LICOR:
Uma linguagem de programação de robôs

Marco Antonio Busetti de Paula (*)

RESUMO

O presente trabalho descreve o desenvolvimento de uma linguagem de programação de robôs denominada LICOR.

A linguagem consiste em uma biblioteca de sub-rotinas específicas para manipuladores, associada a uma linguagem popular de alto nível, a linguagem PASCAL.

A finalidade da linguagem LICOR é permitir que o usuário não se preocupe com os detalhes do controle de baixo nível dos robôs. Assim sendo, LICOR é uma base para o desenvolvimento hierárquico de sistemas mais complexos que incluem o nível de especificação de movimentos no espaço de trabalho, envolvendo os sensores externos, e mais acima do nível de descrição de tarefas de maneira mais simbólica.

O objetivo desta linguagem é programar movimentos através das coordenadas de posição dos atuadores, associadas a um tempo. Os pontos programados são interpolados por uma curva suave que serve de referência para o movimento dos vários atuadores, compondo então o movimento do robô.

Utilizou-se um manipulador experimental para o teste da linguagem. Este manipulador possui coordenadas rotativas com 5 graus de liberdade e uma garra como dispositivo de extremidade. Os atuadores são motores de corrente contínua que possuem codificadores incrementais com sensores de posição. A parte eletrônica de controle foi implementada baseada em um microcomputador compatível com o IBM-PC.

As características mecânicas, elétricas, eletromecânicas e a eletrônica associada ao manipulador estão descritas no trabalho.

Exemplos típicos de programas em PASCAL utilizando a biblioteca LICOR, dirigidos à telemanipulação e a aplicações industriais, foram implementados, mostrando a aplicabilidade da linguagem.

Alguns testes experimentais foram realizados com o manipulador, tais como a resposta à trajetória de referência, como o degrau, a rampa e senóides. Além disto foram realizados testes de manipulação de objetos.

ABSTRACT

This work describes the design, development and tests of a robot control language called LICOR.

This language is implemented as a library of robot specific subroutines for a popular high level language, the Language PASCAL.

The aim of LICOR is to allow the programmer to execute tasks in robotics using only PASCAL and these subroutines, without worrying about details of the robot's low-level control.

An experimental manipulator was used to conduct tests of the language. Mechanical, electrical, electromechanical and electronics characteristics of this manipulator were also described.

Typical programs for telemanipulation and industrial automation were showed as examples, and the results were also showed and analyzed.

(*) Marco Antonio Busetti de Paula, *é Professor do Departamento Acadêmico de Eletrotécnica do CEFET-PR; Mestre em Engenharia Elétrica pela COPPE-UFRJ; Doutorando pela Universität Paderborn.*

1. INTRODUÇÃO

Com a crescente evolução de automação industrial, notamos com mais freqüência o uso de robôs industriais em linhas de manufaturas e em células flexíveis, como por exemplo na indústria automobilística. Conseqüentemente, surgiram várias linguagens de programação de robôs para diferentes finalidades.

Podemos citar, como exemplo, a linguagem VAL, desenvolvida pela Unimation, Inc. para o controle de robô PUMA. Este é um exemplo típico de uma linguagem específica para um determinado robô. Entretanto, existem outras linguagens que são complementos de linguagens computacionais já existentes. Entre estas, podemos citar as linguagens "AR-BASIC" e "ROBOT-BASIC", bibliotecas da linguagem BASIC e a "JARS", biblioteca da linguagem PASCAL. Geralmente as linguagens de programação de robôs possuem comandos específicos ao movimento do dispositivo de extremidade.

Este trabalho descreve o desenvolvimento de uma linguagem de programação de robôs, denominada LICOR. O nome escolhido para esta linguagem, LICOR, foi obtido pela contração da expressão: Linguagem de Controle de Robôs.

A linguagem consiste em uma biblioteca de sub-rotinas específicas associada a uma linguagem popular de alto nível — PASCAL, para a realização de tarefas com robôs.

A finalidade da linguagem LICOR é permitir que o usuário não se preocupe com os fatores inerentes ao controle de baixo nível dos robôs.

O objetivo desta linguagem é programar movimentos através de coordenadas de posição, associadas a um tempo, dos atuadores do manipulador. A trajetória a ser realizada entre os pontos programados é interpolada a esta curva e servirá de referência para o movimento.

Para a verificação experimental da linguagem LICOR, utilizamos um manipulador mecânico, uma parte eletrônica e um programa de interface. Este sistema é esquematizado na figura 1.

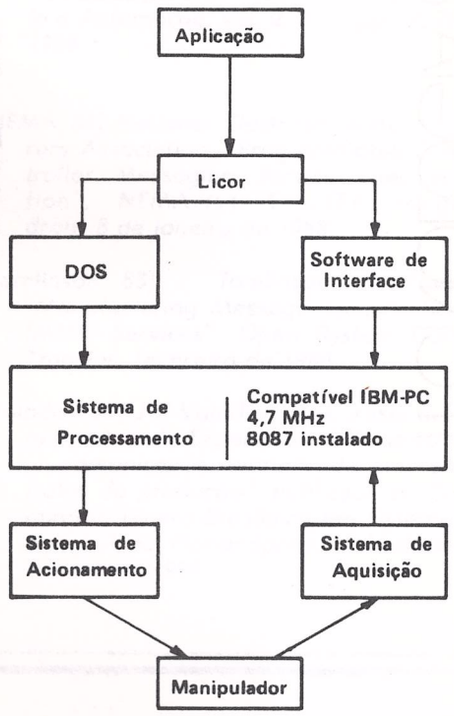


FIG. 1 — Diagrama de bloco de um Sistema Usuário de LICOR.

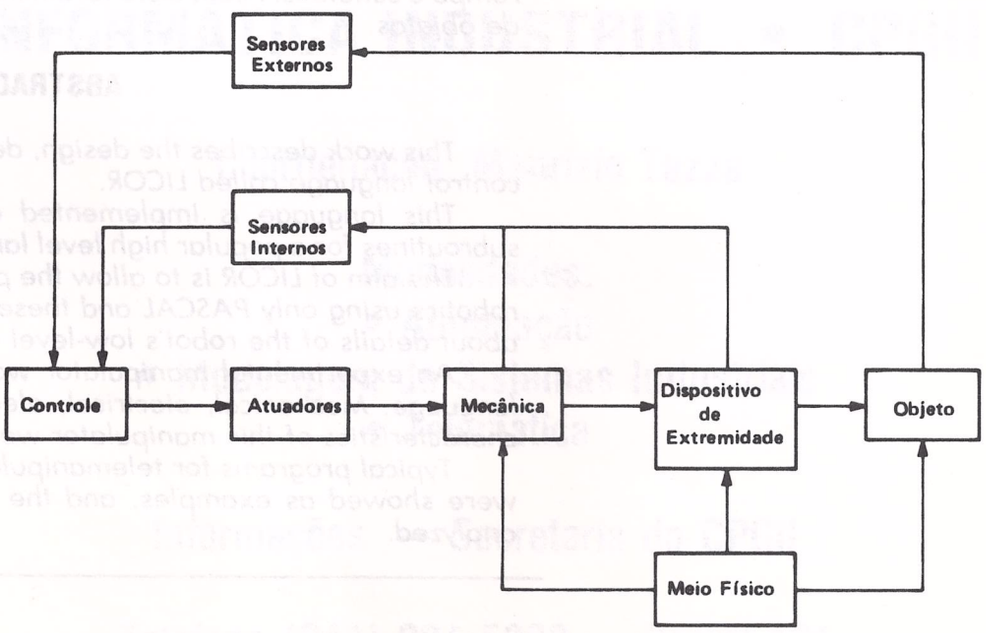


FIG. 2 — A estrutura do sistema do robô industrial.

Segundo o RIA (Robot Institute of America), um robô industrial é um manipulador reprogramável multifuncional projetado para mover materiais, ferramentas ou dispositivos especiais, através de movimentos programados para a execução de uma variedade de tarefas.

AMARAL [1] e CRITCHLOW [2] descrevem um manipulador como uma estrutura mecânica, composta de engrenagens, elementos de transmissão e acionadores, possuindo graus de liberdade suficientes para a execução de tarefas destinadas ao robô.

A figura 2 mostrada por KREUZER e TRUCKENTRODT [3] descreve o sistema de um robô industrial e ilustra a estrutura do sistema de um robô industrial, caracterizando-o como um conjunto de subsistemas. Ao controle estão associados o software e parte do hardware eletrônico, constituindo o coração do sistema. Os atuadores são responsáveis pelo fornecimento da potência mecânica ao manipulador e os sensores pela realimentação de parâmetros físicos ao controle, tais como posição, velocidade e torque entre outros. Os sensores externos percebem a interação do objeto e do manipulador como o meio físico. A mecânica consiste na estrutura propriamente dita associada às engrenagens e elementos de transmissão. O dispositivo de extremidade, como garra, aparelho de solda, ferramenta entre outros, é específico da tarefa a ser realizada pelo manipulador.

A figura 3 mostra o manipulador experimental utilizado ao lado do sistema de acionamento dos atuadores.

2. CLASSIFICAÇÃO DAS LINGUAGENS DE PROGRAMAÇÃO DE ROBÔS

CRAIG [4] caracteriza as linguagens de programação de robôs em 3 tipos:

— por aprendizagem;
— explícitas para robôs;
— de tarefas.

A linguagem de programação por aprendizagem consiste em mover o robô através de uma caixa de aprendizagem ou de um "Joystick", descrevendo uma trajetória, que pode ser armazenada no sistema de processamento de duas formas:

a) Ponto a ponto, onde são memorizados apenas os pontos chaves do movimento, de forma que o caminho entre dois pontos consecutivos não é determinado, mas sim função do controle de robô.
b) Trajetória contínua, onde são memorizados pontos através da discretização da trajetória com uma freqüência de amostragem suficientemente alta, capaz de reconstituí-la posteriormente.

As linguagens de programação explícita para robôs podem ser divididas em três categorias:

a) Linguagens específicas de manipulação: caracterizadas pelo desenvolvimento de uma nova linguagem dirigida para aplicações específicas de robôs. São desenvolvidas especialmente para o controle de manipuladores sem a necessidade do suporte de uma

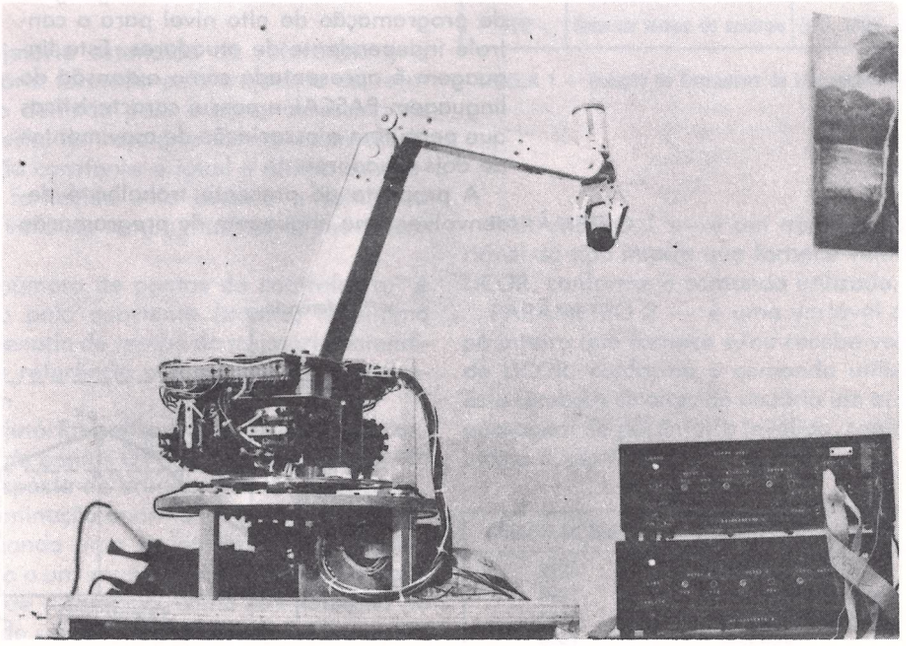


FIG. 3 — Visão geral do manipulador experimental.

linguagem de uso geral. Citamos como exemplos a linguagem VAL utilizada pela Unimation, Inc. e a linguagem AL desenvolvida pela Universidade de Stanford.

b) Biblioteca de comandos de robôs para uma linguagem computacional existente, que consiste em acrescentar a uma linguagem popular um pacote de sub-rotinas específicas. Isto permite ao usuário escrever programas em uma linguagem comum, como por exemplo Pascal ou Basic, fazendo uso de chamadas freqüentes das sub-rotinas pré-definidas, a fim de realizar a tarefa necessária. Como exemplos estão incluídos o AR-BASIC da American Robot Corporation e ROBOT-BASIC da Intelledex, ambos implementados como bibliotecas de linguagem Basic, e o JARS, desenvolvido pela NASA's Jet Propulsion Laboratory, baseado em Pascal.

c) Biblioteca de comandos de robôs para uma nova linguagem de uso geral, que consiste em criar uma linguagem suporte para uma biblioteca de sub-rotinas pré-definidas para um robô específico, como o caso da AML desenvolvida pela IBM e da RISE pela Silma, Inc.

A linguagem de programação de tarefa capacita o usuário a realizar tarefas pré-definidas que consideram, além do movimento do manipulador, o ambiente de trabalho a ele associado. Esta possibilita decisões de desvio de obstáculos, previsão de colisões e escolha automática do movimento ótimo.

TAYLOR et alii [5] descrevem que, em estudos realizados em programas de aplicação para robótica, uma larga porcentagem de linguagens não são específicas de robôs e sim caracterizadas como extensões de linguagens de programação de uso geral.

MESHKAT [6] descreve uma linguagem de programação de alto nível para o controle independente de atuadores. Esta linguagem é apresentada como extensão da linguagem PASCAL e possui características que permitem a associação de movimentos de dois atuadores.

A proposta do presente trabalho é desenvolver uma linguagem de programação explícita para robôs denominada LICOR, através de uma biblioteca de sub-rotinas específicas de uma linguagem popular de alto nível, a linguagem PASCAL.

O objetivo da linguagem LICOR é a programação de movimentos de robôs através de parâmetros de posição e tempo para cada atuador individualmente.

3. DEFINIÇÕES E CONVENÇÕES UTILIZADAS NA LINGUAGEM LICOR

Com a finalidade de evitar ambigüidades, estabelecemos algumas definições e convenções. As figuras 4 e 5 auxiliam as definições e convenções a serem dadas.

Ponto de trajetória de referência (p_i^k) — é um par de coordenadas do plano t x X, do atuador K, fornecido pelo usuário.

$$p_i^k \triangleq (t_i^k, x_i^k),\ i \in [1, n]$$

onde:

t — é o eixo do tempo da trajetória;

X — é o eixo de posição da trajetória;

t_i^x — é a coordenada de tempo de referência do atuador K fornecida pelo usuário;

x_i^k — é a coordenada de posição de referência do atuador K fornecida pelo usuário.

Ponto inicial de trajetória de referência (p_o^k) é um par de coordenadas do plano t x X, do atuador k.

$$p_o^k \triangleq (0, x_o^k)$$

onde x_o^k é a posição do atuador k no momento inicial ($t = 0$).

Trajetória de referência do atuador k é uma curva suave no plano t x X que passa pelo ponto inicial e pelos pontos de trajetória de referência. A linguagem LICOR implementa esta curva com funções cúbicas do tipo spline. Estas funções são definidas em cada intervalo "i" de trajetória que corresponde ao trecho de trajetória compreendido entre p_{i-1}^k e p_i^k. As funções cúbicas do tipo spline são descritas analiticamente pelos coeficientes de trajetória, sendo necessários neste caso quatro coeficientes.

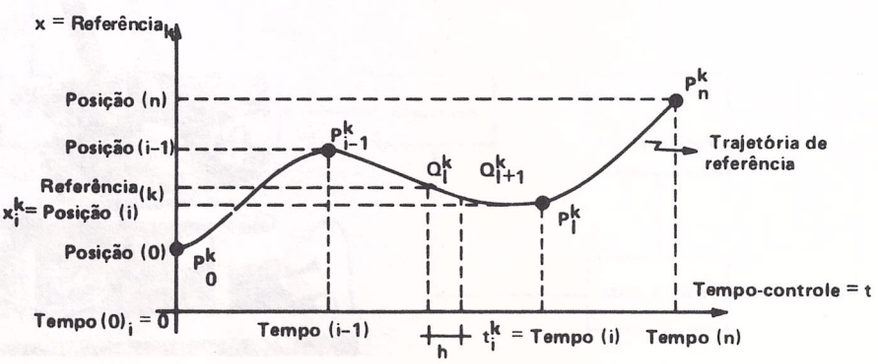


FIG. 4 — Trajetória padrão utilizada pela linguagem LICOR.

A discretização da trajetória de referência do atuador k com um período de amostragem h fornece os pontos de controle Q_j^k deste atuador.

$$Q_j^k \triangleq (t_j , x_j), j \quad [0, m]$$

onde "m" é o número de pontos de controle.

Os pontos de controle Q_j^k servem de referências para o controlador do atuador k.

Movimento é o conjunto simultâneo das trajetórias de referência, conforme ilustra a figura 5.

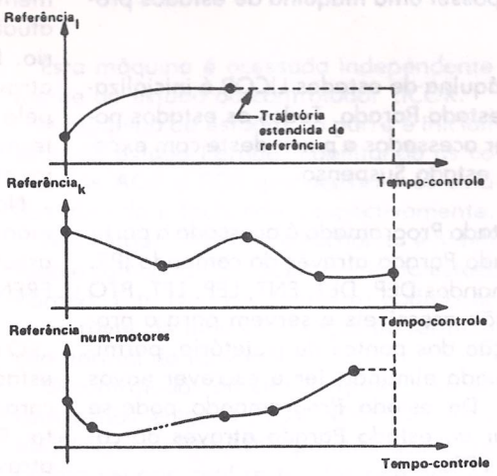


FIG. 5 — Movimento padrão utilizado na linguagem LICOR.

Trajetória estendida de referência é a trajetória formada pela trajetória de referência definida pelo usuário acrescida de um trecho de trajetória com coordenada de posição constante e igual à última coordenada fornecida pelo usuário e estendida até o valor de tempo máximo do movimento.

O número de pontos de controle "m" é obtido pelo quociente (inteiro) da última coordenada de tempo da trajetória estendida de referência e o período de amostragem h.

Trajetória realizada discretizada é o conjunto de pontos $\bar{Q}_j^k$ resultantes amostragem da resposta do atuador k.

Terminação anormal de movimento ocorre quando uma trajetória é interrompida devido a um impedimento de seguir a trajetória de referência, como por exemplo no caso de colisão com obstáculos. Esta terminação é definida por um valor máximo de erro.

4. A ESTRUTURA DA LINGUAGEM LICOR

A linguagem LICOR é acessada pelo programa usuário através de chamadas ao procedimento:

LICOR (COMANDO, PARÂMETRO 1, PARÂMETRO 2)

onde:

COMANDO — é um mnemônico que seleciona a tarefa a ser executada. São 22 os mnemônicos possíveis, sempre compostos por 3 letras. Estes mnemônicos são apresentados na tabela 1.

COMANDO	TAREFA	DO ESTADO
ABO	Abortar movimento	DEG, MOV, SUSP
AGR•	Abrir a garra	—
DEG	Realizar degrau de posição	PARADO
DEP	Eliminar programação de ponto	PROG
DET	Eliminar programação de trajetória	PROG
ENT	Entrar em ponto de trajetória	PROG
EST	Retornar estado atual	TODOS
FGR	Fechar a garra	—
FIM	Finalizar LICOR	TODOS
FPR	Finalizar programação	PROG
INI	Inicializar LICOR	—
IPR	Iniciar programação	PARADO
LEP	Ler posição programada	PROG
LET	Ler tempo programado	PROG
MOV	Executar movimento	PARADO
PGR	Parar movimento da garra	—
POS	Retornar posição atual	TODOS
PTO	Retornar número de pontos de trajetória	PROG
RET	Retornar movimento	SUSP
SEL	Selecionar atuador	PROG
SUS	Suspender movimento	MOV
TPO	Retornar tempo de controle	DEG, MOV, SUSP

TABELA 1 — Relação de Comandos da Linguagem LICOR.

PARÂMETRO 1 — é um número ou variável do tipo inteiro que fornece valores a LICOR, conforme o comando utilizado.

PARÂMETRO 2 — é uma variável do tipo inteiro que fornece e/ou recebe valores de LICOR, conforme o comando utilizado. Este também sinaliza ao usuário um erro ou passagem de parâmetro inválido, conforme indica a tabela 2.

CÓDIGO DE ERRO	DESCRIÇÃO DO ERRO
9990	Comando Inválido
9991	Parâmetro 1 Inválido
9992	Parâmetro 2 Inválido
9993	Erro Fatal

TABELA 2 — Relação de Códigos de Erro da Linguagem LICOR.

5. O CONTROLADOR DA LINGUAGEM LICOR

O objetivo do controlador da linguagem LICOR é gerenciar os comandos determinando a tarefa a ser realizada. Conforme ilustra a figura 6, o controlador da linguagem LICOR é representado por uma máquina de estados, composta de 5 estados:

— Parado,
— Programando,
— Movendo,
— Degrau,
— Suspenso.

Os comandos EST e POS, leitura do estado e da posição atual, respectivamente, podem ser acessados de qualquer estado do controlador.

Para os comandos AGR, PGR e FGR, que realizam tarefas com a garra, a linguagem LICOR possui uma máquina de estados própria.

A máquina de estados LICOR é inicializada no estado Parado. Todos os estados podem ser acessados a partir deste com exceção do estado Suspenso.

O estado Programado é acessado a partir do estado Parado através do comando IPR. Os comandos DEP, DET, ENT, LEP, LET, PTO e SEL são acessíveis e servem para a programação dos pontos de trajetória, permitindo ainda eliminar, ler e escrever novos pontos. Do estado Programando pode-se retornar ao estado Parado através do comando FPR.

O estado Movendo caracteriza a execução de um movimento pré-programado; pode ser acessado a partir dos estados Parado ou Suspenso, conforme a execução de um novo movimento, comando MOV, ou na continuidade de um movimento pré-iniciado, comando RET, respectivamente. O abandono do estado Movendo ocorre através de comandos ou de uma sinalização de terminação normal ou anormal de movimento. No caso do abandono ocorrer através de comandos, a máquina de estado assume os estados Parado ou Suspenso conforme o comando ABO e SUS, respectivamente. Entretanto, nesta transição de estados, a máquina assume um estado temporário intermediário denominado FRENANDO com a finalidade de parar rapidamente os atuadores. Caso o abandono ocorre por sinalização de terminação do movimento, a máquina de estado assume o estado Parado. O comando TPO é habilitado.

O estado Degrau caracteriza o deslocamento de um ou mais atuadores da posição atual para uma posição definida pelo usuário. É acessado a partir do estado Parado através do comando DEG e abandonado pelo comando ABO ou por sinalização de terminação normal ou anormal do movimento, retornando ao estado Parado.

No caso do abandono ocorrer pelo comando ABO, a máquina de estado também assume o estado temporário intermediário FRENANDO. O comando TPO é habilitado.

O estado Suspenso é acessado a partir do estado Movendo através do comando SUS e caracteriza a interrupção de um movimento. Pode-se retornar ao estado Movendo através do comando RET, retomando o movimento interrompido. O abandono também pode ocorrer com a transição para o estado Parado, através do comando ABO. O comando TPO é habilitado.

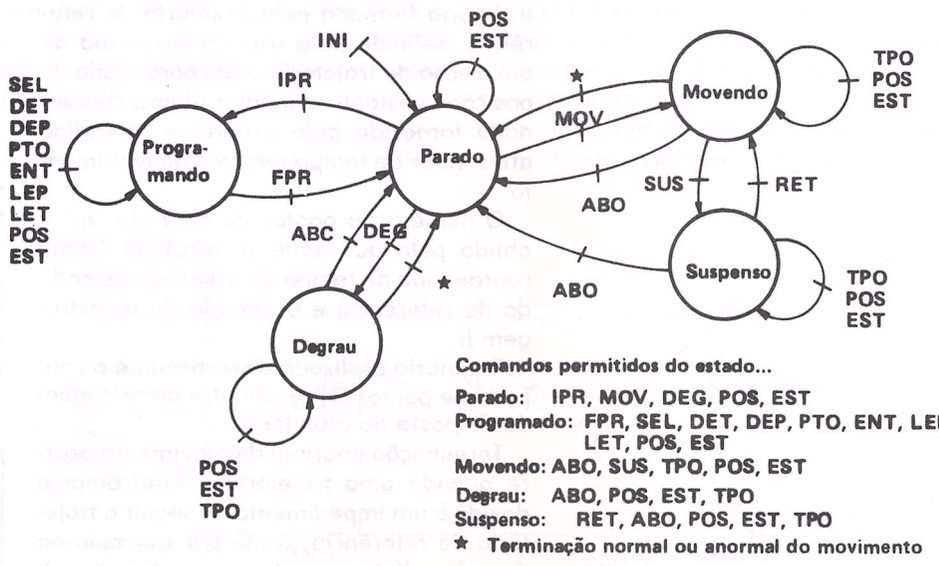
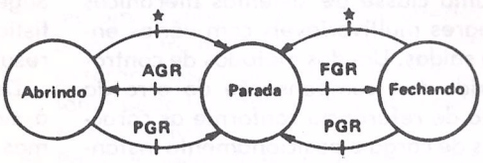


FIG. 6 — A máquina de estado do controlador LICOR.

Conforme ilustra a figura 7 a garra possui uma máquina de estados composta de três estados:

— Parada,
— Abrindo,
— Fechando.



Comandos permitidos do estado da garra

Parada: AGR, FGR.
Abrindo: PGR.
Fechando: PGR.

★ **Terminação normal do movimento.**

FIG. 7 — A máquina de estado da garra.

Esta máquina é acessada independentemente do estado do controlador LICOR.

A máquina de estados da garra é inicializada no estado Parada, habilitando os comandos AGR e FGR que acessam os estados abrindo e fechando, respectivamente.

Nos estados Abrindo e Fechando é habilitado o comando PGR que retorna ao estado Parado. Este retorno, também, pode ocorrer através de uma sinalização de terminação normal de movimento.

O controlador de LICOR, quando se encontra nos estados MOVENDO, DEGRAU, SUSPENSO ou PARADO, permite que o programa usuário realize tarefas que não envolvam a biblioteca LICOR, apresentando, portanto, uma forma de concorrência. Esta característica é importante, por exemplo, no tratamento dos sensores externos quando um movimento está sendo realizado.

6. O DIAGRAMA DE FLUXO DE DADOS DA LINGUAGEM LICOR

A figura 8 ilustra o diagrama de fluxo de dados da linguagem LICOR caracterizando três processos:

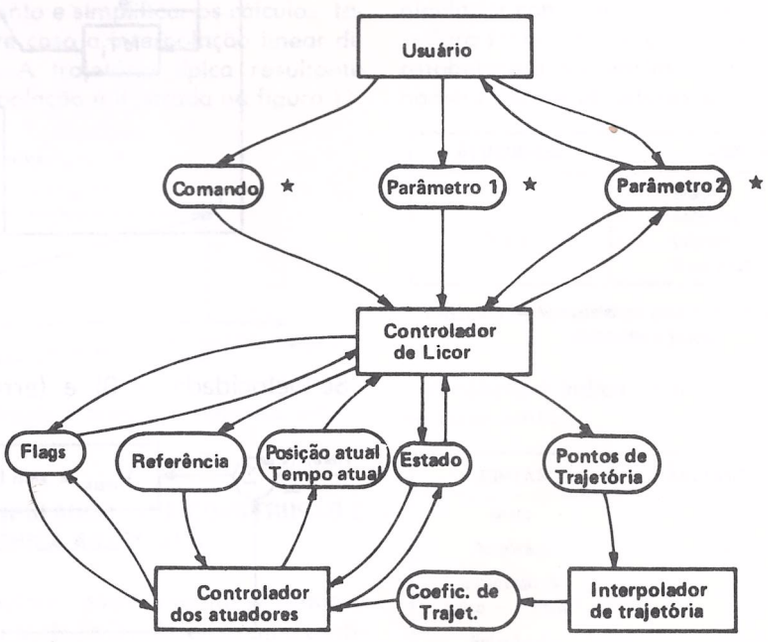
— Controlador de LICOR,
— Controlador dos atuadores,
— Interpolador de trajetória.

No controlador de LICOR está implementada a máquina de estados, descrita na seção 5. O controlador LICOR é acessado pelo usuário através dos dados comando, parâmetro 1 e parâmetro 2. Os dados comando e parâmetro 1 fluem do usuário para o controlador, ao passo que o parâmetro 2 pode ser modificado pelo usuário ou pelo controlador LICOR.

O controlador dos atuadores é responsável pelo controle individual dos atuadores. Ele é ativado pelos flags e pelo estado do controlador LICOR. Os dados de referência de posição ou coeficientes de trajetória são passados para o controlador dos atuadores do controlador de LICOR e do interpolador de trajetória, respectivamente.

O controlador dos atuadores retorna os dados de posição atual de cada atuador e o tempo de controle para o controlador de LICOR.

O interpolador de trajetória tem o objetivo de fornecer ao controlador dos atuadores os coeficientes de trajetória a partir dos dados de pontos de trajetória fornecidos pelo controlador de LICOR.



★ **Argumento de procedimento**

FIG. 8 — Diagrama de fluxo de dados de LICOR.

7. O CONTROLADOR DOS ATUADORES

Na literatura sobre métodos de controle de robôs, destacamos:

VUKOBRATOVIC & STOKIC [7] que descrevem a complexidade do controle de robôs e manipuladores, por estes pertencerem a uma classe de sistemas mecânicos não lineares multivariáveis com várias entradas e saídas. Um dos métodos de controle sugerido é a compensação do erro da trajetória de referência conforme as características de carga e posicionamento instantâneo do manipulador (feedforward).

PAUL [8] e FU et alii [9] que sugerem a implementação do controle individual dos atuadores como um nível hierárquico mais baixo do controle de todos os atuadores interrelacionados.

SNYDER [10] que descreve diferentes técnicas de controle dos atuadores individualmente, destacando a técnica de controle de velocidade.

BIHN e STEVE [11] que descrevem a utilização de um controlador PID como controle de trajetória do robô PUMA 560 da Unimation, Inc. Os coeficientes de ganho deste controle foram determinados experimentalmente por tentativa e erro. Dizem, ainda, que o ganho de todas as juntas foram reduzidos para produzir um sistema mais estável, devido às oscilações causadas pelo forte acoplamento entre as juntas do robô. Sugerem que técnicas de controle mais sofisticadas deverão ser usadas para produzir resultados melhores.

Devido à filosofia da linguagem LICOR e à simplicidade de implementação, adotamos o controle individual dos atuadores. Após a realização de alguns testes experimentais e baseado na literatura descrita implementamos o controlador dos atuadores conforme ilustra a figura 9.

Quando a velocidade do atuador for nula e o erro de posição menor que o erro admissível, o atuador é desligado e o controlador deste atuador é desativado.

Os atuadores utilizados no manipulador experimental possuem uma zona morta elevada, e são caracterizados como um sistema não linear, conforme descreve a seção 9.

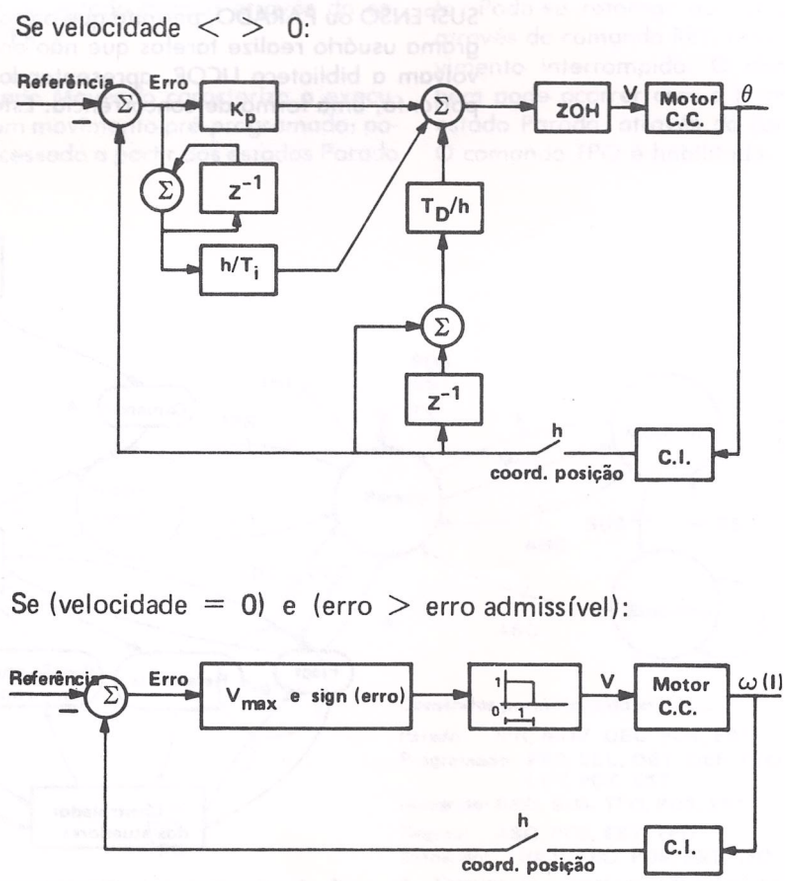


FIG. 9 — Diagrama de blocos do controlador dos atuadores.

Face a isto, foram definidas duas leis de controle conforme a região de operação do atuador:

a) Para a velocidade diferente de zero é utilizado um controlador PID sendo as constantes K_p, T_i e T_d ajustadas experimentalmente por tentativa e erro. A estrutura deste PID pode ser vista como uma malha de velocidade interna e uma outra malha PI de posição mais externa. Assim, uma velocidade constante pode ser mantida somente pelo termo derivativo.

b) Para a velocidade nula com erro maior que o erro admissível aplica-se ao motor um pulso de tensão máxima conforme o sinal do erro de posição do atuador.

8. O INTERPOLADOR DE TRAJETÓRIA

KREUZER e TRUCKENTRODT [3] relatam que para definir uma trajetória a partir dos pontos de trajetória são utilizadas, com maior freqüência em robótica, as interpolações linear e spline.

A interpolação linear possui características de simplicidade de implementação e rapidez de processamento.

A interpolação spline descreve uma curva suave que passa pelos pontos de trajetória, a fim de evitar variações bruscas de velocidade do atuador.

O interpolador de trajetória da linguagem LICOR utiliza a interpolação cúbica tipo spline, descrevendo uma trajetória típica, conforme ilustra a figura 10.

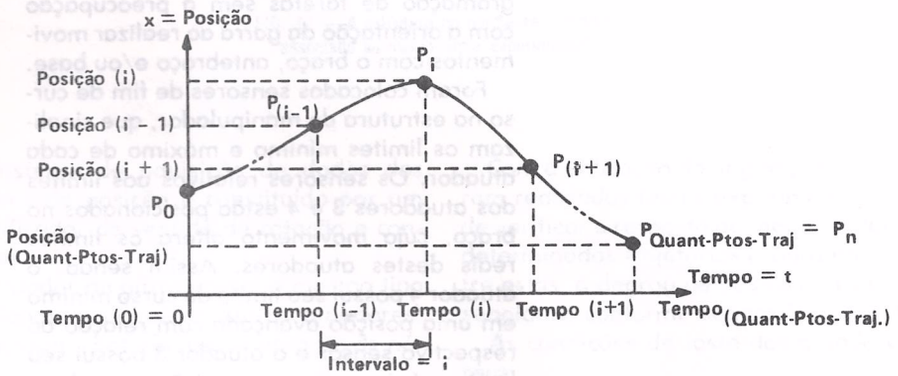


FIG. 10 — Traçado de trajetória por interpolação spline.

É interessante observar que, para que a curva interpolada seja satisfatória do ponto de vista do movimento, os pontos de trajetória devem ser convenientemente fornecidos pelo usuário.

Um caso particular da interpolação spline ocorre quando a trajetória é composta por apenas um ponto de trajetória além do inicial. Neste caso, a curva resultante será uma reta. A fim de minimizar o tempo de processamento e simplificar os cálculos, fazemos neste caso a interpolação linear diretamente. A trajetória típica resultante desta interpolação é ilustrada na figura 11.

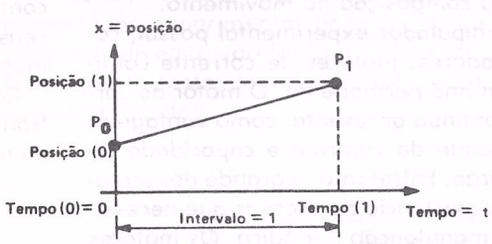


FIG. 11 — Traçado de trajetória por interpolação linear.

9. O MANIPULADOR, SUA CONSTRUÇÃO E ELETRÔNICA ASSOCIADA

O manipulador experimental utilizado é descrito por SCIESZKO [12] tendo uma configuração com coordenadas rotativas, com 5 graus de liberdade e uma garra como dispositivo de extremidade. A transmissão dos movimentos é feita através de cabos de aço, permitindo a colocação dos atuadores na base do manipulador. Nesta base existem quatro atuadores: um para o movimento da junta do braço, outro para a junta do antebraço e mais dois que formam, através de um mecanismo diferencial, os movimentos do pulso, inclinação — "pitch" e rotação — "roll". A base na qual estão situados estes atuadores é giratória, e está associada ao quinto atuador, formando asim um manipulador com cinco graus de liberdade.

Para facilidade de exposição, a tabela 3 estabelece a nomenclatura para o relacionamento entre atuadores e juntas.

ATUADORES	JUNTAS
1	Braço
2	Antebraço
3 e 4	Diferencial de Pulso
5	Base rotativa

TABELA 3 — Nomenclatura para o relacionamento entre atuadores e juntas.

A tabela 4 indica os ângulos de rotação de cada junta.

JUNTAS	ÂNGULOS LIMITES
Braço	+ 45°
Antebraço	+ 45°
Inclinação do pulso — "Pitch"	+ 45°
Rotação do pulso — "Roll"	+ 180°
Base	+ 150°

TABELA 4 — Ângulos Limites de Rotação das Juntas.

A redução dos atuadores 3 e 4 são iguais. A rotação longitudinal — "roll" e o movimento de inclinação — "pitch" do pulso são realizados através dos atuadores 3 e 4, quando estes se movem respectivamente em sentidos opostos ou no mesmo sentido com a mesma velocidade. Este mecanismo diferencial possui a característica de apresentar entre os atuadores 3 e 4 uma diferença angular constante no movimento de inclinação e uma soma angular constante no movimento de rotação.

Os movimentos de braço e antebraço não alteram a posição angular do pulso devido ao desacoplamento dos movimentos de antebraço em relação ao braço e do pulso em relação aos movimentos de braço e antebraço. Esta geometria simplifica a programação de tarefas sem a preocupação com a orientação da garra ao realizar movimentos com o braço, antebraço e/ou base.

Foram colocados sensores de fim de curso na estrutura do manipulador, que sinalizam os limites mínimo e máximo de cada atuador. Os sensores relativos aos limites dos atuadores 3 e 4 estão posicionados no braço, cujo movimento altera os limites reais destes atuadores. Assim sendo, o atuador 4 possui seu limite de curso mínimo em uma posição avançada com relação ao respectivo sensor e o atuador 3 possui seu limite máximo em uma posição recuada.

A posição inicial do manipulador, quando ele está recolhido (posição de repouso), é quando todos os atuadores estiverem nos respectivos sensores de fim de curso de limite mínimo, exceto o atuador 4, que estará em uma posição avançada em relação a este.

Na extremidade do manipulador está situada uma garra com mecanismo de barras paralelas que permitem movimentos paralelos da pinça. Como atuador é utilizado um motor de corrente contínua montado no próprio pulso. Como o movimento de abrir e fechar da garra não é considerado um grau de liberdade, o atuador, a ela associado, será tratado separadamente sem fazer parte da composição do movimento.

O manipulador experimental possui, como atuadores, motores de corrente contínua com ímã permanente. O motor de corrente contínua apresenta, como vantagem, simplicidade de controle e capacidade de sobrecarga. Entretanto, a grande desvantagem é a existência de escovas que necessitam de manutenção periódica. Os motores de corrente contínua utilizados no manipulador possuem uma zona morta elevada, ou seja, existe uma faixa de tensão, na qual o motor não se move. Assim sendo, este motor é caracterizado como um sistema não linear. Este tipo de motor foi contudo utilizado devido ao seu baixo custo.

O manipulador experimental possui sensores de posição e de fim de curso.

O codificador incremental foi o sensor de posição utilizado no manipulador experimental devido às suas características de simplicidade, custo e tamanho reduzido. Este sensor é constituído de um disco ótico ranhurado, acoplado diretamente ao eixo do motor, a fim de aumentar a resolução e precisão de posicionamento do manipulador, devido às reduções dos atuadores. O disco ótico utilizado com 12 ranhuras, permite uma resolução de 30° de variação angular no motor, que é refletida para as juntas, e que estão representadas na tabela 5.

JUNTAS	RESOLUÇÃO ANGULAR (GRAU)
Braço	0,0885
Antebraço	0,0875
Base	0,0425
Pitch	0,152
Roll	0,23

TABELA 5 — Resolução angular das juntas do manipulador experimental.

O codificador incremental utiliza detetores fotoelétricos para obtenção do sinal correspondente à posição. Estes detetores são posicionados de tal forma que os sinais elétricos possuam uma defasagem de 90°. Os detetores fotoelétricos utilizados foram chaves óticas infravermelho, modelo MCC 860 T. As desvantagens deste tipo de sensor são o complicado processamento de sinal para reconhecimento do sentido de rotação, a medida de posição não ser absoluta e a sensibilidade dos detetores fotoelétricos em relação à vibração e à poluição. Uma das vantagens da utilização do codificador incremental é a precisão do sensor estar relacionada unicamente ao contador de posição a ele associado. O fato da medida de posição não ser absoluta implica o manipulador recorrer a uma posição de referência sempre que o sistema for ligado ou resetado. Esta referência foi dada pelos sensores de fim de curso mínimo dos atuadores. Além disto, estes sensores têm a finalidade de limitar a faixa de trabalho destes atuadores.

Os sensores de fim de curso, normalmente utilizados, são as microchaves de fim de curso, sistemas óticos, capacitivos e indutivos. Os três últimos possuem a vantagem de não ter contato mecânico, mas, em contrapartida, são de custo elevado. Os sensores utilizados no manipulador experimental foram microchaves de fim de curso.

Conforme ilustra a figura 12, a parte eletrônica associada ao manipulador experimental é composto de três sistemas:

— processamento,
— acionamento, e
— aquisição.

O sistema de processamento é constituído por um microcomputador de 16 bits, compatível com o IBM-PC com as seguintes características:

— Monitor de vídeo 12" monocromático,
— Acionador de disco flexível de 5 ¼ ",
— Memória principal mínima (RAM): 256 Kbytes,
— Co-processador 8087, e
— Freqüência de clock: 4,7 MHz.

O sistema de acionamento dos atuadores é composto por um chopper e um gerador de sinal PWM.

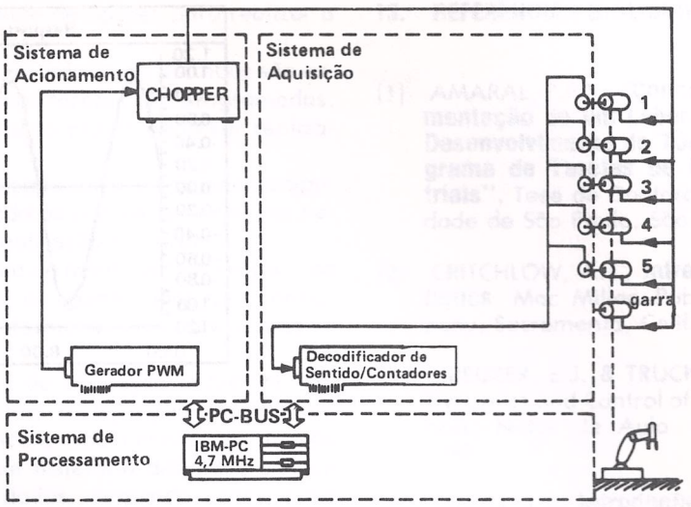


FIG. 12 — A estrutura do hardware eletrônico associado ao manipulador experimental.

O sistema de aquisição de dados dos sensores de posição é constituído por um decodificador de sentido de rotação e contadores.

O atuador da garra possui o mesmo tipo de acionamento que os outros atuadores, sem possuir, contudo, realimentação de posição.

O programa de interface tem as seguintes tarefas:

— Inicialização dos sistemas de aquisição e acionamento,

— Leitura dos dados do sistema de aquisição,

— Fornecimento de referência para o sistema de acionamento, e

— Supervisão dos sensores fim de curso.

10. OS TESTES EXPERIMENTAIS E EXEMPLOS DE APLICAÇÃO

A linguagem LICOR foi utilizada em um manipulador experimental no qual foram realizadas diversas tarefas com a finalidade de aplicar, testando e depurando, todos os seus comandos.

Com a utilização da linguagem LICOR, foram realizados testes experimentais, a fim de verificar a resposta do manipulador para determinadas trajetórias programadas; entre estas, o degrau de posição e trajetórias senoidais, conforme sugere SNYDER [10].

As condições de teste dos atuadores foram:

— Configuração do manipulador que proporciona o máximo momento de inércia para o atuador a ser testado;

— Parâmetros de controle ajustados a fim de minimizar o "overshoot", conseqüentemente aumentando o tempo de assentamento. Este tempo é elevado, também, devido à zona morta do motor e ao período de amostragem utilizado.

As respostas dos atuadores ao degrau realizado, da coordenada de posição mínima à máxima do respectivo atuador, são mostradas na figura 13, tomando-se como exemplo os atuadores 3 e 4.

A trajetória senoidal padrão adotada foi:

— Com amplitude máxima e mínima correspondente às respectivas coordenadas de posição máxima e mínima dos atuadores;

— Com período correspondente à velocidade máxima dos atuadores.

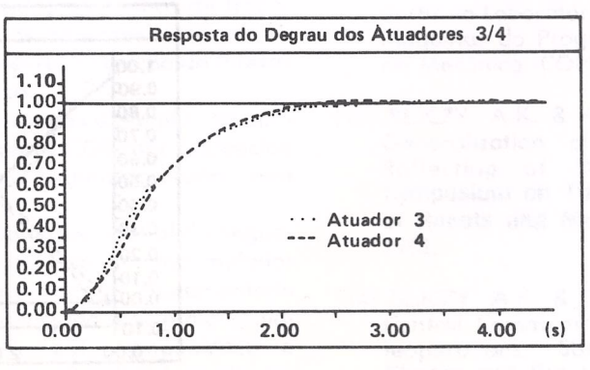


FIGURA 13

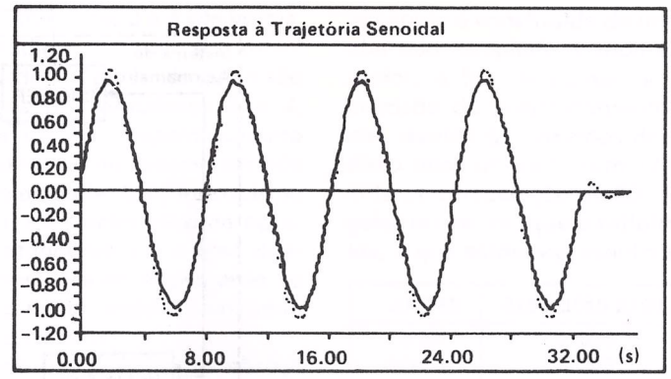


FIGURA 14

A figura 14 ilustra a resposta do atuador 1, tido como exemplo, à trajetória senoidal, destacando a referência e a trajetória realizada.

A referência é uma aproximação de uma função senoidal, obtida pela interpolação de 30 pontos de trajetória programados. Estes pontos foram calculados seguindo os critérios da trajetória senoidal padrão adotada.

Implementamos alguns exemplos de aplicação da linguagem LICOR, entre eles, os dirigidos à telemanipulação e aplicações industriais.

BEJCZY e HANDLYKKEN [13] e BEJCZY e CORKER [14] descrevem a telemanipulação como sendo o controle remoto de um manipulador através de dispositivo que o simula, podendo ser uma réplica mecânica deste ou um simples "joystick". Este dispositivo pode fornecer referências de posição, velocidade, torque, entre outros.

A telemanipulação é útil para a realização de:

— Tarefas em ambientes agressivos, como os corrosivos ou contaminados com elementos biológicos e nucleares;

— Tarefas de difícil acesso para o homem, como trabalho em águas profundas.

O exemplo de telemanipulação implementado com linguagem LICOR consiste em controlar o manipulador experimental através de um "joystick" que ativa os atuadores deste com velocidade constante.

Os parâmetros fornecidos pelo usuário são a região de operação do atuador, posição mínima e máxima e o tempo de realização desta trajetória. Estes parâmetros definem a velocidade do atuador como:

$$\text{VELOCIDADE } |I| = \frac{\text{Pos Max } |I| \quad \text{Pos Min } |I|}{\text{Tempo } |I|}$$

sendo o atuador identificado por "I".

O movimento é realizado após a programação das trajetórias, conforme a posição do "joystick". A trajetória é composta por um ponto, cujo parâmetro de posição é a máxima ou mínima referente ao atuador e o parâmetro de tempo calculado em função da velocidade desejada e posição atual. A programação de uma trajetória com um ponto implica uma interpolação linear entre este e o ponto atual, conforme descrito na seção 8, caracterizando uma velocidade constante do atuador.

Temos como exemplo a figura 15 que mostra resposta típica dos atuadores ao movimento "pitch".

O movimento da garra é controlado apenas com os comandos de abrir e fechar.

Um exemplo de telemanipulação realizado foi o de retirar objetos de um lugar e colocá-los em outro, com auxílio de "joysticks". Para isto foram utilizados dois "joysticks" com dois graus de liberdade cada um. Assim conseguimos o movimento de quatro juntas simultâneas, garantindo uma

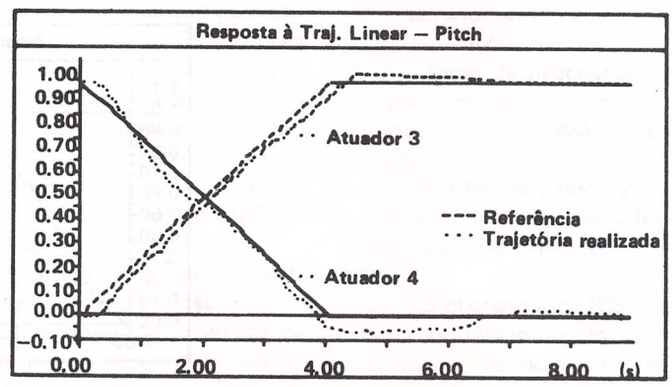


FIGURA 15

boa flexibilidade de opções para realizar a tarefa proposta.

As aplicações industriais de robôs são caracterizados por tarefas pré-programadas. Esta programação prévia pode ser realizada por:

— Aprendizado, através de telemanipulação, salvando os pontos de trajetória necessários à realização da tarefa;

— Definição do movimento no espaço de trabalho onde as coordenadas tridimensionais são transformadas em coordenadas de juntas.

O exemplo de aplicação industrial, implementado com a linguagem LICOR, consiste em executar movimentos em função dos pontos de trajetória definidos em um arquivo de dados, seguindo entre estes pontos uma trajetória suave.

Como exemplo de aplicação, foi realizada uma tarefa repetitiva de retirar uma esfera da extremidade de um trilho inclinado e soltá-la na outra extremidade. A programação dos pontos de trajetória foi feita previamente com auxílio da telemanipulação. Observamos que o movimento realizado foi suave e em conformidade com os pontos programados.

11. CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma linguagem de programação de robôs.

Esta linguagem caracterizou-se pela simplicidade na programação das tarefas, apresentando um bom desempenho quando a tarefa é programada com pontos de trajetória, não se determinando o movimento entre estes pontos.

A linguagem LICOR apresentou, como vantagem, permitir ao usuário a implementação de programas aplicativos de robótica, sem a preocupação com os fatores de controle de baixo nível do manipulador. Além disso, esta linguagem mostrou-se uma boa base para o desenvolvimento hierárquico de sistemas mais complexos. Estes sistemas poderiam incluir um nível de especificação de movimentos no espaço de trabalho, envolvendo os sensores externos, e mais acima ao nível de descrição de tarefas de forma simbólica.

Foram implementados exemplos de aplicação da linguagem LICOR que evidenciou o diálogo simples e seguro do usuário com o robô.

Para verificação experimental da linguagem LICOR foi utilizado um manipulador mecânico. Assim sendo, foi implementada uma eletrônica associada a este manipulador, caracterizada pela simplicidade e apresentando boa confiabilidade e eficiência na realização dos testes.

13. REFERÊNCIAS BIBLIOGRÁFICAS

[1] AMARAL, P.F.S., "**Concepção e Implementação de um Laboratório para o Desenvolvimento de Técnicas e Programa de Tarefas de Robôs Industriais**", Tese de Doutorado, Universidade de São Paulo, São Paulo, 1985.

[2] CRITCHLOW, A.J., **Introduction to Robotics**, Mac Millan Publishing Company, Sacramento, Califórnia, 1985.

[3] KREUZER, E.J. & TRUCKENTRODT, A., Dynamics and Control of Industrial Robots, Notas de Aula, COPPE/UFRJ, 1988.

[4] CRAIG, J.J., **Introduction to Robotics, Mechanics & Control,** Addison-Wesley Publishing Company, Stanford, 1986.

[5] TAYLOR, E.; SUMMER, P. & MEYER, J., AML: A Manufacturing Language, **Int. J. of Robotics Research,** vol 1, nº 3, Fall 1982.

[6] MESHKAT, S., "High-level Motion Control Programming Using DSPS", **Control Engineering,** vol. 35, nº 2, Fevereiro, 1988.

[7] VUKOBRATOVIC̄, M. & STOKIC̄, D., **Control of Manipulations Robots. Theory and Application,** Communications and Control Engineering Series, Germany, 1982.

[8] PAUL, R.P., **Robot Manipulator — Mathematics, Programming, and Control. The Computer Control of Robot Manipulator**, The MIT Press, Masachusetts, 1981.

[9] FU, K.S.; GONZALES, R.C. & LEE, C.S.G., **Robotics: Control, Sensing, Vision and Intelligence,** Mc-Graw Hill, 1987.

[10] SNYDER, W.E., "Micro-Computer Based Path Control", **Robotics Age**, vol. 2, nº 1, Spring 1980.

[11] BIHN, D.G. & STEVE HSIA, T.C., "Universal Six-Joint Robot Controller, **IEEE Control Systems Magazine,** vol. 8, nº 1, 1988.

[12] SCIESZKO, J.L., Projeto de um Manipulador Mecânico, Relatório Interno nº 04/87 do Laboratório de Dinâmica das Máquinas do Programa de Engenharia Mecânica, COPPE/UFRJ, 1987.

[13] BEJCZY, A.K. & HANDLYKKEN, A.K., Generalization of Bilateral Force-Reflecting of Manipulators, **4th Symposium on Theory and Practice of Robots and Manipulator,** Poland, 1983.

[14] BEJCZY, A.K. & CORKER, K., Manual Control Communication in Space Teleoperations, **5th Symposium on Theory and Practice of Robots and Manipulator,** Italy, 1984.