

A FLYING OBJECT USING HARDWARE IMPLEMENTED VISION PROCESSING AND MOTOR CONTROL SYSTEM WITH ADAPTIVE NEURAL NETWORK

Hitoshi Yamada, Johane Takeuchi, Gen Matsumoto, and Michinori Ichikawa

Laboratory for Brain-Operative Devices, Riken BSI,
2-1 Hirosawa, Wako, Saitama, 351-0198, Japan
hitoshi@yamadada.com

ABSTRACT

We made a flying object with 4 CMOS video cameras and 4 propellers. The processor consists of visual processing module, the attitude calculation module and adaptive module implemented by using neural feedback network. The visual processing modules, which are including 3 CMOS cameras, output displacements of images for each local area using difference between frames. The attitude calculation module integrates those displacement values and output rotational velocity of each axis (yaw, roll, pitch). The last CMOS camera observes ground, and its processing module output relative position between the ground and the flying object. The adaptive module finally calculates motor outputs basically depending on PID feedback control loop with reinforcement learning, and the learning method is weight changes as neural network between derived parameters. A preliminary manual tuning fixes initial parameters of the PID control. And on-line learning could tune those parameters more precisely during actual flight experiments.

We introduced nonlinear response functions and switching mechanism of those responses into the adaptive module. For instance, nonlinear response functions are followings

1. Reduce feedback gains to cool down oscillation
2. Add offset value to cancel supposed disturbance
3. Stop all motors to avoid fatal damage.

1. INTRODUCTION

Our aim is creating a computer, which mimics a function of a brain. Especially a flying object reported in this paper was build up to evaluate functions of motor and behavior control. We have already presented an experiment of a small indoor helicopter (ref. [1]), which was focused on visual processing and motor control inspired by an insect "fly".

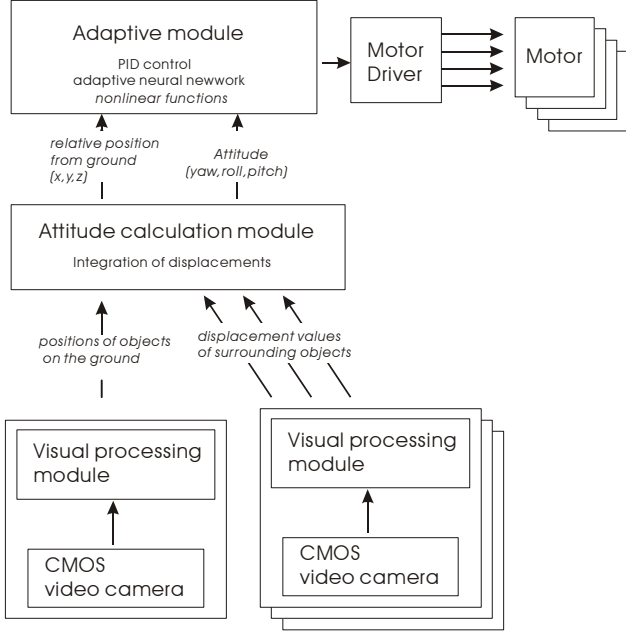
Essential difference between biological eyes and video cameras is discontinuity of time because such camera scans sequentially in certain readout time. To mimic such

biological eye system, analog processing seems to be essential [Ref.2]. But we had overcome this problem with digital circuits using proper accumulating technique for each frame in time domain. In this paper Image Processing section describes these techniques, which is actually implemented as electrical circuit on FPGA.

2. IMPLEMENTATION

The flying object made of cross shaped carbon chassis mounting electronics and 4 motor units. The custom made carbon chassis's weighted 39.6g and its length is about 53cm. Four electric DC motor (Mabuchi FK180SH) drives 33cm (diameter) propellers, those two are driven clockwise and the other two are counter clockwise. Total weight of the flying object except battery is about 380g. It flies with DC 7.2V battery or wiring power supply.

The following figure is schematic of the object, which has 4 CMOS cameras, those controller, attitude calculation module, adaptive control unit, and motors.



3. ELECTRONICS

Four CMOS color image sensor “Omni Vision OV6630” was used to this experiment mounted on a PCB (Olympus komasa). The following is specs of the CMOS sensor.

- Array Element: (CIF) 352(H) x 288 (V)
- Frame speed: 40 FPS
- Scan: Progressive Scan
- Field of View: 60degree(H)
- Resolution: 0.17degree/pixel (horizontally)
- Power supply: 3.3V
- Dimension (komasa) : 10mm(W) x 20mm(D) x 6.6mm(H)
- Weight (komasa) : 1.0g

Totally 5 FPGA (Field Programmable Gate Array), and conventional CPU were used to perform this experiment. Three FPGA (Xilinx XCV100TQ144) were used to do image processing, which is mounted on each custom made PCB. A FPGA (Xilinx XCV300PQ240) integrate those three images processing FPGA's and calculate attitude values of the flying object. And it is also mounted on a custom made PCB with motor drive circuit.

The CPU (Hitachi SH4) is mounted on customized PCB (ALPHA PROJECT Co.,Ltd. AP-SH4-0A), and do adaptive control using values of the attitude.

4. IMAGE PROCESSING

Four individual processing units perform two kinds of visual processing. The first kind circuit derives displacement of surrounding object, which is used for stabilizing horizontal attitudes of the flying object so as to be a gyro function. This process was achieved using 3 CMOS cameras heading horizontally. The second kind processing, which is performed on the fourth camera,

derives positions of known markers on the ground, which lead absolute location and yaw of the flying object.

4.1. Displacement values of surrounding objects

3 CMOS camera and those processing circuits find displacement values of surrounding objects as a moving vector mentioned below. Then those moving vector will be used to calculate attitudes of yaw, roll and pitch of the flying object. In this section, we will show you newly way of finding a moving vector, which is suitable to implement as hardware and have great advantage to signal to noise ratio. And this method can have also advantage for solving problems of discrete sampling time of frames.

A moving vector \vec{m} can be described, as follows under condition of $M(u, v)$ is minimum, if a present frame is $f(x, y)$ and a past frame is $g(x, y)$.

$$\vec{m} = (u, v)$$

$$M(u, v) = \sum |f(x, y) - g(x + u, y + v)|$$

We have implemented a circuit to derive values, which alternate this moving vector, and the circuit has an advantage for problems of discrete sampling time and signal to noise ratio.

First we prepared following two data set, those describes a kind of mean value of each frame $Y(x, y, t)$.

$$Y_f(x, y, t) = \frac{1}{2} \sum_n \{Y_f(x, y, n) + Y_f(x, y, n - 1)\}$$

$$Y_s(x, y, t) = \frac{1}{2} \sum_m \{Y_s(x, y, m) + Y_s(x, y, m - 1)\}$$

n and m correspond to frame number of scan, and x and y indicate location in each frame (352(H) x 288 (V)).

i.e.

$$n=0, 1, 2, 3, 4, \dots, t$$

$$m=0, 2, 4, 6, \dots, t$$

$$x=1, 2, 3, \dots, 352$$

$$y=1, 2, 3, \dots, 288$$

$$Y_f(x, y, 0) = Y_s(x, y, 0) = 0$$

The actual values of those function $Y_f(x, y, t)$ and $Y_s(x, y, t)$ stored in frame memory. A calculation of the function as actual electric circuit needs only 3 clocks to read and to write values of frame memory.

Next, five set of subtraction of those two functions are prepared.

$$D_{fs}^{(1)}(t) = |Y_f(x, y - 1, t) - Y_s(x, y, t)|$$

$$D_{fs}^{(2)}(t) = |Y_f(x - 1, y, t) - Y_s(x, y, t)|$$

$$D_{fs}^{(3)}(t) = |Y_f(x, y, t) - Y_s(x, y, t)|$$

$$D_{fs}^{(4)}(t) = |Y_f(x+1, y, t) - Y_s(x, y, t)|$$

$$D_{fs}^{(5)}(t) = |Y_f(x, y+1, t) - Y_s(x, y, t)|$$

Actual calculation circuit of these $D^{(i)}(t)$ was implemented easily using 5 line buffers and also needed 3clocks to accomplish calculations. Then we induced a threshold level T_h to mask values of the functions $D^{(i)}(t)$ as follows,

$$S^{(i)}(t) = T[D^{(i)}(t)]$$

, where a mask operator is

$$T[D^{(i)}(t)] = D^{(i)}(t), (D^{(i)}(t) \geq T_h)$$

$$T[D^{(i)}(t)] = 0, (D^{(i)}(t) < T_h).$$

The latency of this circuit is 1 clock.

We counted number of pixels in a frame, which is under condition of $S^{(i)}(t) > T_h$.

$$C^{(i)}(t) = \sum_{x,y} U[S^{(i)}(t)]$$

i.e.

$$U[S^{(i)}(t)] = 1, (S^{(i)}(t) \geq T_h)$$

$$U[S^{(i)}(t)] = 0, (S^{(i)}(t) < T_h)$$

$$i = 0,1,2,\dots,5, \quad x = 1,2,3,\dots,352, \quad y = 1,2,3,\dots,288$$

This procedure was done at the end of each frame. Thus we get values of $C_{fs}^{(i)}(t)$ accordingly.

By taking a same procedure between $Y_f^{(i)}(x \pm 1, y \pm 1, t)$ and $Y_f^{(i)}(x, y, t)$ as

$$D_{ff}^{(i)}(t) = |Y_f(x \pm 1, y \pm 1, t) - Y_f(x, y, t)|, i=1,2,4,5$$

we can get $C_{ff}^{(i)}(t)$, those corresponding to $D_{ff}^{(i)}(t)$. This value leads edge information of image by averaging as

$$E_{ff}^{(i)}(t) = \frac{\sum_i D_{ff}^{(i)}(t)}{4}, \text{ where } i=1,2,4,5.$$

These values can induce angular and strength information of moving vector as follows

$$\Theta(t) = \tan^{-1} \left(\frac{C_{fs}^{(i)}(t)}{C_{fs}^{(j)}(t)} \right)$$

, where

$$i=1 (C_{fs}^{(1)}(t) \geq C_{fs}^{(5)}(t)),$$

$$i=0 (C_{fs}^{(1)}(t) < C_{fs}^{(5)}(t)),$$

$$j=1 (C_{fs}^{(4)}(t) \geq C_{fs}^{(2)}(t)),$$

$$j=0 (C_{fs}^{(4)}(t) < C_{fs}^{(2)}(t)).$$

And the strength

$$A(t) = \frac{D_{fs}^{(3)}(t)}{E_{ff}(t)}$$

Finally we got a moving vector $M_{fs}(u_{fs}, v_{fs})$

$$u_{fs}(t) = A(t) \cdot \cos(\Theta(t))$$

$$v_{fs}(t) = A(t) \cdot \sin(\Theta(t))$$

These values are to be used in attitude calculation circuit to derive yaw, roll, and pitch of the object.

4.2. Position of known markers on the ground

The fourth CMOS camera, which is gazing to the ground, derives two marker's positions on the ground. These markers positions are to be used to calculate a position (x, y, z) and absolute yaw value of the flying object on a x,y,z-coordinate of the ground.

In this experiment, the camera discriminate markers observing its color and brightness. The two vivid markers are colored in red and blue, and landed on the ground with a certain distance each other. A visual processing circuit, which is implemented in FPGA, derives those positions

$$\vec{p}_a = (p_{a1}, p_{a2}, 0)$$

$$\vec{p}_b = (p_{b1}, p_{b2}, 0)$$

at 25msec as frame rate.

5. ATTITUDE CALCULATION

Attitude calculation module, which consists of a FPGA, derives a location $(p_x^{(0)}, p_y^{(0)}, p_z^{(0)})$, attitude (*Yaw, Roll, Pitch*) of the flying object. Derived locations of markers above are expressed as $\vec{p}_a = (p_{a1}, p_{a2}, 0)$ and $\vec{p}_b = (p_{b1}, p_{b2}, 0)$. If a assumption that the object's attitude remains horizontally, easy translation gives us the location $p_x^{(0)}, p_y^{(0)}$, which is in xyz-coordinate on the ground, as follows

$$p_x^{(0)} = \frac{1}{2}(p_{a1} + p_{b1})$$

$$p_y^{(0)} = \frac{1}{2}(p_{a2} + p_{b2}).$$

A height $p_z^{(0)}$ of the flying object can be expressed as

$$p_z^{(0)} = \frac{\cos \theta}{\sin \theta} \cdot \frac{1}{2} \cdot |\vec{p}_b - \vec{p}_a|$$

, where θ is a FOV (field of view) corresponding a distance

$$\frac{1}{2} \cdot |\vec{p}_b - \vec{p}_a|.$$

As for a value of a height of the object, we used an abbreviated value

$$p_z = |\vec{p}_b - \vec{p}_a|$$

instead of $p_z^{(0)}$. Via this crude abbreviation, we can get a related value of actual height for controlling the object with only one camera. The efficiency of this value p_z is already tested at a previous experiment of Ref.[1]. Thus a location value (p_x, p_y, p_z) was had over to next stage as adaptive control unit, where notations are slightly changed for convenience below as $p_x = p_x^{(0)}$, $p_y = p_y^{(0)}$.

The vector \bar{p}_a and \bar{p}_b can also derive an absolute value of yaw as

$$Yaw_{abs} = \tan^{-1} \left(\frac{p_{b2} - p_{a2}}{p_{b1} - p_{a1}} \right).$$

This value is also handed over to the next control unit. The values (p_x, p_y) is actually calculated in FPGA at 40FPS, and p_z and Yaw_{abs} were derived by taking data tables inside FPGA, which size is 2048 word as 32x32 matrix for each values.

Moving vectors $M_{fs}(u_{fs}, v_{fs})$ and number of edges $E_{ff}(t)$ mentioned above create $(Yaw, Roll, Pitch)$ according to following treatment. Yaw has more precise time resolution than Yaw_{abs} , so the Yaw is used to stabilize yaw motion as derivative value in gyro control and Yaw_{abs} is used for attitude control.

Moving vector $M_{fs}(u_{fs}, v_{fs})$ represent moving of whole area of images, which means the notation x and y is $x = 1, \dots, 352$, $y = 1, \dots, 288$.

Each image of CMOS camera is divided into 9 areas that is,

Horizontally: $x = 1, \dots, 117 : 118, \dots, 234, 235, \dots, 352$

Vertically: $y = 1, \dots, 96 : 97, \dots, 192, 193, \dots, 288$.

So the moving vector is also expressed as matrix bellow

$$U^{(i)} = [u_{kl}^{(i)}], V^{(i)} = [v_{kl}^{(i)}]$$

$$i = 1, 2, 3, k = 1, 2, 3, l = 1, 2, 3$$

, where i is a notation of camera number, k and l are notations of areas in frames. For example, $U^{(1)} = [u_{kl}^{(1)}]$ is expanded as

$$U^{(1)} = \begin{bmatrix} u_{11}^{(1)} & u_{12}^{(1)} & u_{13}^{(1)} \\ u_{21}^{(1)} & u_{22}^{(1)} & u_{23}^{(1)} \\ u_{31}^{(1)} & u_{32}^{(1)} & u_{33}^{(1)} \end{bmatrix}.$$

In the same manner, edges $E_{ff}(t)$ is also expressed as matrix

$$E^{(i)} = [e_{kl}^{(i)}].$$

We induced a threshold level T_{edg} as certainty value of the area, Yaw can be finally calculated from the moving vector of horizontal direction $U^{(i)}$ as below

$$Yaw = \frac{\sum_{kli} u_{kl}^{(i)} \cdot \text{sgn}(e_{kl}^{(i)})}{k \cdot l \cdot i}$$

i.e.

$$\text{sgn}(x) = 1 \quad (x \geq T_{edg})$$

$$\text{sgn}(x) = 0 \quad (x < T_{edg}).$$

$Roll$ and $Pitch$ is also derived from vertical moving vectors $V^{(i)}$ and the edges $E^{(i)}$ as follows.

$$Roll = \text{Max}[\text{sgn}(e_{kl}^{(2)}) \cdot v_{kl}^{(2)} \oplus \text{sgn}(e_{mn}^{(3)}) \cdot v_{mn}^{(3)}]$$

,where the operator \oplus is

$$a \oplus b = 0 \quad (a \cdot b = 0)$$

$$a \oplus b = a + b \quad (a \cdot b \neq 0).$$

And the combination of k, l, m, n are limited as

$$(k, l, m, n) = (1, 1, 3, 3), (1, 2, 3, 2), (1, 3, 3, 1), (2, 1, 2, 3), (2, 2, 2, 2), (2, 3, 2, 1), (3, 1, 1, 3), (3, 2, 1, 2), (3, 3, 1, 1).$$

The operator $\text{Max}[a_{ij}]$ finds maximum value a_{ij} comparing in its absolute value $|a_{ij}|$.

The last value $Pitch$ is expressed in equation

$$Pitch = \text{Max}[\text{sgn}(e_{kl}^{(1)}) \cdot v_{kl}^{(1)}].$$

Thus finally attitude values

$$(p_x, p_y, p_z), Yaw_{abs}, (Yaw, Roll, Pitch)$$

were derived and to be passed over to adaptive control network.

6. MOTOR CONTROL

The flying object has four propellers and those motors. If the motor were labeled as counter clockwise manner $(Mot_1, Mot_2, Mot_3, Mot_4)$, (Mot_1, Mot_3) runs clockwise and (Mot_2, Mot_4) runs counter clockwise. Each power supplied to each motor are expressed

$$\vec{P} = (P_1, P_2, P_3, P_4).$$

For controlling derivatives of attitude and height of the object with these powers, we should give powers like following manners.

$$\frac{d}{dt}(Roll) : \vec{P} + \vec{R}, \quad \vec{R} = (0, r, 0, -r), \quad r: \text{signed value}$$

$$\frac{d}{dt}(Pitch) : \vec{P} + \vec{T}, \quad \vec{T} = (t, 0, -t, 0), \quad t: \text{signed value}$$

$$\frac{d}{dt}(Yaw) : \vec{P} + \vec{Y}, \quad \vec{Y} = (y, -y, y, -y), \quad y: \text{signed value}$$

$$\frac{d}{dt}(p_z) : \vec{P} + \vec{A}, \quad \vec{A} = (a, a, a, a), \quad a: \text{signed value}$$

These power modification vectors are not generally independent. For example, Roll and Height has slight

cross talk because a rolling motion decreases its lift force. This kind of cross talk is one of the hardest problems for controlling flying objects generally. If those parameters including the cross talk have ideal value, the kinetic equation can be solved analytically. So as to say, it is predictable. However there is some ambiguity caused by variances, for example, friction of motor unit, distortion of body and so on, the system can not be predictable with parameters mentioned above. So this kind of problem should be solved in adaptive control neural network mentioned below section.

7. ADAPTIVE CONTROL

Adaptive control unit will finally give motor output according to values derived above. Calculation of this section was performed on conventional CPU (Hitachi SH4) using software description. The control method is basically based on PID control, but some nonlinear functions are induced. To overcome the problems about cross talk, a neural network with back-propagation method is used.

Parameters of describe present situation are as follows, those are derived above sections.

$$\vec{s} = \begin{pmatrix} \text{Roll} \\ \text{Pitch} \\ \text{Yaw} \\ \text{Yaw}_{abs}^d \\ p_x^d \\ p_y^d \\ p_z^d \end{pmatrix} = [s_i]$$

, where

$$\begin{aligned} \text{Yaw}_{abs}^d &= \text{Yaw}_{abs}^{obs} - \text{Yaw}_{abs} \\ ,(\text{Yaw}_{abs}^{obs} : \text{objective value of } \text{Yaw}_{abs}) \\ p_x^d &= p_x^{obs} - p_x \\ , (p_x^{obs} : \text{objective value of } p_x) \\ p_y^d &= p_y^{obs} - p_y \\ , (p_y^{obs} : \text{objective value of } p_y) \\ p_z^d &= p_z^{obs} - p_z \\ , (p_z^{obs} : \text{objective value of } p_z) \end{aligned}$$

This vector should transform as

$$\vec{s}^{(1)} = A^{(1)} \vec{s}$$

Here the $A^{(1)}$ is expressed below.

$$A^{(1)} = \begin{bmatrix} g_{11} & w_{11} & w_{12} & w_{13} & g_{12} & 0 & 0 \\ w_{21} & g_{11} & w_{22} & w_{23} & 0 & g_{12} & 0 \\ w_{31} & w_{32} & g_{31} & g_{32} & 0 & 0 & 0 \\ 0 & 0 & 0 & g_{41} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{71} & w_{72} & w_{73} & w_{74} & w_{75} & w_{76} & g_{71} \end{bmatrix}$$

g_{ij} are gains of each parameter, and expanded as

$$g_{ij} = g_{ij}^{(0)} + g_{ij}^{(1)}, i=1,2,3,4,7, j=1,2,3.$$

$g_{ij}^{(0)}$ are initial gains, those are settled by manually, and

$g_{ij}^{(1)}$ are additional gains those change automatically according to error function mentioned below.

Offsets operator should also added as

$$\vec{s}^{(2)} = A^{(2)} + \vec{s}^{(2)}$$

,where $A^{(2)}$ is

$$A^{(2)} = [d_{kl}^{(0)} + d_{kl}^{(1)}], k=1,2,3,\dots,7, l=1,2,3,\dots,7.$$

$d_{kl}^{(0)}$ are initial fixed values and $d_{kl}^{(1)}$ are changeable values with a learning.

$W = [w_{kl}]$ is a weight matrix as cross talk neural network between each element of vector \vec{s} . This weight matrix changes randomly and fixes when corresponding error elements $\varepsilon_m^{(0)}$ (mentioned below) should be under a condition $\varepsilon_m^{(0)} < t_h$, where t_h is an initially fixed threshold level. In the same manner, gains $g_{ij}^{(1)}$ and offsets $d_{kl}^{(1)}$ change randomly. And W , $g_{ij}^{(1)}$, $d_{kl}^{(1)}$ should be zero at condition of

$$|\bar{\varepsilon}^{(1)}| \geq T_{zero} \text{ or } |\vec{s}| \geq T_{zero},$$

, where T_{zero} is a threshold level for learning. The absolute value $|w_{kl}|$ is limited under T_w . $|g_{ij}^{(1)}|$ and $|d_{ij}^{(1)}|$ are also limited under T_{gd} . Each T_w and T_{gd} is under 10% of maximum value of $|s_i|$.

Error vectors $\bar{\varepsilon}^{(n)}$ are expressed as follows.

$$\bar{\varepsilon}^{(0)} = \vec{s}^{(2)} - \vec{s} = [\varepsilon_m^{(0)}], m=1,2,3,\dots,7$$

$$\bar{\varepsilon}^{(1)} = \int_{t-t_d}^t \vec{s}(t) dt = [\varepsilon_m^{(1)}]$$

, where a t_d is a proper integration time (fixed value).

An emergency operator was added to above transformation as

$$\text{Emg}[\vec{s}^{(2)}]$$

, where

$$\text{Emg}[\vec{a}] = \vec{a}, (|\bar{\varepsilon}^{(1)}| < T_{emg} \text{ or } |\vec{s}| < T_{emg})$$

$$Emg[\bar{a}] = 0, (|\bar{\varepsilon}^{(1)}| \geq T_{emg} \text{ or } |\bar{s}| \geq T_{emg}).$$

T_{emg} is a threshold value of emergency to avoid bigger damage to body.

Finally input parameters \bar{s} was transformed to

$$\bar{s}^{(3)} = Emg[A^{(2)} + A^{(1)}\bar{s}] = Emg[A^{(2)}] + Emg[A^{(1)}\bar{s}]$$

$$\bar{p} = Emg[A^{(2)}] = [p_i]$$

$$\bar{q} = Emg[A^{(1)}\bar{s}] = [q_i]$$

The output values for four motors are

$$\bar{P} = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = \bar{p} + q_1\bar{R} + q_2\bar{T} + q^{(34)}\bar{Y} + q_7\bar{A}$$

, where

$$q^{(34)} = q_3, |q_4| < Tq,$$

$$q^{(34)} = q_4, |q_4| \geq Tq,$$

$$\bar{R} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix}, \bar{T} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}, \bar{Y} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}, \bar{A} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Tq is a threshold level of yaw movement.

8. CONCLUSION

In this paper, a flying object with 4 CMOS camera and 4 propellers on cross-shaped carbon chassis was used to study an original image processing and the kinetic control method. Two types of image processing was actually implemented on FPGA and verified with a attitude calculation module, which is also implemented on a FPGA. The method is new and useful for this kind of control experiment. It has overcome a problem lies between sampling time and motions, which is essentially important to mimic biological eyes.

The kinetic control method as adaptive learning control is still now tested in various types. We presented one of them in this paper.

9. ACKNOWLEDGEMENT

We would like to express appreciate to Dr. Takashi Tominaga for fruitful discussions on this manuscript.

10. REFERENCES

- [1] A flying robot controlled by a biologically inspired vision system: M. Ichikawa, H. Yamada, and J. Takeuchi, Proc. ICONIP 2001, p677-680
- [2] Fly-like visuomotor responses of a robot using aVLSI motion-sensitive chips: Shih-Chii Liu, Alessandro Usseglio-Viretta, Biol. Cybern. 85, 449-457 (2001)