Revista Iberoamericana de Ingeniería Mecánica. Vol. 8, N.º 3, pp. 33-42, 2004

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA PROGRAMAÇÃO DE ROBÔS COM INTERFACE COM FICHEIROS CAD

J. NORBERTO PIRES [1], T. GODINHO [1], P. FERREIRA [2]

[1] Universidade de Coimbra, Departamento de Engenharia Mecânica

[2] Inst. Superior de Engenharia de Coimbra, Departamento de Engenharia Mecânica

3030-201 Coimbra, Portugal

(Recibido 5 de abril de 2004, para publicación 1 de junio de 2004)

Resumo – Os robôs desempenham actualmente na indústria um papel fundamental, devido à sua grande versatilidade. Assim foi proposto o desenvolvimento de uma aplicação para programação de robôs com interface com ficheiros CAD. Este tipo de solução revela-se muito útil em aplicações do tipo soldadura robotizada, sendo esta a aplicação mais difundida de robôs industriais. Assim, usando software CAD, pode ser desenvolvido em gabinete um pré programa do robô manipulador, bastando para isso que exista o desenho correcto da célula de produção e das peças. O desenhador-projectista deverá apenas respeitar um conjunto de regras simples. O programa final é gerado junto do manipulador fazendo um conjunto de pequenas e rápidas alterações nos parâmetros do pré-programa. Desta forma incrementa-se a versatilidade da célula robotizada, podendo dar resposta às rápidas exigências da produção, que está cada vez mais diversificada e personalizada.

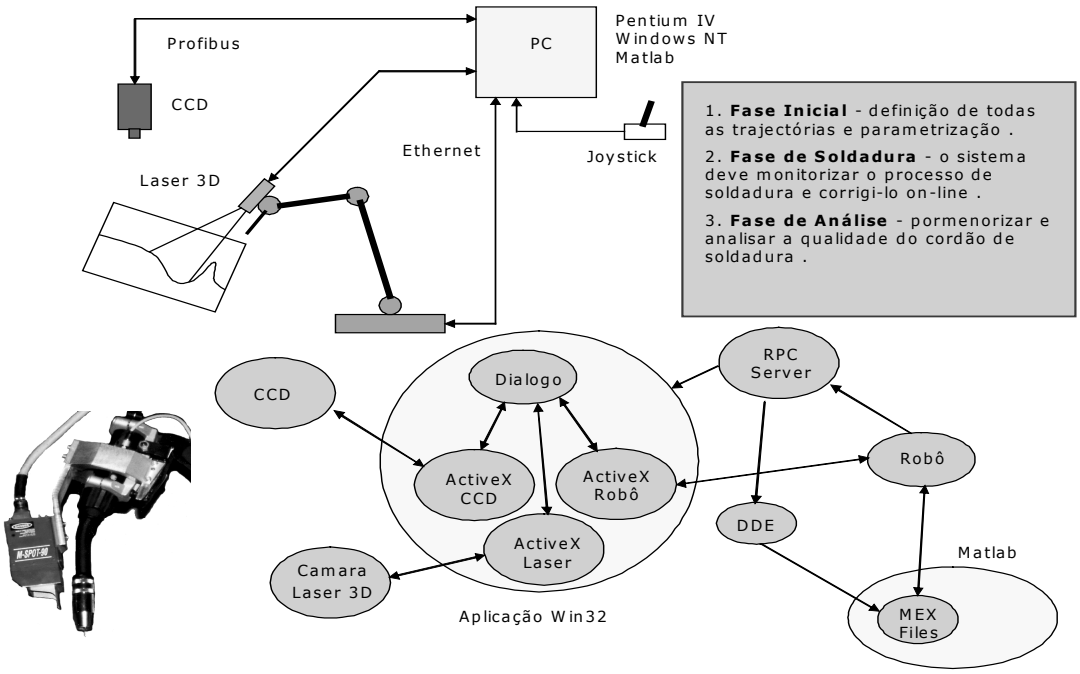
1. INTRODUÇÃO

Os robôs manipuladores industriais são máquinas fascinantes e ao mesmo tempo bastante complexas quer do ponto de vista mecânico e estrutural, quer do ponto de vista eléctrico, electrónico e informático ou até mesmo do ponto de vista de controlo. Pode dizer-se que os robôs são verdadeiros sistemas Mecatrónicos. Apesar disso, apresentam-se ainda como máquinas algo limitadas em termos de programabilidade, controlabilidade e, até em termos de exploração remota, nomeadamente para a utilização em ambientes distribuídos [1-5]. Na era das telecomunicações, da Internet e da comunicação global, os robôs devem ser máquinas de fácil acesso disponibilizando interfaces de comunicação e de programação suficientemente poderosos. A sua utilização deverá ser coordenada, tendo em conta a estrutura produtiva em que se inserem, e não limitada a células de produção mais ou menos isoladas. Neste contexto, a utilização de redes industriais interligando os controladores dos vários robôs de uma determinada instalação produtiva, incluindo autómatos programáveis, controladores de outros sistemas, controladores de robôs móveis, etc., é muito importante.

Na grande maioria das empresas os produtos são desenvolvidos utilizando uma aplicação de CAD (Computer Aided Design). Seria por isso de esperar que fosse fácil programar as estruturas produtivas utilizando o programa de CAD ou ficheiros de CAD. No caso de soldadura robotizada, estamos interessados em trajectórias sobre a peça. Essas trajectórias podem ser definidas pelo operador, sob o desenho das peças, utilizando layers próprias. Se algumas regras forem definidas, é possível identificar as operações executadas nessas layers e extrair a informação pretendida sob a forma de programas para o robô.

2. SISTEMA UTILIZADO

O sistema desenvolvido (Fig. 1) é composto por um robô industrial e respectivo controlador, uma máquina de soldadura e um computador que permite gerir a aquisição de dados e controlar a posição do robô, conjuntamente com a selecção dos parâmetros de soldadura (de acordo com uma base de dados),



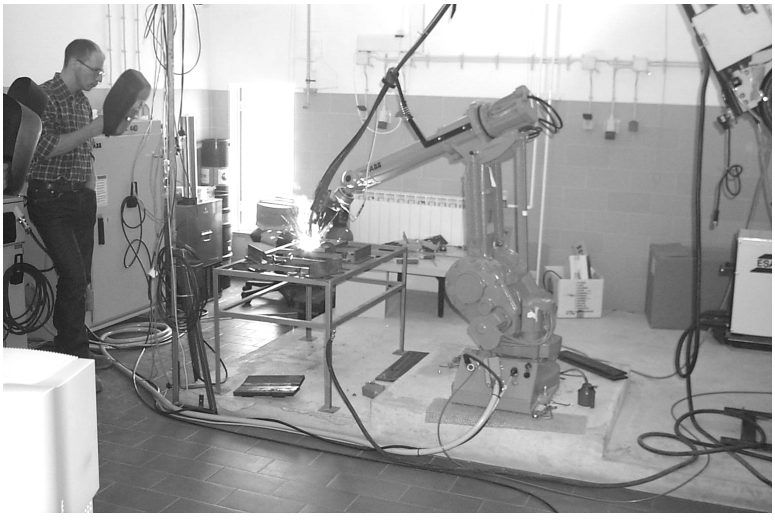


Fig. 1. Sistema de soldadura robotizada para aplicações industriais.

usada para descrever o processo de soldadura. Esta aplicação usa o já referido componente de software PCROB [6-9].

A fonte de alimentação da máquina de soldadura MIG é controlada a partir do robô usando a sequência de soldadura apresentada na Fig. 2, e uma estratégia de programação do tipo cliente servidor. A temporização marcada na figura é configurável na rotina que controla a sequência de sinais da máquina de soldadura, e permite iniciar, monitorizar e terminar o arco eléctrico.

O software do controlador do robô funciona como um servidor, disponibilizando ao cliente uma série de serviços (Fig. 3) que constituem a sua função primária. O robô pode então iniciar ou terminar um procedimento de soldadura, pode ser comandado para seguir trajectórias, simular o processo na sua totalidade ou passo-a-passo. Para isso o utilizador envia ao controlador a definição completa da tarefa de soldadura (pontos, parâmetros de soldadura associados (velocidade, tensão e intensidade), tipo de trajectória e precisão), que a armazena.

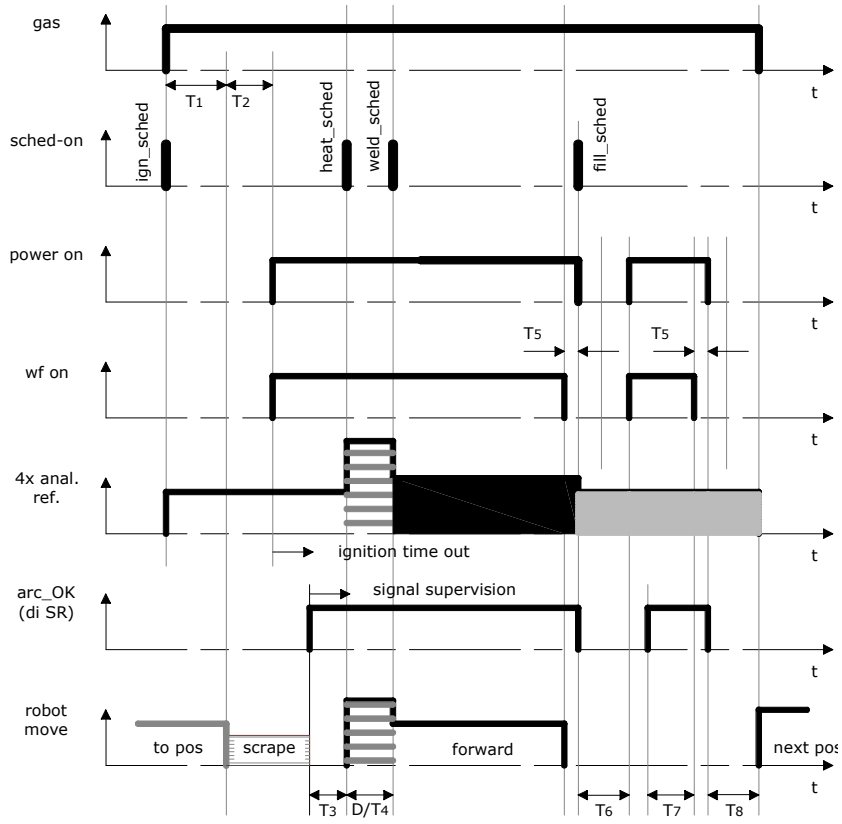


Fig. 2. Sequência de soldadura implementada pelo controlador do robô (todos os tempos são programados pelo utilizador).

```
while never_end;
  switch decision
    case 1: call routine_1; break;
    case 2: call routine_2; break;
    ...
    case n: call routine_n; break;
  end_switch;
end_while;
```

```
PROC weld()
  weldon:=0; i:=1;
  WHILE ((decision1=9401) AND (i<=numberpoints) AND (i>=1)) DO
    weldpoint:=i;
    wd_iref:=trj_voltage{i}; feed_iref:=trj_current{i};
    wd_href:=trj_voltage{i}; feed_href:=trj_current{i};
    wd_ref:=trj_voltage{i}; feed_ref:=trj_current{i};
    IF (trj_type{i}=0) THEN
      weld_on;
      weldon:=1;
    ENDIF
    ppos:=trj{i}; pvel:=trj_vel{i};
    pzone:=trj_prec{i}; ptype:=trj_mode{i};
    move_gen;
    IF (weldon=1) AND ((i+1>numberpoints) OR (trj_type{i+1}=1)) THEN
      weld_off;
      weldon:=0;
    ENDIF
    i:=i+1;
  ENDWHILE
  IF (weldon=1) THEN
    weld_off;
    weldon:=0;
  ENDIF
ENDPROC
```

Figura 3 – Robô a funcionar como servidor.

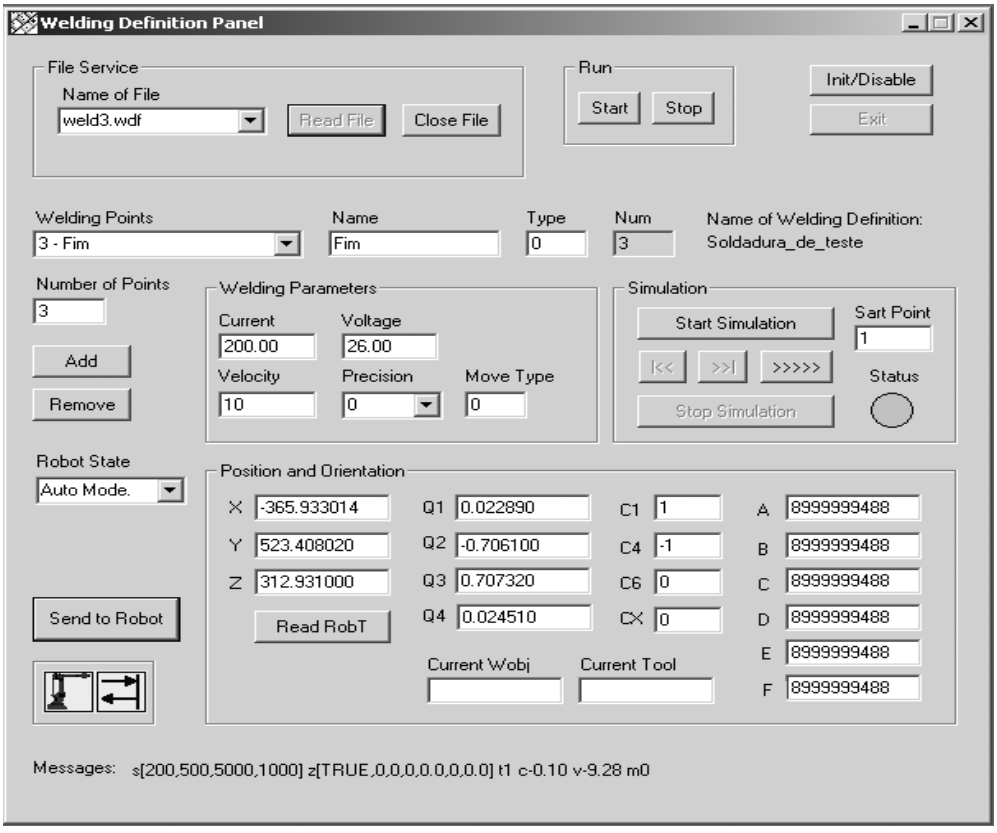


Fig. 4. Painel da definição da soldadura.

2.1. Painel de soldadura

Com esta aplicação (Fig. 4) o utilizador consegue manipular os pontos de soldadura que podem ser inicialmente obtidos a partir de um modelo de CAD da peça que se pretende soldar. Estes pontos podem ser modificados ou ajustados, de forma a evitar colisões e tornar o processo mais eficiente. Todos os pontos são referidos à ponta da tocha de soldadura (Tool Center Point) e descritos relativamente a um sistema de eixos definido na mesa de trabalho. O utilizador pode ajustar os pontos simplesmente através da movimentação do robô para a posição desejada. Essa movimentação pode ser feita através do PC ou transferindo o controlo para a consola portátil do robô, o que, geralmente, é mais prático.

Uma outra função importante é a função de simulação, que permite executar os movimentos exactos programados para a soldadura, sem no entanto ligar a tocha. O operador tem a ideia de que o robô estaria a soldar quando a indicação status (indicação luminosa por software) mudar para vermelho. Esta função é importante e permite verificar todo o programa de soldadura desenvolvido.

Esta ferramenta também recebe eventos do robô, como, por exemplo, mudanças de estado, estado actual do equipamento de soldadura e sinais IO, etc. O estado do programa em execução no controlador e a ligação via rede são constantemente monitorizados, por forma a evitar danos quer no equipamento, quer nas pessoas. Em situações de erro são bloqueados todos os comandos remotos operacionais. Os eventos são chamadas RPC efectuados pelo controlador do robô para um servidor RPC [11] que se encontra a correr no PC.

2.2. Painel de ajuste

Com esta ferramenta (Fig. 5) é possível fazer o ajuste de pontos on-line e adquirir pontos qualquer que seja a posição/configuração do robô e do estado do respectivo programa. Basicamente é uma aplicação

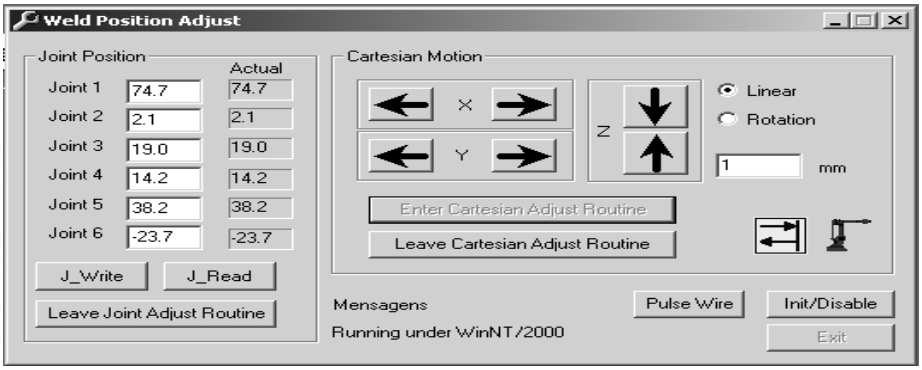


Fig. 5. Painel para ajuste de posições.

que permite ao utilizador a movimentação do robô a partir do PC, usando comandos de movimento cartesianos X, Y, Z ou movimentos de juntas.

Como exemplo do tipo de programação usado, apresenta-se de seguida o código de duas das funções de programa da fig. 5.

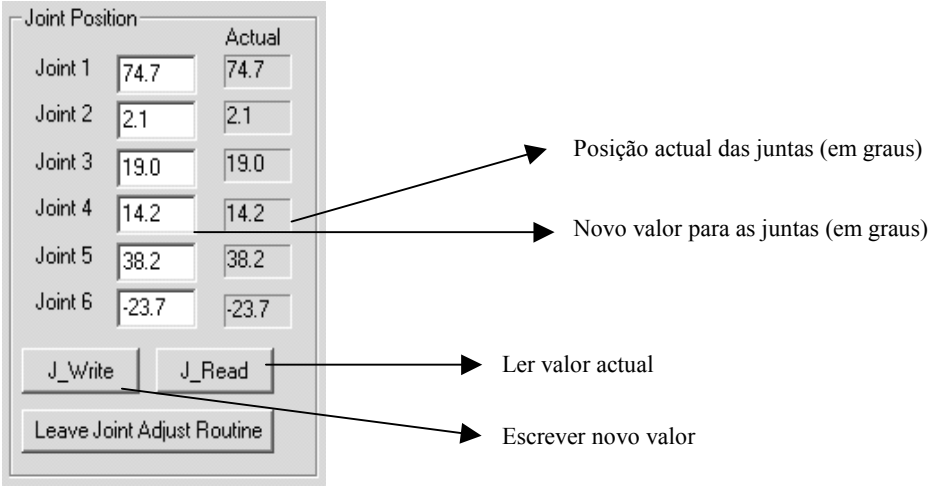
Pulsar arame na tocha

Esta função corresponde a pulsar uma saída digital (doFEED) que acciona por instantes o motor que controla o arame da tocha. Antes de pulsar o arame, é necessário regular a velocidade de avanço de arame, usando uma saída analógica (aoFEED_REF) cujo valor é proporcional à velocidade de avanço.

```
void CWeldadjustDlg::Onpulse()
{
        float value=0.25;
        long valuer;
        nResult = m_pon.WriteAnalog(LPCTSTR("aoFEED_REF"),&value);
        if (nResult <0) m_msg.SetWindowText("Failled to update anlog reference.");
        valuer=1;
        nResult = m_pon.WriteDigital(LPCTSTR("doFEED"),&valuer);
        if (nResult <0) m_msg.SetWindowText("Failled to write digital output.");
        Sleep(500);
        valuer=0;
        nResult = m_pon.WriteDigital(LPCTSTR("doFEED"),&valuer);
        if (nResult <0) m_msg.SetWindowText("Failled to write digital output.");

}
```

Movimentação do robô especificando os ângulos das juntas.



Rotina em RAPID a correr no controlador do robô

```
    IF decision1 = 9101 THEN
            WHILE decision1 > 0 DO
                    MoveAbsJ joints_now, velc, zone, toolt;
            ENDWHILE
            decision1 := 123;
   ENDIF
```

Acesso à rotina de serviço.

Rotinas correspondentes à função "Escrever novo valor" do menu anterior (C++)

```
void CWeldadjustDlg::Onjwrite()
{  VARIANT var; BOOL all_done;  float warn[12]; float value =9101;  long i; CString msg;
 SAFEARRAY FAR *psa; SAFEARRAYBOUND rgsabound[1];
 if (!on_cycle)
 {    nResult = m_pon.WriteNum(LPCTSTR("decision1"),&value);
          if (nResult <0) m_msg.SetWindowText("Failled to enter routine.");  on_cycle = TRUE;
          m_leavej.EnableWindow(TRUE);
          CWeldadjustDlg::Onjread(); return;  }
 m_j1.GetWindowText(msg); if (msg.GetLength()) warn[0] = (float) atof(msg);
 m_j2.GetWindowText(msg); if (msg.GetLength()) warn[1] = (float) atof(msg);
 m_j3.GetWindowText(msg); if (msg.GetLength()) warn[2] = (float) atof(msg);
 m_j4.GetWindowText(msg); if (msg.GetLength()) warn[3] = (float) atof(msg);
 m_j5.GetWindowText(msg); if (msg.GetLength()) warn[4] = (float) atof(msg);
 m_j6.GetWindowText(msg); if (msg.GetLength()) warn[5] = (float) atof(msg);
 warn[6] = (float)9e9; warn[7] = (float)9e9;
 warn[8] = (float)9e9; warn[9] = (float)9e9;
 warn[10] = (float)9e9; warn[11] = (float)9e9;
 all_done = FALSE; VariantInit(&var);
 rgsabound[0].lLbound = 0; rgsabound[0].cElements = 12;
 var.vt = VT_ERROR; psa = SafeArrayCreate(VT_R4, 1, rgsabound);
 for (i=0;i<=11;i++)
 {    var.fltVal = warn[i];    if (SafeArrayPutElement(psa, &i, &var.fltVal) != S_OK) all_done =
TRUE; }
 var.parray = psa;   if (!all_done) {var.vt = VT_ARRAY | VT_R4; SafeArrayLock(psa);
m_pon.WriteJointTarget("joints_now",&var);}  else AfxMessageBox("Ooops!");
VariantClear(&var);}
```

Escreve novo valor na variável de valores de juntas "joints_now".

Nota: como forma de passar matrizes de dados, utilizam-se neste exemplo *safearrays* que são transferidas usando o tipo VARIANT (ver detalhes na documentação do Microsoft Visual C++).

3. APLICAÇÃO DESENVOLVIDA

O trabalho desenvolvido consistiu no desenvolvimento uma aplicação para programação de robôs com interface para ficheiros CAD (ver figura 6). Este tipo de aplicação revelou-se muito útil em aplicações de soldadura robotizada, que é de longe a aplicação mais difundida em robôs **[1]**. Revelou-se também muito útil em tarefas de montagem de componentes mecânicos, seguimento de arestas, lixagem, rebarbagem, polimento de peças, aliado a aplicações de controlo de força [3]. Assim, usando software CAD, pode ser desenvolvido em gabinete um pré programa do robô manipulador, bastando para isso, que exista o

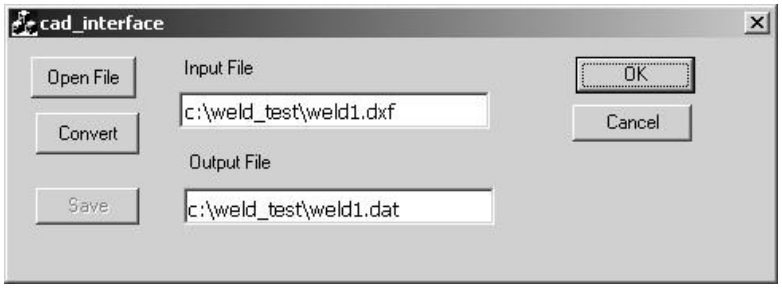


Fig. 6. Painel de interface de ficheiros CAD.

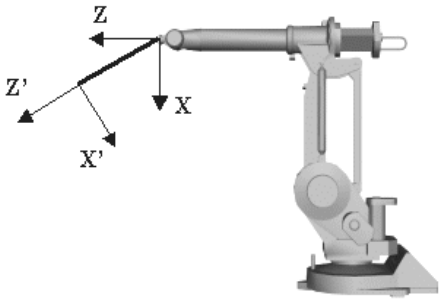


Fig. 7. A direcção do sistema de coordenadas.

desenho correcto da célula de produção e das peças a maquinar. O desenhador-projectista deverá apenas respeitar um conjunto de regras simples, como por exemplo a peça a ser soldada deverá ser feita em 3D e posicionada na mesa de trabalho. Todos os percursos que o robô terá de efectuar serão descritos por layers (ver figura 3) e no final de cada ponto terá de existir o sistema de orientação que está no centro da ferramenta. O programa final é gerado junto do manipulador fazendo um conjunto de pequenas e rápidas alterações nos parâmetros do pré-programa. Desta forma incrementa-se a versatilidade da célula robotizada, podendo dar resposta às rápidas exigências da produção, que está cada vez mais diversificada e personalizada.

Sendo assim optou-se por definir as trajectórias de soldadura de uma forma simples, de fácil configuração e generalista. Tendo no entanto que definir todos os parâmetros necessários para definir o tipo de trajectória pretendida. Como cada trajectória é definida por um ponto de início e de fim de movimento e de forma a simplificar a operação de programação de movimentos, considera-se que o ponto de início de trajectória, é a localização actual do robô tendo que o utilizador definir o ponto de fim de trajectória. Ao definir o ponto de fim de trajectória, o utilizador além das coordenadas desse ponto, define a velocidade movimento, a precisão com que esse movimento se irá efectuar, bem como, o tipo de trajectória (linear, juntas ou circular), e a operação a efectuar durante o movimento (passagem ou soldadura).

Quando se pretende definir uma trajectória de soldadura é também necessário definir os parâmetros com que este processo irá ser efectuado (tensão, corrente). A definição dos outros parâmetros de soldadura, a velocidade, distância do eléctrodo à peça, orientação da tocha em relação à peça é feita a quando da escolha do movimento a efectuar.

O sistema de orientação por defeito está localizado no centro do eixo 6 do robô. No entanto como o habitual é ter-se uma *tool* associada a esse mesmo eixo, o aconselhado é ter-se o *TCP (Tool Center Point)* no centro da nossa ferramenta. A orientação do sistema de coordenadas é descrita por uma matriz de rotação, que define a direcção do sistema de coordenadas pretendido em relação ao sistema de coordenadas de referência (figura 7). Os eixos do sistema de coordenadas rodado (x, y, z), são vectores que podem ser expressos em relação ao sistema de coordenadas de referência, por:

$$\begin{aligned} x &= (x_1, x_2, x_3) \\ y &= (y_1, y_2, y_3) \\ z &= (z_1, z_2, z_3) \end{aligned} \tag{1}$$

Isto significa que, a componente x do vector x em relação ao sistema de coordenadas de referência vai ser x_1, a componente do y vai ser x_2, etc. Estes vectores podem ser agrupados numa matriz, matriz de rotação, na qual os vectores formam uma das colunas:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \tag{2}$$

Para descrever de uma forma mais concisa, esta matriz de rotação, podemos calcular os seguintes quaternios (baseados nos elementos da matriz de rotação):

$$
\begin{aligned}
q_1 &= \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2} \\
q_2 &= \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2} \\
q_3 &= \frac{\sqrt{-x_1 + y_2 - z_3 + 1}}{2} \\
q_4 &= \frac{\sqrt{-x_1 - y_2 + z_3 + 1}}{2}
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
\text{sinal}\, q_2 &= \text{sinal}\left(y_3 - z_2\right) \\
\text{sinal}\, q_3 &= \text{sinal}\left(z_1 - x_3\right) \\
\text{sinal}\, q_3 &= \text{sinal}\left(x_2 - y_1\right)
\end{aligned}
\tag{4}
$$

4. ENSAIO

Neste exemplo mostra-se como se efectua o planeamento do ficheiro de CAD de uma soldadura de uma peça estrutural. De forma a não complicar o desenho só se apresenta uma trajectória de soldadura e a respectiva orientação da tocha. Na Figura 8 mostra-se a peça de trabalho, que é composta por chapas de 4 mm, montadas conforme se mostra na figura. A origem do sistema de eixos está localizada no centro da raiz da soldadura.

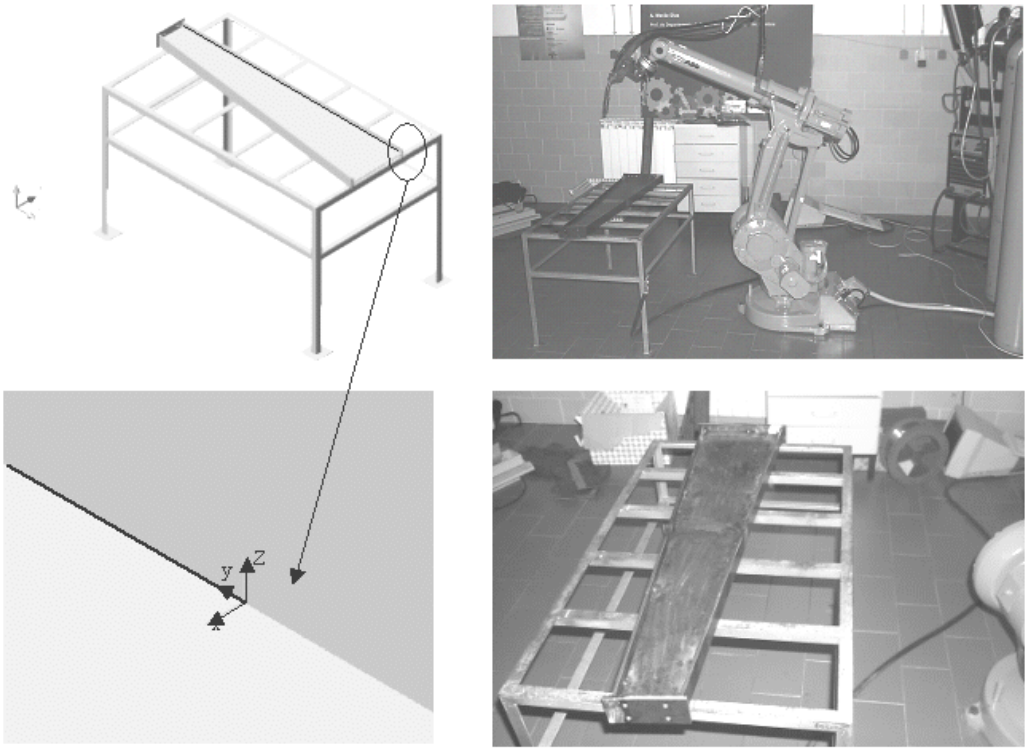


Fig. 8. Aspecto do objecto de trabalho e da peça que se pretende soldar.

Tab. 1. Exemplo de um ficheiro de soldadura.

	Soldadura_de_teste	Nome do ficheiro
	4	Número de pontos definidos no ficheiro
Definição do ponto 1	Ponto 1	Ponto número x
	Origem	Nome do ponto
	1	Tipo de ponto (soldadura – 0, passagem - 1)
	656.419922	Coordenada x do ponto
	-444.451813	Coordenada y do ponto
	730.853149	Coordenada z do ponto
	0.091980	Quaterno q1 do ponto
	0.001690	Quaterno q2 do ponto
	0.995760	Quaterno q3 do ponto
	0.002070	Quaterno q4 do ponto
	0	Quadrante cf1 do ponto
	-1	Quadrante cf4 do ponto
	0	Quadrante cf6 do ponto
	0	Quadrante cfx do ponto
	8999999488	Eixo externo ex1 do ponto
	8999999488	Eixo externo ex2 do ponto
	8999999488	Eixo externo ex3 do ponto
	8999999488	Eixo externo ex4 do ponto
	8999999488	Eixo externo ex5 do ponto
	8999999488	Eixo externo ex6 do ponto
	0.00	Intensidade [A]
	0.00	Tensão [V]
	100	Velocidade [mm/s]
	5	Precisão
	0	Tipo de movimento (linear – 0, circular – 1, juntas - 2)
	...	
Definição do ponto 4	Ponto 4	
	Fim	
	1	
	684.311096	
	-443.820709	
	581.514465	
	0.092050	
	0.001700	
	0.995750	
	0.002130	
	0	
	-1	
	0	
	0	
	8999999488	
	8999999488	
	8999999488	
	8999999488	
	8999999488	
	8999999488	
	0.00	
	0.00	
	10	
	5	
	0	

Na tabela 1, apresenta-se a configuração completa da trajectória para um dois pontos de soldadura, que constam num ficheiro de teste.

5. CONCLUSÃO

O caso apresentado neste artigo mostra a utilidade oferecida pelo sistema desenvolvido. Obter um programa de robô completo a partir de um ficheiro CAD é possível e relativamente simples de implementar. Exige algumas regras e é algo dependente do tipo de acesso que existe ao robô manipulador. Desta forma, a solução apresentada não é geral e está algo dependente do tipo de máquina usada, mas o princípio foi demonstrado e pode ser estendido a outros casos.

Os desenvolvimentos práticos efectuados durante a elaboração deste trabalho mostram que a robotização é óptima em determinados processos como por exemplo a soldadura. A utilização de robôs no processo de soldadura constitui a aplicação mais difundida e a solução mais rentável.

REFERÊNCIAS

[1] United Nations and International Federation of Robots, "Wold Industrial Robots 1996: Statistics and Forecasts", *New York: ONU/IFR*, 2000.

[2] G Bolmsjö., M. Olsson, G. Nikoleris, and K. Brinkm, "Task programming of welding robots", *In Proceedings of the Int. Conf. on the Joining of Materials*, JOM-7, pages 573-585, May-June 1995.

[3] J. N. Pires, "Realização de Controlo de Força em Robôs Manipuladores Industriais", *Tese de Doutoramento*, Coimbra, 1999.

[4] J. Sá da Costa e J. N. Pires, "Future Welding Robot Developments", *Journal Robótica*, n.º 41, January, 2001.

[5] J. N. Pires, A. Loureiro, T. Godinho, P. Ferreira, B. Fernando, J. Morgado, "Object Oriented and Distributed Software Applied to Industrial Robotic Welding", *Industrial Robot: An International Journal*, pp. 149-162, Volume 29, Number 2, 2002.

[6] J. N. Pires, J. M. G. Sá da Costa, "Object Oriented and Distributed Approach for Programming Robotic Manufacturing Cells", IFAC Journal on Robotics and Computer Integrated Manufacturing, February 2000.

[7] J. N. Pires, "Object-oriented and distributed programming of robotic and automation equipment", Industrial Robot, An International Journal, MCB University Press, July 2000.

[8] J. N. Pires, "Interfacing Robotic and Automation Equipment with Matlab", IEEE Robotics and Automation Magazine, September 2000.

[9] J. N. Pires, T. Godinho, P. Ferreira, "CAD interface for automatic robot welding programming", Sensor Review Journal, MCB University Press, July 2002.

[10] J. N. Pires, "Force/torque sensing applied to industrial robotic deburring", Sensor Review Journal, MCB University Press, July 2002.

[11] RAP, Service Protocol Definition, ABB Flexible Automation, 1996.

ROBOT PROGRAMMING USING CAD FILES

Abstract – Industrial robots play an important roll in industry, due to their flexibility. It is common to have detailed data in CAD files, which is not used satisfactorily. In this paper we propose an application capable of extracting robot motion information from the CAD data. The motion information must be added by the user in the CAD file, defining the way the robot should approach parts, should handle them and finish the job. The extracted information is considered a pre-program that should be tuned with the real robot using a small set of rules. In this way versatility of robotic cells is improved and its programming is simplified.