PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 43, NO. 4, PP. 251–261 (1999)

HYBRID POSITION AND FORCE CONTROL ALGORITHM EXPANSION OF A ROBOT CONTROL SYSTEM

Gergely FODOR* and Gábor TEVESZ**

*WÖHRLE Hungária Kft.
H–1116 Budapest, Mohai út 30/b, Hungary
Phone: (+36-1) 204-3060, Fax: (+36-1) 204-3054
e-mail: mail@woehrle.hu
**Budapest University of Technology and Economics
Department of Automation and Applied Informatics
H-1111 Budapest, Goldmann György tér 3. IV. 433, Hungary
Phone: (+36-1) 463-2870, Fax: (+36-1) 463-2871
e-mail: tevesz@aut.bme.hu

Received: April 27, 2000

Abstract

At the Technical University of Budapest within the scope of a project supported by the Hungarian Research Fund (OTKA, grant No. T029072) an experimental robot control system has been developed. Last year a new component, a six-axis force/torque sensor and the related control unit were added to the system, allowing to implement the hybrid position and force control algorithm. The purpose of this study is to describe the new hardware component and its tasks within the control system. The paper presents an overview of the control algorithm and summarises the architecture of the experimental robot control and the force/torque sensor systems. Then the architecture of the hardware designed and its computing task are outlined. Finally, estimation for the cycle time required by the calculations is given.

Keywords: robot control, hybrid position and force control, force/torque sensor.

1. Introduction

Since a detailed description of the hybrid position and force control algorithm and its implementation by the robot control system can be found in [1] [4], here just the main thoughts are discussed which are necessary for the further understanding. Advanced robot control algorithms are based on the non-linear dynamic model of the robot:

$$H(q)\ddot{q} + h_{ccfg}(q, \dot{q}) = H(q)\ddot{q} + h_{ccf}(q, \dot{q}) + h_g(q) = \tau, \qquad (1)$$

where H, h_{ccf} and h_g are the generalised inertia, and the centripetal, Coriolis, friction forces, and the gravity effect, respectively.

The computation of the driving torque (τ) requires remarkable computing power with a cycle time of 1–10 msec, therefore a multiprocessor architecture,

referred to as the experimental robot control system implements the control algorithms. One of the most powerful control algorithms is the hybrid position and force control.

2. The Hybrid Position and Force Control

The basic idea of the hybrid position and force control [2] is to use the feedback of a six-component force/torque sensor in the control algorithm. The algorithm is based on the kinetics of the robot given in the operating space derived from (1):

$$H^*(x)\ddot{x} + h^*_{ccf}(x, \dot{x}) + h^*_g(x) = F, \tag{2a}$$

$$\tau = J^T F, \tag{2b}$$

where J is the Jacobian of the robot.

Applying non-linear dynamic decoupling:

$$F = F_{\text{motion}} + F_{\text{active}} + F_{ccfg}, \tag{3a}$$

where F_{motion}, F_{active}, F_{ccfg} are the forces in the operating space representing the motion, the active force control, and the summarised effect of the centripetal force, Coriolis force, friction and gravity components, respectively. Let us choose the components of F in the following forms:

$$F_{\text{motion}} = \hat{H}^*(x) S u_{\text{motion}}, \tag{3b}$$

$$F_{\text{active}} = \tilde{S} u_{\text{active}} + \hat{H}^*(x) \tilde{S} u_{\text{damp}}, \tag{3c}$$

$$F_{ccfg} = \hat{H}^*_{ccf}(x, \dot{x}) + \hat{h}_g(x), \tag{3d}$$

where $\hat{H}, \hat{h}_{ccf}$ etc. denote the values of the nominal robot model, and U_{motion}, U_{active}, U_{damp} are outputs of a PID, a PI and a P controller (see *Fig. 1*). S and $\tilde{S}$ are the so-called general task specification matrices. S defines the possible directions of motion of the robot's end effector in K_0 (reference frame of the robot), while $\tilde{S}$ defines the directions of force and torque to be exerted. The matrices can be constants, can change with the configuration or can continuously change in time. Finally, the equation of the hybrid position and force control is:

$$\tau = J^T(q) \left\{ \hat{H}^*(q)[S u_{\text{motion}} + \tilde{S} u_{\text{damp}}] + \tilde{S} u_{\text{active}} \right\} + \\ + \hat{h}_{ccf}(q, \dot{q}) + \hat{h}_g(q) - J^T \hat{H}^*(q) \dot{J} \dot{q}. \tag{4}$$

The architecture of the robot control algorithm based the above equation is shown in *Fig. 1*.

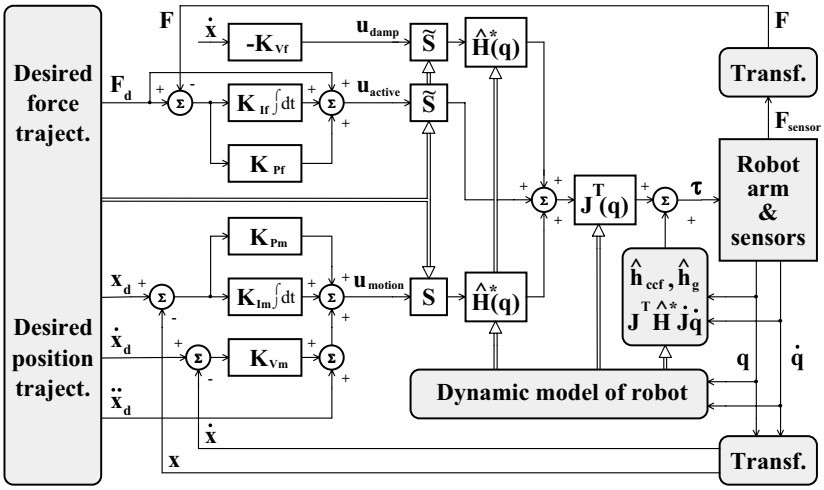

Fig. 1. Principle of the hybrid position and force control

3. The Transformation of the Force/Torque Vector

The errors in the hybrid position and force control are calculated in K_0, therefore the generalised force/torque vector (F in *Fig. 1*) has to be transformed from K_S (frame of the transducer) to K_0 [2]. Let $T_{S,0}$ be the homogeneous transformation from K_S to K_0:

$$T_{S,0} = \begin{bmatrix} A_{S,0} & p_{S,0} \\ 0^T & 1 \end{bmatrix}. \tag{5a}$$

If f_S and n_S are the measured force and torque vectors in K_S then:

$$f_0 = A_{S,0}^T f_S, \tag{5b}$$

$$n_0 = A_{S,0}^T (n_S + f_S \times p_{S,0}). \tag{5c}$$

Or in another form:

$$\begin{pmatrix} f_0 \\ n_0 \end{pmatrix} = \begin{bmatrix} A_{S,0}^T & 0 \\ ([p_{S,0}\times]A_{S,0})^T & A_{S,0}^T \end{bmatrix} \begin{pmatrix} f_S \\ n_S \end{pmatrix}, \tag{5d}$$

where $[p_{S,0}\times]$ is the matrix of vector product:

$$[p_{S,0}\times] \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_z & 0 \end{bmatrix}. \tag{5e}$$

Since $T_{S,0}$ depends on the current position of the robot's end effector (or with other words depends on the joint variables) the matrix has to be calculated in real time.

4. Architecture of the Experimental Robot Controller

The experimental robot controller [1] was developed for a six degree of freedom NOKIA-PUMA 560 robot arm. The architecture of the controller system can be seen in *Fig. 2*.

The system is based on an IBM compatible PC. Depending on the computational demand, the realisation of the control algorithm is performed by three or four home developed so-called ARC (Advanced Robot Controller) cards. Each ARC card realises a two-processor system containing a communication preprocessing unit (i386EX) and a signal processor (TMS320C31).

Each ARC card has two main communication channels. The first one is an 8/16-bit connection toward the host via the ISA bus. The second one is a standardised CAN (Controller Area Network) based serial link. This multi-master bus allows a non-synchronised connection between the cards to be as fast as 1 Mbaud. As it will be described in Section 7 the transformation of the force/torque

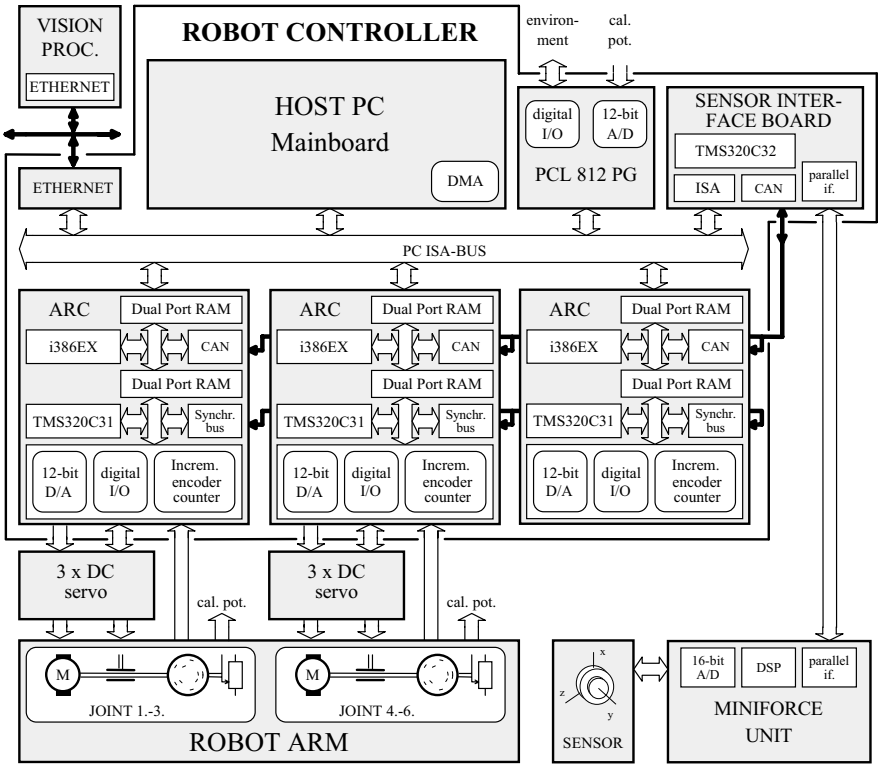

Fig. 2. Architecture of the experimental robot controller

vector requires large computing capacity, that is why it is realised as a separate card called ‘Sensor Interface Board’. This block is the main subject of this paper. The description of the six-axis force torque sensor can be found in the next section.

The most important communication channel on the lowest level of the control system is the standardised CAN bus. The aimed cycle time of the control algorithm is approximately 1 msec, the realised communication channels have to satisfy this demand. The information forwarded on this bus is summarised in *Table 1* [1].

Table 1. Data transfer on the CAN bus

Parameter	Data bytes	Compressed size [byte]	Transmission method [byte]	Transmission time [μsec]
q	6×4	16	2×8	216
$\dot{q}$	6×4	16	2×8	216
f	6×4	24	3×8	324
τ	6×2	12	2×6	184
			Total:	940

5. The Six-Axis Force Torque Sensor

A sensor fixed between the robot’s last joint and the end effector is capable to measure the arising forces and torques in the end effector. The sensor consists of a metal spring and several electrical conductors whose resistance change with deformation (strain gages). The strain gages are normally applied in multiplies of four and wired into a Wheatstone bridge configuration. If the bridges are excited with a fixed voltage source the output voltage of the bridges are proportional to the applied force/torque. With the appropriate number of bridge circuits (six) the force and torque values can be measured in three dimensions.

The controlling electronics of the sensor excites the bridge circuits and processes the analogue bridge outputs signals. This unit – called ‘MiniForce’ [3] – performs the analogue-digital conversion of the bridge signals and makes some preprocessing steps like digital filtering and matrix compensation, which eliminates the crosstalk effect between the bridge circuits. Finally, it passes the result via a 16-bit parallel communication channel (RS422 differential lines).

6. The Sensor Interface Board

On the basis of the previous sections the tasks of the sensor interface board can be outlined. The board should:

1. Read the actual force/torque vector (F) from the MiniForce unit

2. Receive the actual joint values (q) via the CAN bus
3. Transform the force/torque vector to frame K_0 according to the current position of the robot's end effector determined by the joint values
4. Send the transformed vector via the CAN bus to the ARC cards, which use it in the control algorithm.

The above tasks and the hardware environment outline the architecture of the sensor interface board [5]. The scheme of the interface board is shown in *Fig. 3*.

The board:

1. Is connected to the host PC ISA bus
2. Controls the MiniForce unit
3. Contains a digital signal processor (DSP) to perform the transformation in real time
4. Is able to communicate via the CAN bus with the ARC cards.

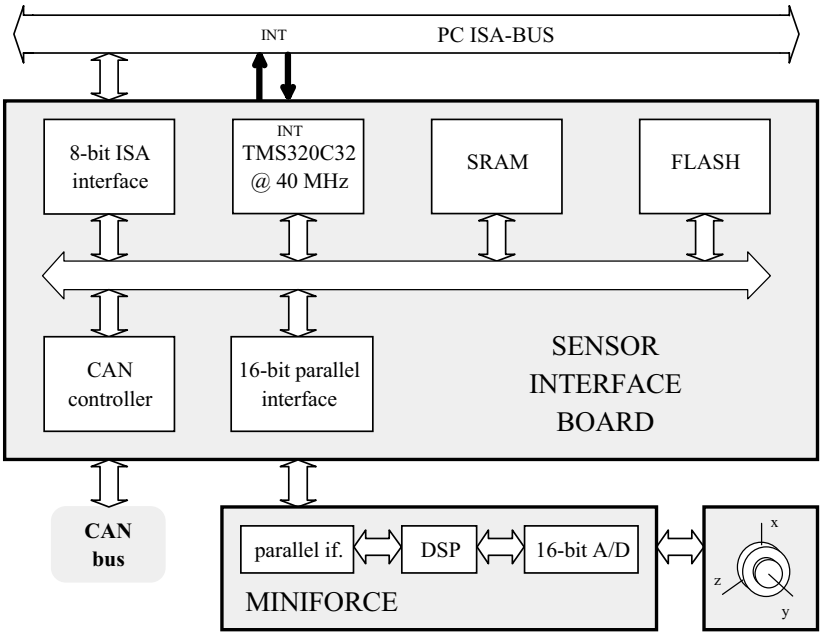

Fig. 3. The sensor interface board

As it was mentioned earlier, the necessary communication for the control algorithm takes place via the CAN bus. The host PC uses the ISA bus of the sensor interface board only for configuration and to download the software of the signal

processor. Because of the comparatively low dataflux, the board uses only the 8-bit ISA bus. The communication is interrupt driven on both sides.

The board is connected to the MiniForce through a parallel interface. The physical interface consists of 24 RS422 lines (16 data and 8 command lines). In spite of the MiniForce unit expects normal processor read/write access cycles on these lines, the read/write cycle sequence is simulated by software through three 8-bit registers. Two bidirectional registers store the 16-bit data, and one register provides the controlling signals (like chip select or address lines).

As the transformation algorithm requires high speed real-time computation, the interface board has been realised with a signal processor. This processor – following the type of the signal processor of the ARC cards – is a TMS320C32 @40MHz. The software is stored in a slow non-volatile flash memory from which the signal processor copies the program code to its internal memory, or to the external zero wait state static RAM.

A stand-alone CAN controller (Intel 82527) provides the communication via the CAN bus.

7. Realisation of the Transformation Algorithm

As described in Section 3 the force and torque vectors measured by the sensor have to be transformed to the reference frame (K_0) of the robot. The matrix $T_{S,0}$ (see *Eq.* (5a)) can be determined on the basis of the robot graph.

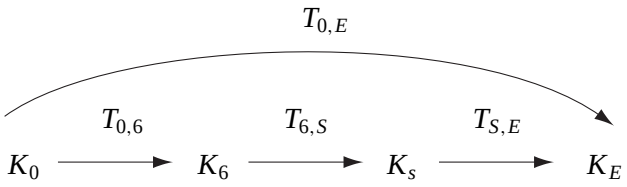

Fig. 4. The robot graph

$$T_{S,0} = T_{S,E} \cdot T_{0,E}^{-1}, \tag{6}$$

where:

T: homogeneous transformations
K_0: reference frame
K_6: frame of the last joint
K_s: frame of the sensor
K_E: frame of the end effector.

Since the MiniForce unit is able to perform the $T_{S,E}$ transformation as a preprocessing step (it does not depend on the joint variables) only $T_{0,E}^{-1}$ has to be

calculated. Calculating $T_{0,E}$ means solving the direct geometry task, which is in the case of a PUMA560 robot arm (the equations can be found in [2]):

$$T_{0,6} = \begin{bmatrix} l_x & m_x & n_x & p_x \\ l_y & m_y & n_y & p_y \\ l_z & m_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

For example:

$$I_x = C_1C_{23}(C_4C_5C_6 - S_4S_6) - S_1(S_4C_5C_6 + C_4S_6) - C_1S_{23}S_5C_6,$$
$$\ldots$$
$$p_z = -S_2a_2 + C_{23}d_4,$$

$$T_{0,E} = T_{0,6}T_{6,E} = T_{0,6}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_E \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} l_x & m_x & n_x & p_x + d_En_x \\ l_y & m_y & n_y & p_y + d_En_y \\ l_z & m_z & n_z & p_z + d_En_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

C_X/C_{XY} and S_X/S_{XY} are the markings of the $\cos(x)/\cos(x+y)$ and $\sin(x)/\sin(x+y)$ functions where the indices represent the actual values of the respective joints. Further on d_E is one of the robot's physical parameters.

In the equations first $C_1, C_2, C_{23}, C_4, C_5, C_6, S_1, S_2, S_{23}, S_4, S_5, S_6$ have to be calculated and stored. According to the math library of the C compiler of the TMS320C3x signal processor family, the sine and cosine functions consist of approximately 40 instructions which takes 2 μs. (Notice: in the calculations we expect that an instruction can be executed in 50 ns (fastest execution). This requires the suitable placing of instructions and data in the internal and external memories.)

It is worth to calculate some quantities in advance as they are used several times in the equations. For example: $C_4C_5C_6 - S_4S_6$ or $C_4C_5S_6 + S_4C_6$.

Summing up the required computing capacity of the direct geometry task with the optimized calculation is as follows:

Table 2. Required computing capacity of the direct geometry task

Function	One operation [ns]	Number of operations	Total [μs]
Sine function	2000	6	12
Cosine function	2000	6	12
Addition/subtraction	100	27	2.7
Multiplication	100	72	7.2
		Total:	33.9

After calculating the direct geometry task, $T_{0,E}$ has to be inverted. Instead of using the general matrix inversion formula $T_{0,E}^{-1}$ can be calculated as follows [4]: Since the orientation part A is orthonormal, we can calculate its inverse by taking its transpose A^T. Based on the definition of the inverse matrix ($T_{0,E}\,T_{0,E}^{-1} = I$):

$$T_{0,E}^{-1} = \begin{bmatrix} A_{0,E} & p_{0,E} \\ 0^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} A_{0,E}^T & -A_{0,E}^T\,p_{0,E} \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} & & -l_{0,E}\cdot p_{0,E} \\ A_{0,E}^T & & -m_{0,E}\cdot p_{0,E} \\ & & -n_{0,E}\cdot p_{0,E} \\ 0^T & & 1 \end{bmatrix}, \tag{9}$$

where $a \cdot b$ denotes the scalar product of vectors a and b.

Taking into consideration that the transpose of a matrix does not require any extra cycles:

Table 3. Required capacity for computing the inverse of $T_{0,E}$

Function	One operation [ns]	Number of operations	Total [μs]
Addition/subtraction	100	6	0.6
Multiplication	100	9	0.9
		Total:	1.5

(It is worth to mention that using the general inversion formula – which requires floating point division – the calculation of $T_{0,E}^{-1}$ would take approximately 36 μs.)

$T_{0,E}^{-1}$ can be written in the following form:

$$T_{0,E}^{-1} = \begin{bmatrix} A_{E,0} & p_{E,0} \\ 0^T & 1 \end{bmatrix}. \tag{10}$$

Finally, the force and torque vectors have to be transformed according to *Eq.* (5c). (Remark: As it was mentioned the MiniForce unit can perform the $T_{S,E}$ transformation, therefore $A_{E,0}$ and $p_{E,0}$ can be written instead of $A_{S,0}$ and $p_{S,0}$.)

The required computing capacity of the transformation is summarised below:

Besides the transformation algorithm the communication time also has to be taken into consideration.

The communication with the MiniForce unit takes place by means of packages. Since the cycle time of the controlling algorithm is given, the so-called ‘asynchronous’ mode is used, which means that the force/torque vectors are automatically updated reducing the communication time.

Table 4. Required computing capacity of the transformation algorithm

Function	One operation [ns]	Number of operations	Total [μs]
Addition/subtraction	100	30	3.0
Multiplication	100	45	4.5
		Total:	7.5

The package containing the force/torque vectors can be read word (16 bit) by word. The user can select the force/torque data format. In our case the 16-bit raw data format (the direct sensor input) is the most suitable because calculating the force/torque values from the sensor input is faster than reading them in 32-bit floating point or fractional format. As it was described in section 6, instead of using the read/write cycle of the processor the sequence is built up from several instructions. The 16-bit read/write cycle subroutines consist of 20 instructions. The package itself – containing the six vectors and some additional information – is 10 word long, thus reading the force/torque data can be accomplished in 10 μs. After transforming the force/torque vectors they have to be sent on the CAN bus to the ARC cards which realise the control algorithm. The controller completely solves the communication, only the package data – the transformed vectors – have to be written. As compared to the transformation algorithm and the communication with the MiniForce unit it takes insignificant time. The reception of the new joint variables can take place parallel to the transformation algorithm. Summing up the functions and their computing capacity:

Table 5. The estimated computing capacity

Function	Total [μs]
Direct geometry task	33.9
$T_{0,E}^{-1}$	1.5
Force/torque transformation	7.5
Read force/torque vectors	10.0
Total:	52.9

In reality, because of the need of additional processor instructions and communication overhead and wait cycles we can estimate the computation time 50–100% longer than the computed above.

8. Conclusions

The sensor interface board is a key element of the experimental robot control system if hybrid position and force control is applied as the controlling algorithm. It relieves the ARC cards of the time demanding force/torque transformation process. As it was presented the time the transformation algorithm takes is well within the estimated time of the control algorithm (1 ms). It means that the algorithm can be surely implemented in high level C language.

To accelerate the development a communication system between the host PC and the interface board has been realised. The software based on the presented algorithm is under development.

References

[1] TEVESZ, G., Architectural Problems of the Hybrid Position and Force Control System of Robots. *Periodica Polytechnica Ser. El. Eng.* **42** (1998), No. 2.

[2] LANTOS, B., *Robotok irányítása* (Robot control), 2nd edition. Akadémiai Kiadó (in Hungarian), 1997.

[3] Cortex Kft., MiniForce 6-axis Force-Torque Intelligent Sensor System. Cortex Kft. 1995.

[4] SOMLÓ, J. – LANTOS, B. – CAT, P. T., *Advanced Robot Contro*l. Hungarian Academic Press, 1997.

[5] FODOR, G., Hatkomponensű erő-nyomatékmérő berendezés illesztése kísérleti robotirányító rendszerhez (Interfacing a Six-axis Force/Torque Sensor to an Experimental Robot Controller). Diploma work. TUB Dep. of Automation (in Hungarian), 1999.