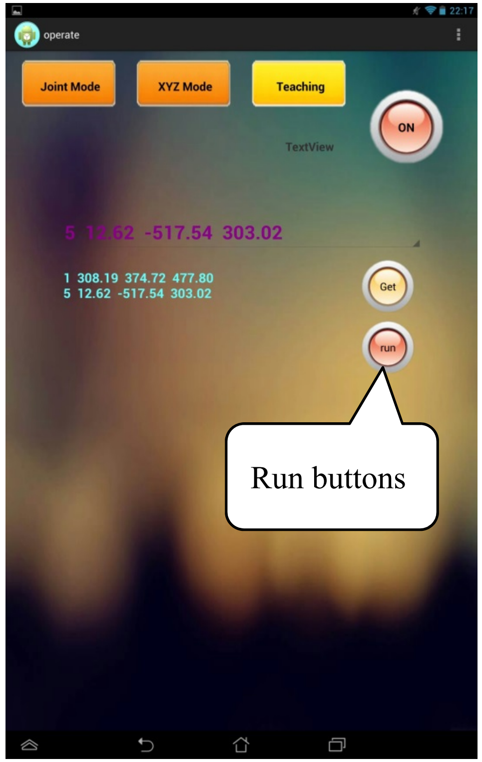
- Tablet
  - xyz Mode



Intelligent Graphic User Interface System
Operation Interface

- Tablet
  - Teach Mode



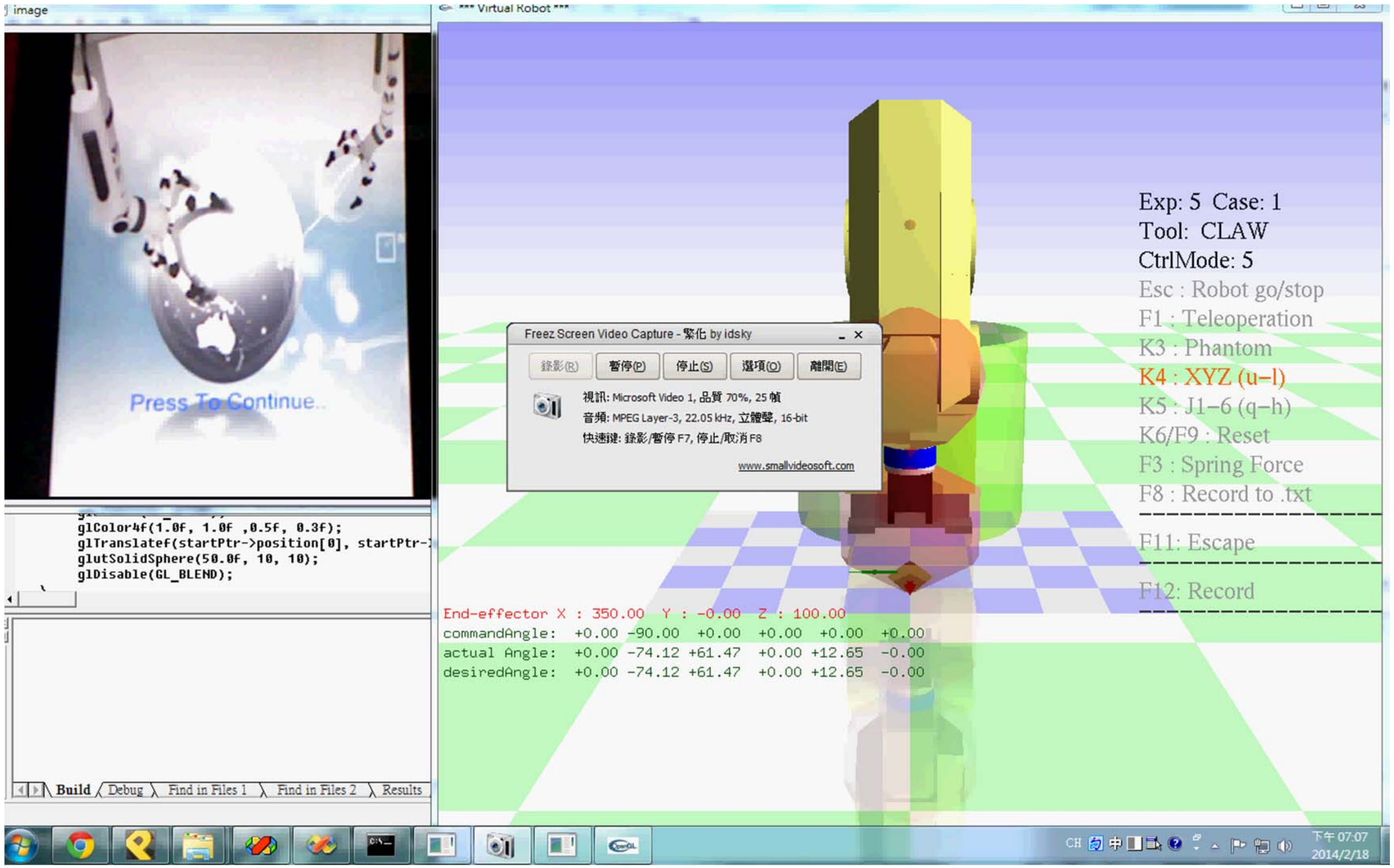


Intelligent Graphic User Interface System
Operation Interface

- Tablet
  - Knob Mode

Intelligent Graphic User Interface System

Demo



Outline

- Advanced Robot Arm
- Real-time OS
- EtherCat Protocol
- Robotic Arm Dynamic Control Technology
- 3D Objects Pose Estimation
- Intelligent Graphic User Interface System
- **Advanced Hand-Eye-Workspace Calibration**

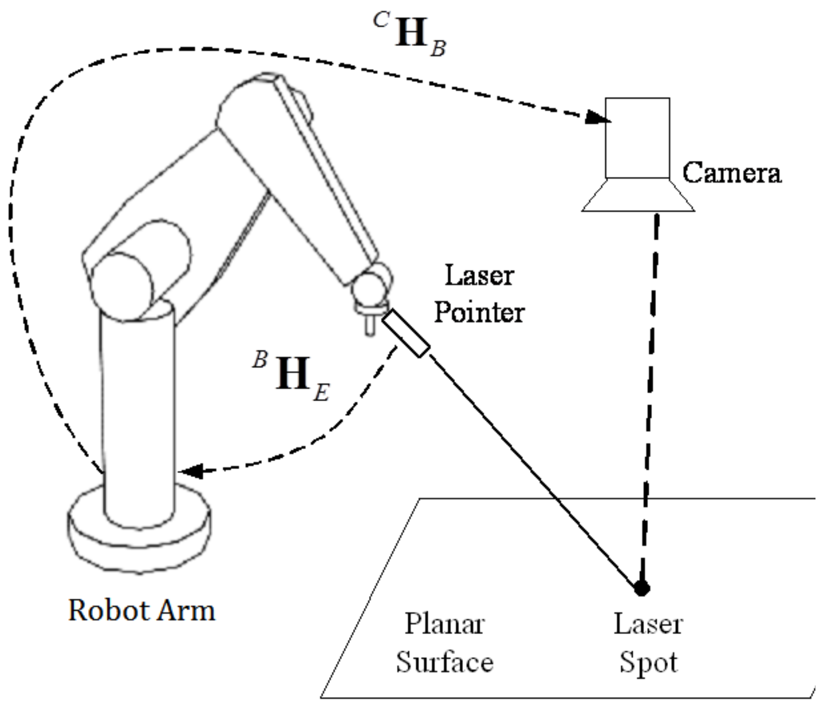
Advanced Hand-Eye-Workspace Calibration

Introduction

- The proposed Hand-Eye-Workspace calibration system including a laser pointer installed on the robot arm, a stationary camera, and a planar surface. The laser pointer beams to the surface, and the camera observes the projected laser spot. It is not necessary for robot arm to enter the view scope of the camera for calibration. The configuration of camera is more flexible.

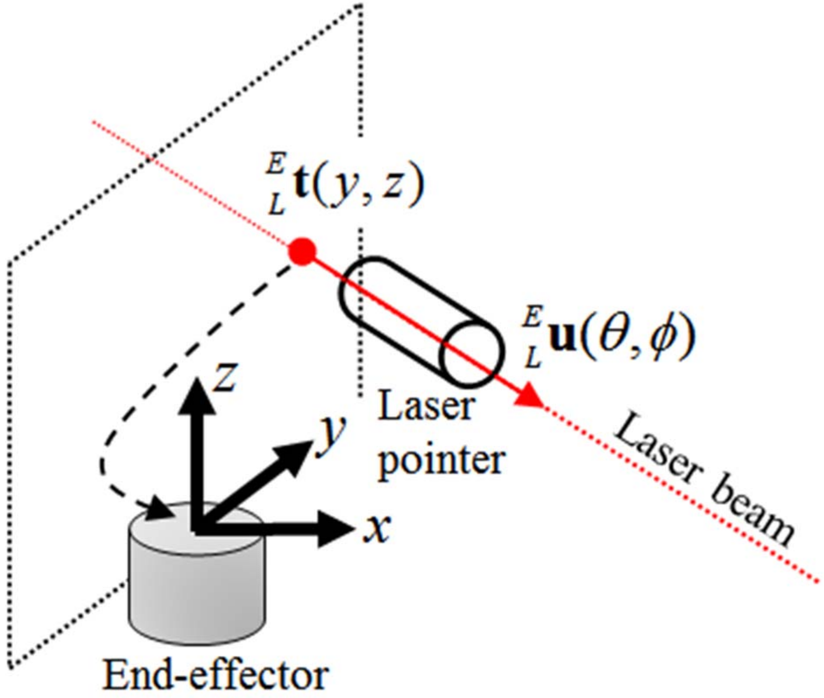
Advanced Hand-Eye-Workspace Calibration
Definition

- ${}^{C}\mathbf{H}_{B}$: The transformation matrix between the fixed camera and the robot base. It represents the camera extrinsic matrix.
- ${}^{B}\mathbf{H}_{E}$: The transformation matrix between the robot base and the end-effector. It represents the robot kinematics.
- The relationship between the laser pointer and the end-effector.
- Plane equation
- Camera intrinsic parameters



Advanced Hand-Eye-Workspace Calibration
Definition

The laser line is defined in the end-effector coordinate with two position parameters (y, z) and two Euler angles (θ, ϕ). The values of four parameters are unique. The direction of line is computed with (θ, ϕ) as



$$
{}^{E}_{L}\mathbf{u}\left({}^{E}_{L}\theta, {}^{E}_{L}\phi\right) = \begin{bmatrix} \cos {}^{E}_{L}\theta \sin {}^{E}_{L}\phi & \sin {}^{E}_{L}\theta \sin {}^{E}_{L}\phi & \cos {}^{E}_{L}\phi \end{bmatrix}^{T}
$$

Advanced Hand-Eye-Workspace Calibration
Line-Plane Intersection

- Laser line

  The laser line in the camera coordinate can be obtained using ${}^{C}\mathbf{H}_{B}$, ${}^{B}\mathbf{H}_{E}$ and $\left({}^{E}_{L}y, {}^{E}_{L}z, {}^{E}_{L}\theta, {}^{E}_{L}\phi\right)$. The direction is denoted as ${}^{C}\mathbf{z}_{laser}$. The position is denoted as ${}^{C}\mathbf{p}_{laser}$.

- Plane definition

  A plane is generally described by the normal vector **n** and the position vector **p** of any point on the plane, but it can be simply defined by the unique position vector whose direction is the normal vector as

$$\mathbf{a} = \begin{cases} \mathbf{n}, & \text{if } \mathbf{n}\cdot\mathbf{p} = 0 \\ (\mathbf{n}\cdot\mathbf{p})\mathbf{n} & \text{otherwise} \end{cases}$$

Advanced Hand-Eye-Workspace Calibration
Line-Plane Intersection

- Line-Plane intersection

  The laser spot on the plane can be defined with the distance d_{laser} as ${}^{C}\mathbf{p}_{spot} = d_{laser}\,{}^{C}\mathbf{z}_{laser} + {}^{C}\mathbf{p}_{laser}$. It matches the geometrical constraint as

$$\left(d_{laser}\,{}^{C}\mathbf{z}_{laser} + {}^{C}\mathbf{p}_{laser} - \mathbf{a}\right)\cdot\mathbf{a} = 0.$$

  The distance d_{laser} can be obtained as

$$d_{laser} = \frac{\|\mathbf{a}\| - {}^{C}\mathbf{p}_{laser}\cdot\mathbf{a}}{{}^{C}\mathbf{z}_{laser}\cdot\mathbf{a}}.$$

Advanced Hand-Eye-Workspace Calibration
Camera Model

- The camera model is a pin-hole type including lens distortion. The 2D position (u ,v) in an image of a 3D point ${}^{C}\mathbf{p} = [{}^{C}x,\ {}^{C}y,\ {}^{C}z]^T$ can be obtained with the intrinsic matrix $\mathbf{K}$ and the lens distortion parameters (κ_1, κ_2, ρ_1, ρ_2) as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \cdot \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \quad \text{with } \mathbf{K} = \begin{bmatrix} f_u & \alpha_c \cdot f_u & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Advanced Hand-Eye-Workspace Calibration
Camera Model

- The camera model is a pin-hole type including lens distortion. The 2D position (u, v) in an image of a 3D point ${}^{C}\mathbf{p} = [{}^{C}x,\ {}^{C}y,\ {}^{C}z]^T$ can be obtained with the intrinsic matrix $\mathbf{K}$ and the lens distortion parameters $(\kappa_1, \kappa_2, \rho_1, \rho_2)$ as

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \left(1 + \kappa_1 \|\mathbf{x}_r\|^2 + \kappa_2 \|\mathbf{x}_r\|^4\right)\mathbf{x}_r + \begin{bmatrix} 2\rho_1 x_r y_r + \rho_2(\|\mathbf{x}_r\|^2 + 2x_r{}^2) \\ \rho_1(\|\mathbf{x}_r\|^2 + 2y_r{}^2) + 2\rho_2 x_r y_r \end{bmatrix}$$

where $\mathbf{x}_r = \begin{bmatrix} x_r & y_r \end{bmatrix}^T = \begin{bmatrix} {}^{C}x/{}^{C}z & {}^{C}y/{}^{C}z \end{bmatrix}^T$ is the ray direction from the camera.

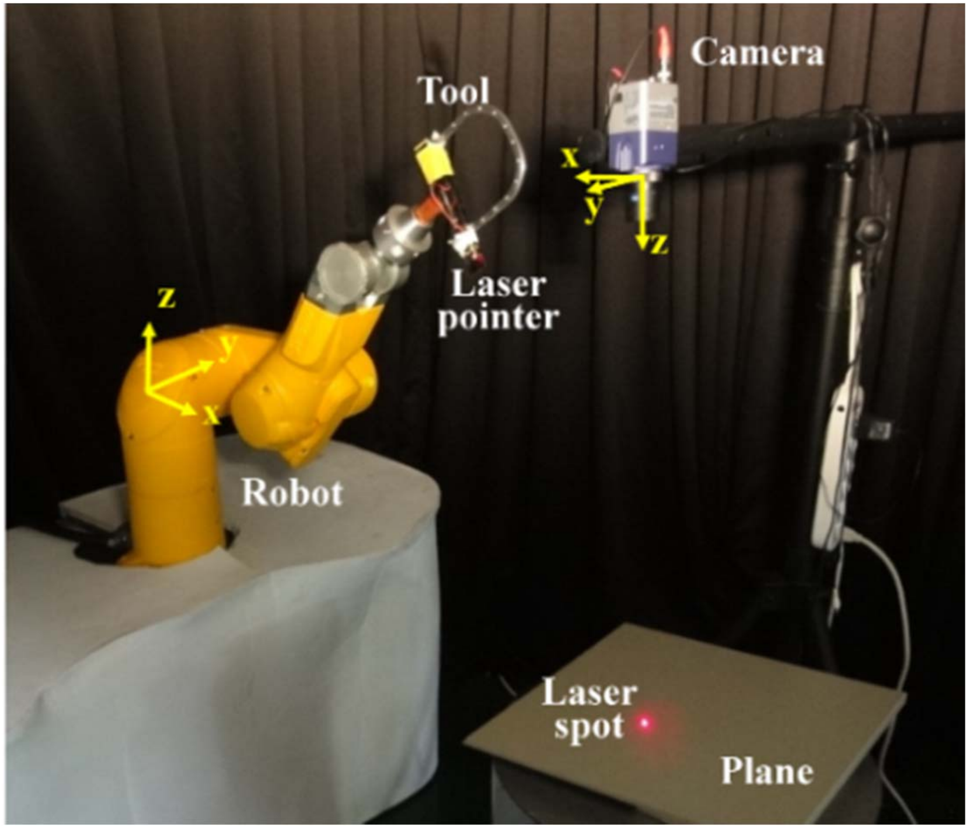
Advanced Hand-Eye-Workspace Calibration
All Parameters

- ${}^{C}\mathbf{H}_{B}$: The transformation matrix between the fixed camera and the robot base. It is defined with the Euler angles $\left({}^{B}_{C}\theta, {}^{B}_{C}\phi, {}^{B}_{C}\varphi\right)$ and the position ${}^{B}_{C}\mathbf{t}$.
- ${}^{B}\mathbf{H}_{E}$: The transformation matrix between the robot base and the end-effector. The robot kinematic parameters.
- The relationship between the laser pointer and the end-effector is defined with Euler angles $\left({}^{E}_{L}\theta, {}^{E}_{L}\phi\right)$ and the position ${}^{E}_{L}\mathbf{t}$.
- Plane equation : $\mathbf{a}$
- Camera intrinsic matrix $\mathbf{K}$ and the lens distortion parameters $(\kappa_1, \kappa_2, \rho_1, \rho_2)$

Advanced Hand-Eye-Workspace Calibration
Experiment

- For Hand-Eye-Workspace calibration, the goal is to estimate the relationship between the robot arm, the camera and the workspace plane. Assuming the robot arm and the camera are pre-calibrated. The estimated parameters include
  - $\left({}_{C}^{B}\theta, {}_{C}^{B}\phi, {}_{C}^{B}\varphi\right)$ and ${}_{C}^{B}\mathbf{t}$: the transformation matrix between the fixed camera and the robot base.
  - $\left({}_{L}^{E}\theta, {}_{L}^{E}\phi\right)$ and ${}_{L}^{E}\mathbf{t}$: the relationship between the laser pointer and the end-effector.
  - $\mathbf{a}$: the plane parameters.

Advanced Hand-Eye-Workspace Calibration
Experiment



Experiment Configuration

Advanced Hand-Eye-Workspace Calibration

Experiment

- The robot arm, Stäubli TX60, is operated to generate multiple laser spots on the plane in the view of camera. The measurement is the (u, v) position of laser spot. The optimal solution is obtained by minimizing the total position errors using nonlinear optimization method (the Levenberg-Marquardt method).
- The laser spot position in the image is obtained by two steps. First, the grey image converts to binary image using the simple threshold method. It can reject the image noise. Second, the average position of light pixels is the laser spot position.

Advanced Hand-Eye-Workspace Calibration Experiment

Calibration Result

Parameter		30 samples	50 samples	Unit
	Initialized	Refined	Refined	
$({}^B_C\theta, {}^B_C\phi, {}^B_C\varphi)$	(-100.14, -3.29, -172.69)	(-102.55, -6.15, -174.22)	(-102.60, -5.88, -173.53)	degree
${}^B_C\mathbf{t}^T$	[1343.73, -4.34, 319.28]	[965.56, -47.41, 440.90]	[976.59, -38.35, 436.62]	mm
$({}^E_L\theta, {}^E_L\phi)$	(-89.93, 80.21)	(-91.15, 84.45)	(-91.13, 84.36)	degree
${}^E_L\mathbf{t}^T$	[0, -30.14, 423.31]	[0, 8.55, 135.06]	[0, 7.49, 140.77]	mm
$\mathbf{a}^T$	[-70.39, 55.95, 690.76]	[-50.57, 35.52, 705.07]	[-61.34, 33.18, 702.78]	mm
RMS	30.32	1.0626	1.1494	pixel

Advanced Hand-Eye-Workspace Calibration

Conclusion

- The proposed calibration system is cost-efficient and flexible for any manipulator. It can be extended to calibrate the camera intrinsic parameters and the kinematic parameters of robot arm.