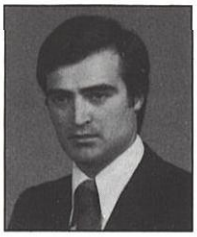
LM



Linguagem de Manipuladores

por: Manuel Gonçalves Soares *

1. Caracterização da linguagem LM

Para melhor caracterizarmos a linguagem LM, vamos expor os critérios que aplicámos aos diversos níveis de programação existentes e relativamente aos quais, aliás, há divergência de opiniões nos diversos sectores da Robótica.

1.1. Tipos de programação em função do nível de abstracção

Conforme o nível de abstracção em que nos colocamos a descrição das tarefas a desempenhar por um robot pode revestir-se de diversas formas de expressão (figura 1).

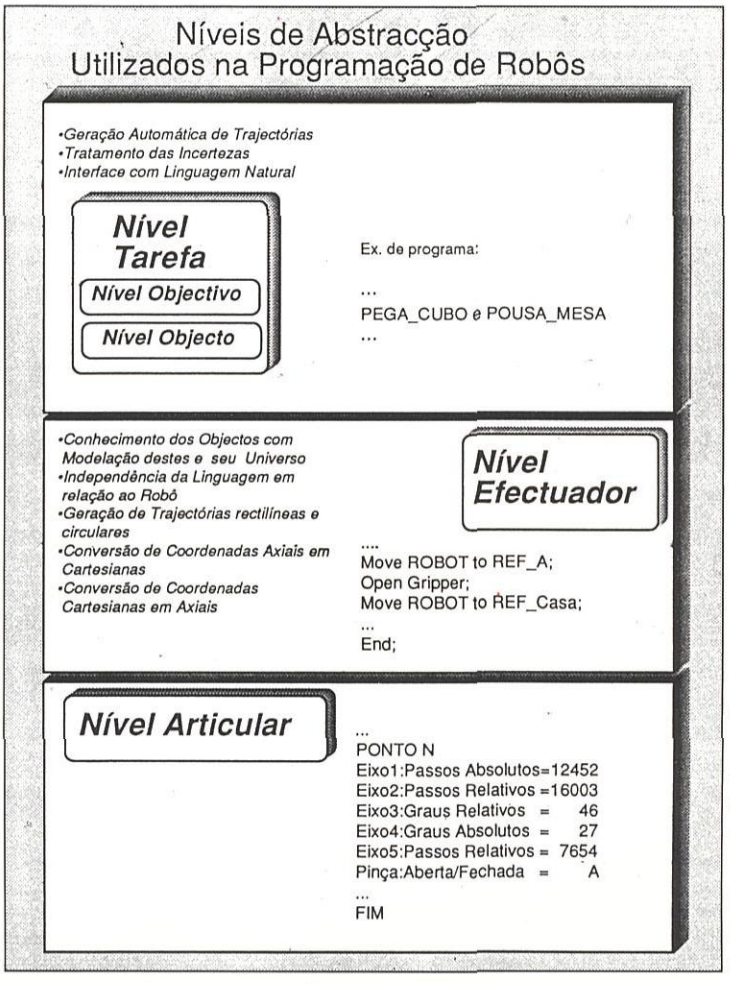


Fig.1

Níveis de Abstracção utilizados na programação de robots

Pretendemos, neste trabalho, apresentar os conceitos mais importantes da linguagem LM. Uma descrição mais detalhada constitui um texto que pode ser consultado (LM — Linguagem de Manipuladores Versão PO1 — Adaptada ao Robô APRA/Lab. IA — linha do Prof. E. Oliveira) na FEUP, ISEP e FCTUCoimbra).

* Professor Adjunto Equiparado/Instituto Superior de Engenharia do Porto

A estes níveis de abstracção, podemos associar diversos tipos de programação, assim como, diversas categorias de sistemas de programação e de controlo/ /comando:

- **O nível articular**, onde a tarefa é descrita em termos dos deslocamentos das articulações do manipulador necessários para a sua realização. **A linguagem LM admite programação a este níve;.**

- **O nível efectuador**, em que o operador intervém para especificar a sucessão de situações (posições, orientações) e acções do efectuador, necessárias à realização da tarefa. É neste nível que se encontra a linguagem LM;

- **O nível objecto**, onde a descrição da tarefa é baseada nos modelos dos objectos susceptíveis de se encontrarem no universo do robot e nas relações entre esses objectos;

- **O nível objectivo**, onde a tarefa é descrita em termos do estado final a atingir e a partir de uma descrição dos objectos.

1.2. Modos de expressão em linguagens robóticas

É, no entanto, ao nível efectuador que, habitualmente, reside a possibilidade de descrever tarefas nas linguagens de programação mais usuais. Dois modos fundamentais de expressão são utilizados, conforme se recorre directamente ao manipulador (programação *Por Aprendizagem*) ou se usa uma linguagem formal (programação *Textual*). É neste segundo modo que se situa a linguagem LM (fig. 2).

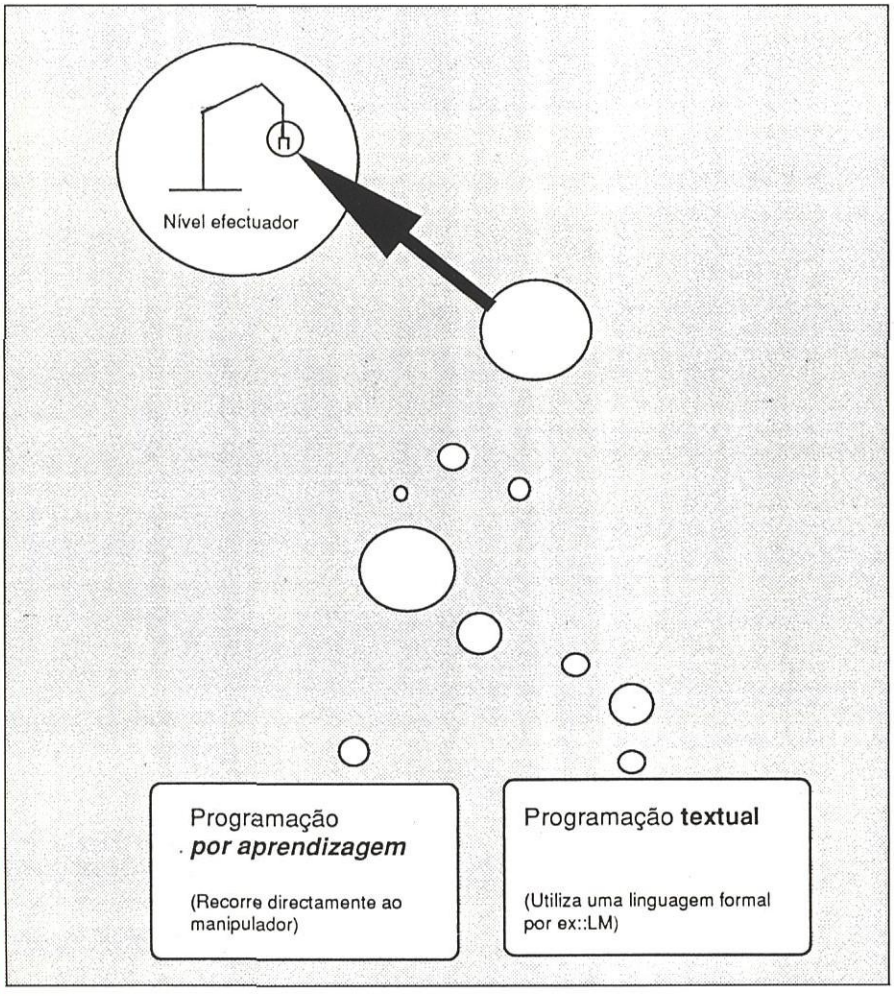


Fig. 2. Modos fundamentais de expressão utilizados na descrição de tarefas ao nível efectuador

1.2.1. LM - Linguagem de Programação Textual

A programação textual foi desenvolvida ao longo da década de 70 em laboratórios de investigação de Robótica e foi exactamente nos finais dessa década que nasceu a linguagem LM [1].

A evolução dos microcomputadores permitiu a realização de controladores para robots, em que todas as tarefas são descritas através de programas escritos em linguagens formais.

A linguagem LM possui, relativamente às linguagens clássicas (FORTRAN, BASIC, PASCAL), tipos de dados e funções próprias para a Robótica, sendo, por esta razã,o incluida nas classes das *linguagens de programação de robots*.

1.2.2. Programação Por Aprendizagem

Relativamente à linguagem LM, é a instrução MANUAL que permite ao programador utilizar este tipo de programação, com o intuito de lhe permitir resolver algumas situações delicadas, normalmente ligadas com a aproximação a objectos.

2. Níveis de programação de uma aplicação

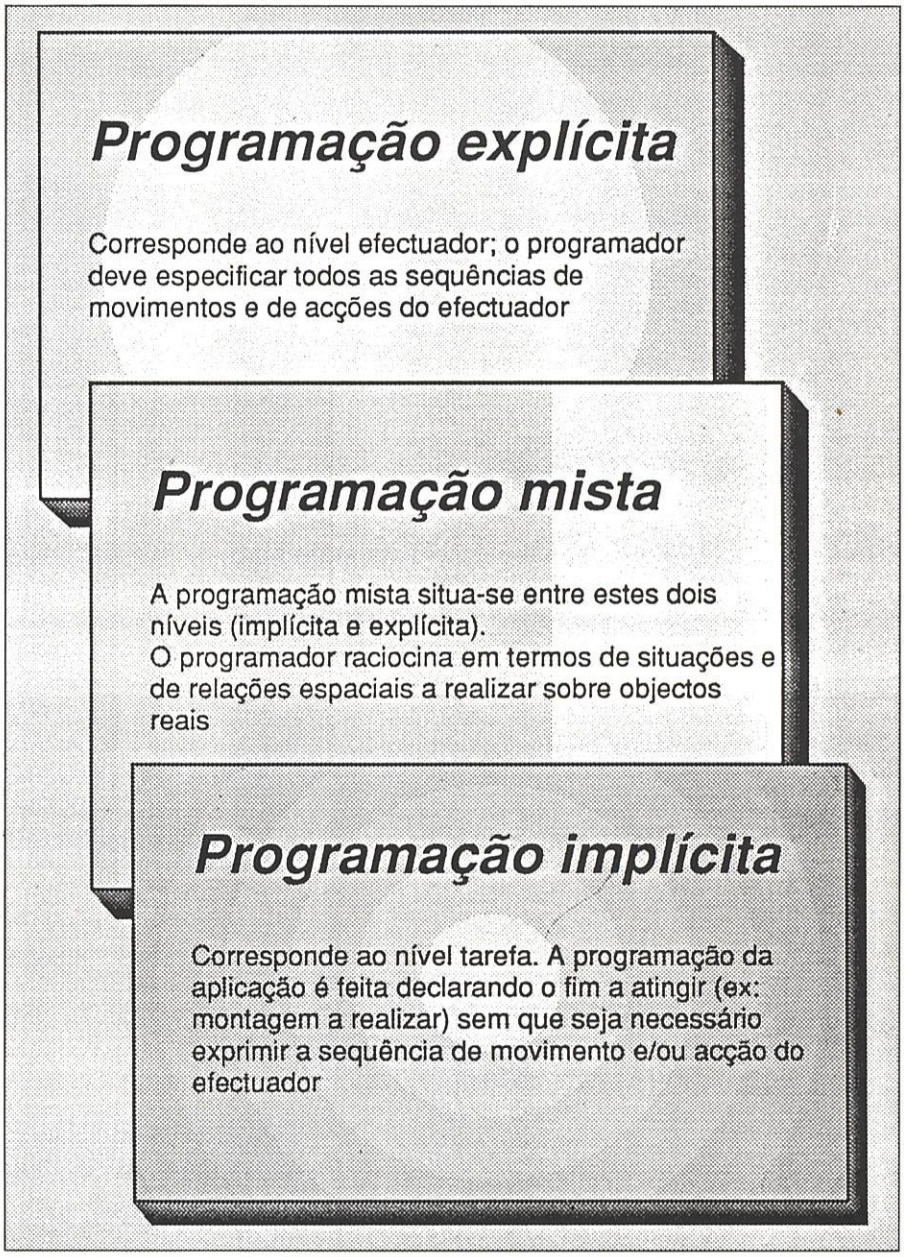
A programação de uma aplicação necessita transformar o manipulador e os objectos reais em objectos

de programação, para, de seguida, lhes aplicar os algoritmos apropriados. Compete ao programador realizar mentalmente estas transformações, adaptando-as à gramática da linguagem utilizada. Ao pretendermos estabelecer uma ligação, entre os níveis referidos na figura 1 e os tipos de programação, é-nos difícil classificar a programação *Textual* (caso da linguagem LM), visto que, tal como outras linguagens, nos permite trabalhar a diferentes níveis. Por esta razão, aceitamos a divisão dos níveis de programação de uma aplicação de acordo com o esquema da figura 3.

Assim, e de acordo com os diferentes níveis de programação, classificámos a programação de uma aplicação em LM ao nível da programação explícita.

Neste tipo de programação, a tarefa é especificada, declarando, explicitamente, toda a sequência de acções do efectuador. Esta sequência comporta a aproximação ao objecto, o pegar no objecto, e o colocar do objecto (que dependem das dimensões geométricas do objecto e do manipulador), e inclui, ainda, o evitar colisões com obstáculos durante o movimento.
O programador é pois inteiramente responsável pela planificação da tarefa. O executivo da linguagem apenas converte as instruções nos dados necessários ao gerador de trajectórias.

Fig. 3. Níveis de programação de uma aplicação



3. LM - Linguagem de programação explícita

A linguagem LM permite descrever uma tarefa em termos dos deslocamentos e das acções a realizar pelo efectuador (que neste momento é uma simples pinça abre/fecha). Tem também a possibilidade de tomar decisões, baseadas nos dados recebidos pelos sensores.
É necessário modelar, geometricamente, o espaço da tarefa a desempenhar, a fim de podermos exprimir as relações funcionais entre os objectos e a pinça (ou outras ferramentas com que equiparmos a extremidade do robot).

O princípio básico para esta modelação, consiste na identificação das posições da extremidade (*pulso*) do manipulador relativamente à posição de um referencial [2].
Em L M este referencial é designado por ROBOT. A figura 4 procura ilustrar este referencial.
Ao elaborarmos qualquer programa em L M, podemos, por comodidade de programaçã,o definir ROBOT como sendo o centro da pinça [3] (basta efectuar uma simples transformação geométrica: translação e/ou rotação, usando TRANSLAT ou ROT [4]) conforme ilustramos na figura. 5.
Este novo referencial é designado

Fig. 4. ***Referencial*** *designado em LM por* ***ROBOT***

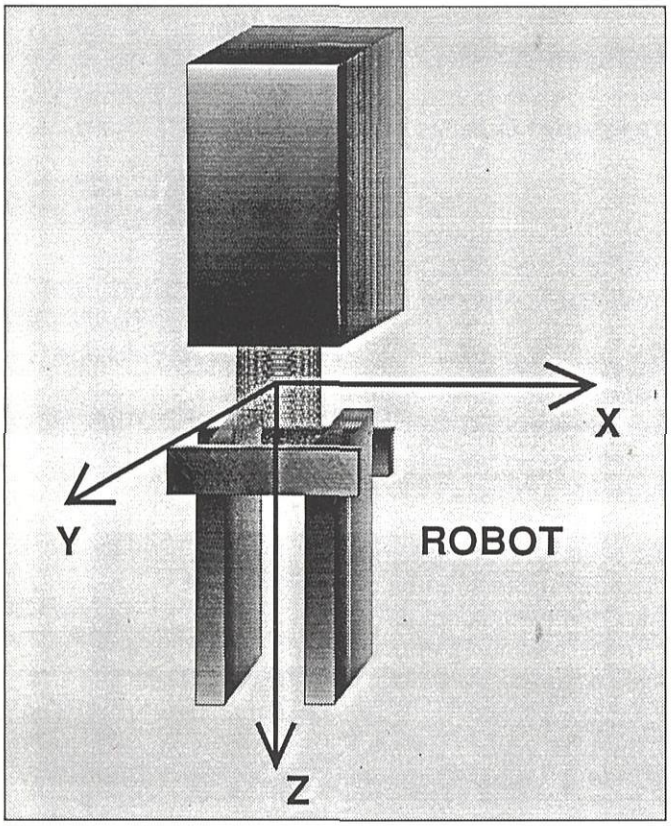


Fig. 6 Referênciais ***ROBOT*** *(centro do pulso) e* ***REF*** *(centro da pinça)*

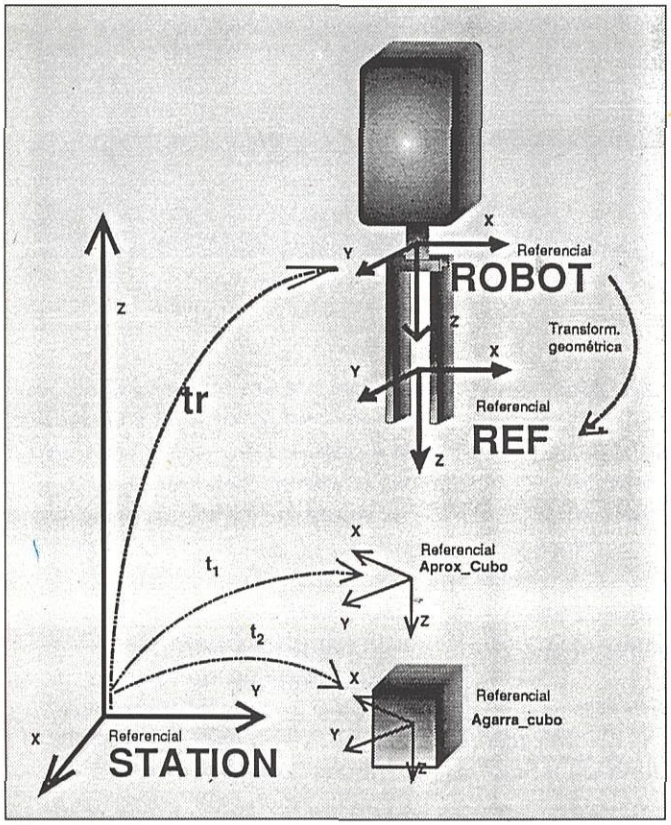
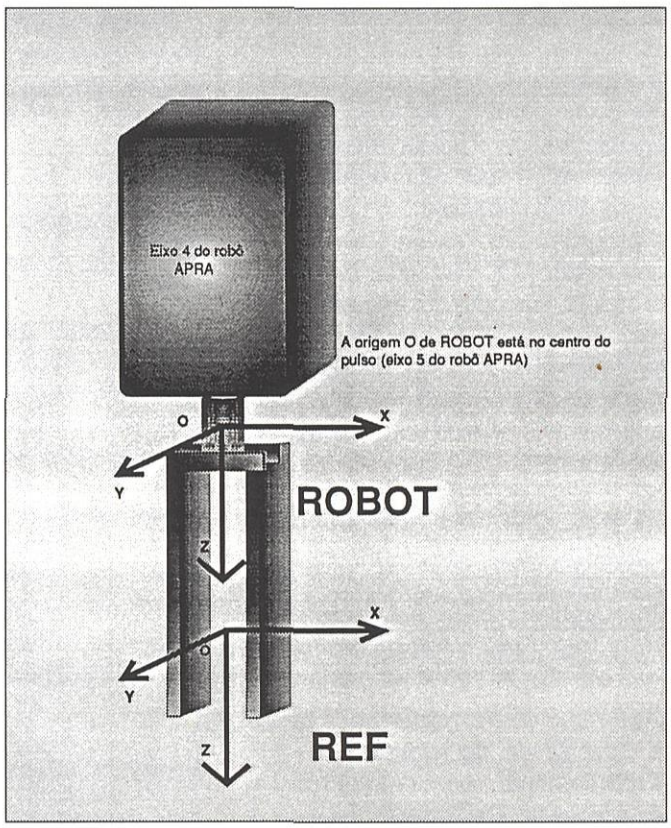


Fig. 5. Referênciais ***ROBOT*** *(centro do pulso) e* ***REF*** *(centro da ferramenta ou da pinça neste caso particular)*



por nós como REF.
Este referencial e a sua versatilidade são destacados na definição do executivo do LM, para o robot com que se trabalhe.

Do ponto de vista do sistema de programação apenas este referencial ROBOT se desloca [5]. A posição deste referencial pode ser representada, num sistema robótico por uma transformação geométrica **tr** (translação e rotação) que dá a posição inicial e a orientação do referencial relativamente ao referencial STATION [6].

Os programas em LM consistem na definição da sequência de movimentos da extremidade do manipulador, com o objectivo de fazer coincidir o referencial REF (previamente definido), com os diversos referenciais dos pontos particulares do ambiente (figura 6).

O nosso sistema LM permite a modelação dos objectos (supostos rígidos) e do ambiente.

Deste modo, ao programarmos nesta linguagem, estamos a criar, para cada objecto, um referencial que traduz a sua localização no espaço.

Um dos papéis essenciais do executivo feito para um robot é deterninar as posições sucessivas do referencial

ROBOT a partir das instruções contidas no programa. Durante a execução, estas posições são convertidas nos valores a atribuir às variáveis articulares (passos dos codificadores) e transmitidas ao gerador de trajectórias do robot.

No entanto, não nos devemos esquecer que dentro da classificação que atribuímos ao LM (linguagem de programação explícita), também se encontra incluido um segundo tipo: as linguagens do tipo primitivas de movimentos.

Estas, ao contrário das linguagens estruturadas (caso de LM), não permitem a modelação do ambiente. Assim temos, de acordo com a figura 7:

4. Linguagem estruturada

4.1. Características gerais

A linguagem LM, como linguagem estruturada, é baseada na linguagem «clássica» PASCAL. Assim, a especificação de uma tarefa, comporta uma zona de declarações, onde são definidas as variáveis, que irão ser utilizadas de acordo com os tipos de dados existentes.

Relativamente aos tipos de dados convencionais e às linguagens do tipo *primitivas de movimentos*, a linguagem LM, por ser estruturada, possui os tipos referencial (FRAME), vector (ARRAY) e transformação (TRANSFORM) para além de funções *standard* e operações sobre estes tipos de dados (composição de transformações, produto escalar, produto vectorial, etc).

A representação de um objecto sólido é feita pelo programador, recorrendo, a variáveis do tipo referencial e do tipo transformação.

Como já dissemos, um único referencial (ROBOT) é deslocável. Para que um referencial de localização referente a um objecto se possa deslocar, devemos criar e destruir ligações entre o referencial do objecto e o referencial ROBOT (AFFIX e UNFIX), de acordo com a tarefa que queremos realizar.

A linguagem LM comporta, ainda, as estruturas *procedimento* e *função*, que facilitam o agrupamento de sequências de instruções, susceptíveis de execução frequente, bem como, a escrita e leitura de programas.

Um programa escrito em linguagem LM contém um ou vários módulos, chamados *unidades de compilação*. Estas unidades de compilação decompõem-se numa unidade obrigatória (programa principal) e em unidades não obrigatórias (procedimentos).

A linguagem LM permite utilizar ainda três tipos de dados elementares: booleanos, inteiros e reais. Na parte declarativa são definidas constantes, vectores, ficheiros, procedimentos ou funções. As instruções iterativas WHILE, REPEAT, FOR encontram-se disponíveis, bem como, as instruções condicionais IF THEN ELSE e CASE.

Os dados específicos, usados em Robótica, são tratados pelos tipos FRAME, o tipo VECTOR

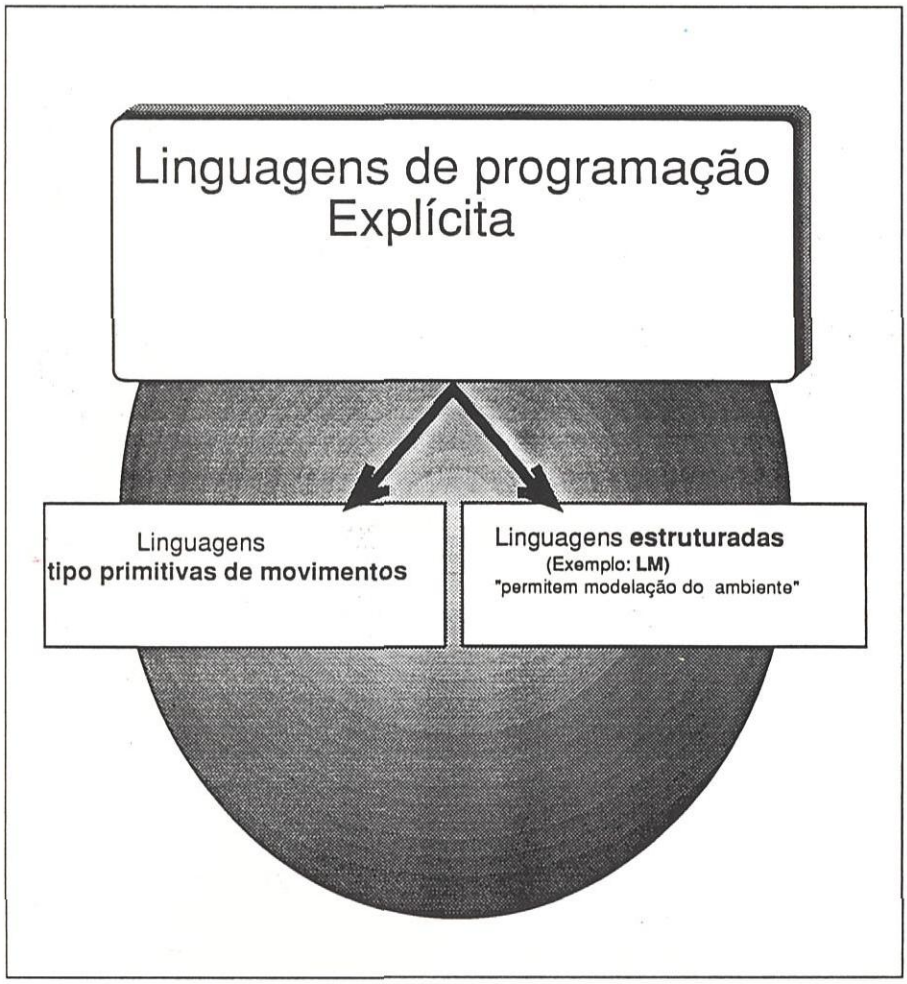


Fig. 7. Divisão das linguagens de programação explícita

e o tipo TRANSFORM. O tipo VECTOR é representado por três reais. Existem também as habituais funções matemáticas e funções standard (ex: produto vectorial, criação de referencial, etc). Um valor do tipo TRANSFORM é um elemento do conjunto dos deslocamentos do espaço geométrico com três dimensões. O valor de uma transformação é representado por doze reais, sob a forma de uma matriz de dimensão 3 por 4, correspondendo à parte rotação (9 reais) e à parte translação (3 reais). A linguagem LM contém operações e funções *standard* para este tipo de dados.

Uma variável do tipo FRAME (referencial) é composta por dois elementos:

- um valor, cuja transformação dá a posição do referencia,l relativamente ao referencial STATION (referencial especial que serve de referência comum ao utilizador e ao LM);

- uma lista, eventualmente vazia, que permite actualizar as ligações entre este e outros referenciais.

É permitida a multiplicação à direita de um referencial por uma transformação R*T (o resultado é um referencial)[7].

4.2. LM face às novas tendências

A linguagm LM não é interpretada, ao contràrio das actuais linguagens que se estão a desenvolver nos principais centros de investigação, [8] ou que temos notado, em algumas implementações industriais. Temos a consciência de que se o utilizador do robot for forçado a repetir continuamente o ciclo **editar - compilar - ligar** *(linking)* **- executar** desta linguagem compilada, a produtividade do programador será baixa, face às linguagens interpretadas, em que, as instruções individuais da linguagem podem ser executadas durante o desenvolvimento do programa e feito o seu correspondente depuramento (*debugging*)[9].

[1] *Professor Latombe e sua equipa (Mazer, Miribel, Dufay, etc) do IMAG em Grenoble.*

[2] *Usamos este termo referencial como equivalente a FRAME (inglês) ou REPERE (francês).*

[3] *Ou da ferramenta no caso de não estarmos a considerar o caso particular da pinça.*

[4] *Funções intrínsecas da linguagem LM. Ex: TRANSLAT (VZ, 10.0) significa uma translação de 10.0 mm ao longo do eixo Z.*

Ver «LM - Linguagem de manipuladores» pág. 3.3 .

[5] *Ver LM - Linguagem de manipuladores» pág. 3.9 .*

[6] *Ver «LM - Linguagem de manipuladores» pág. 3.4 .*

[7] *Ver «LM - Linguagem de manipuladores» pág. 3.19 .*

[8] *Pese o facto de as termos visto em ambientes de programação embrionários (MIT, LIFIA, LAAS) sem esquecer que a VAL II difundida na indústria está instalada, por exemplo, no robot Puma da F.C.T.U.C.*

[9] *Em contrapartida, para acções repetitivas, o código compilado é, obviamente, mais rápido que o interpretado.*

1990