Electrónica e Telecomunicações

universidade de aveiro

theoria poiesis praxis

AVEIRO • SET. • 95 • VOL. 1 • Nº 4

Revista do **D**epartamento de **E**lectrónica e Telecomunicações da **U**niversidade de **A**veiro

Electrónica e Telecomunicações

Revista do Departamento de
Electrónica e Telecomunicações
da Universidade de Aveiro

Editores:

Francisco Vaz
José Luis Oliveira

Morada e Secretariado:

Departamento de Electrónica
e Telecomunicações
Universidade de Aveiro
Campo Universitário
3800 AVEIRO
Portugal

Artes Gráficas :

Sérgio Cabaço

Impressão e Acabamentos :

Gratigamelas, Ind. Gráf.Lda.

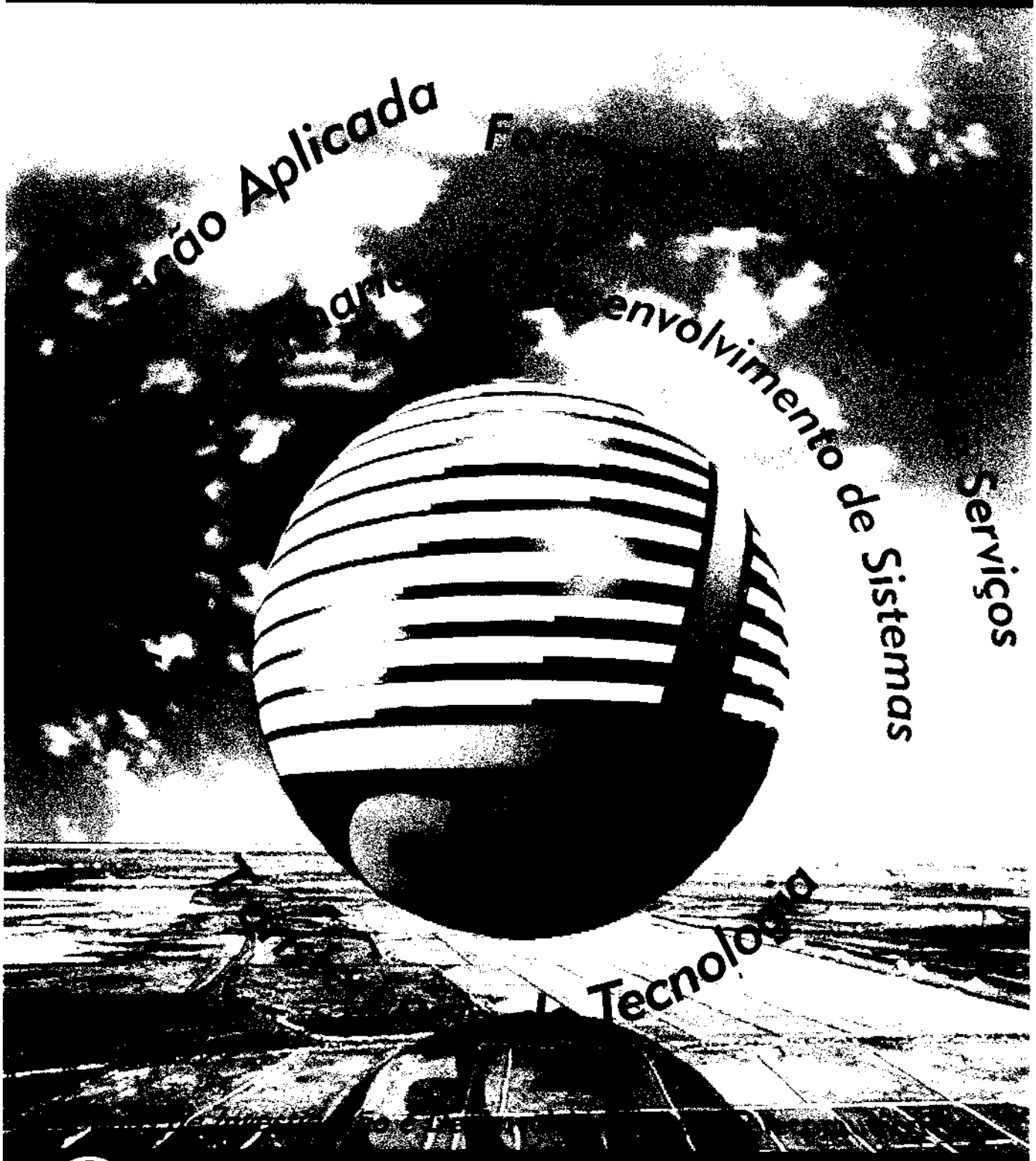
Tiragem : 600 exemplares

Editorial

A publicação do quarto número da revista Electrónica e Telecomunicações é a confirmação de que alguns dos objectivos iniciais foram atingidos. Com efeito os membros do corpo docente e discente do DETUA habituaram-se a usar esta revista para comunicarem à comunidade científica e profissional os resultados das suas actividades pedagógicas e de investigação. O número de artigos recebidos permitiu esta edição e, muito provalvelmente, um novo número a breve prazo.

Para os responsáveis pela edição existe, no entanto, um objectivo que está longe de estar alcançado: Uma maior difusão da revista. A venda continua a ser fundamentalmente restrita ao DETUA e um esforço considerável deverá ser feito por forma a interessar profissionais e empresas com actividade nesta área. É este o grande desafio que se coloca para o futuro.

CENTRO DE ESTUDOS DE TELECOMUNICAÇÕES



cet
CENTRO DE ESTUDOS DE TELECOMUNICAÇÕES

PORTUGAL TELECOM

Rua Engº José Ferreira Pinto Basto - 3810 AVEIRO - Telefone: 034-381831 - Fax: 034-24723 - E-mail: cet@cet.pt

Índice

Determinação da Velocidade do Sangue na Artéria Pulmonar com Imagens Cintigráficas de Primeira Passagem

J.P. Saro*, J.A. Rafael*, J.P. Lima**, M.F. Botelho**
*INESC - DETUA, **IBILI - FMUC

Resumo- **Descreve-se uma nova metodologia para a determinação da velocidade do sangue na artéria pulmonar, com o recurso a imagens cintigráficas de primeira passagem. O objectivo final deste trabalho, é procurar estabelecer uma correlação entre essa informação e a existência de Hiperpressão Pulmonar em pacientes. Após uma abordagem sobre esta patologia e sobre as características das imagens de primeira passagem, segue-se a descrição da metodologia desenvolvida, apresentando-se por fim o resultado da aplicação do método sobre um modelo físico e um exame real.**

Abstract- **A new methodology to compute the pulmonary artery blood velocity using first pass scintigraphic images is presented. This project final goal, is the establishment of one correlation between pulmonary artery blood velocity and Pulmonary Hiperpressure. After a short approach about this pathology and the first pass images, the developed methodology is presented, as well the results obtained with one physical model and one real exam.**

I. INTRODUÇÃO

A hiperpressão pulmonar resulta da existência de valores elevados de pressão sanguínea no percurso que vai desde o lado direito do coração até à barreira alvéolo-capilar [1]. Estes anormais valores de pressão podem ser devidos a comunicações estabelecidas entre ambos os lados do coração ou até mesmo entre a crossa da aorta e a artéria pulmonar. Estas comunicações serão então não só responsáveis pela mistura de sangue arterial e venoso, como também pelo aumento dos valores de pressão sanguínea, visto que o ventrículo esquerdo bombeia o sangue com pressões 15 a 20 vezes (em média) superiores ao ventrículo direito [2].

O organismo tenta adaptar-se a esta anomalia com o hiper-desenvolvimento de fibras elásticas e musculares nas paredes dos vasos sanguíneos, para que estes tenham suficiente complacência para suportar as fortes ondas de pressão. Em casos muito adiantados, a própria barreira alvéolo-capilar aumenta de espessura, dificultando a transferência de fluidos.

Não se tratando de uma patologia muito comum, o facto é que esta se repercute imenso no quotidiano dos seus portadores devido à sua grande morbilidade. Esta patologia quando se manifesta em crianças com malformações congénitas, está por si associada a grande mortalidade quando segue a sua evolução normal.

A hiperpressão pulmonar pode ser detectada através da medição de parâmetros hemodinâmicos na artéria pulmonar (velocidade, pressão, etc.), no entanto isso apenas tem sido realizável com cateterismo. Presentemente, recorre-se a uma biópsia pulmonar para um diagnóstico de certeza de hiperpressão pulmonar. Este método ao detectar a presença de fibras elásticas e musculares em vasos de diâmetro inferior a 30 μm, permite diagnosticar a existência desta patologia. Não existe assim ao dispor do corpo clínico, uma técnica não invasiva que permita não só a detecção precoce da hiperpressão pulmonar, como também o acompanhamento clínico do paciente, e preferencialmente, com o máximo de conforto e segurança para este último.

As imagens obtidas por intermédio de câmaras de raios gama após a injecção intravenosa de um traçador radioactivo no braço do paciente, permitem seguir a evolução do produto desde a sua introdução, até ao seu espalhamento nos pulmões após a sua passagem pelo coração. Apesar da pobre resolução espacial destas imagens, quando comparadas com as obtidas por outras técnicas, como por exemplo as de Tomografia Axial Computorizada ou as de Ressonância Magnética, o facto é que as imagens planares da Medicina Nuclear conseguem fornecer uma incomparável informação funcional. O trabalho que se apresenta procura a determinação da velocidade do sangue na artéria pulmonar a partir da informação funcional contida na sequência de imagens cintigráficas, protagonizada pela evolução do traçador radioactivo.

II. MATERIAIS E MÉTODOS

A. *Exames de Primeira Passagem*

Nos exames estudados, a aquisição de imagens é feita com o recurso a uma Câmara de Raios Gama, após a injecção intravenosa de um traçador radioactivo, isto antes de qualquer recireulação do traçador, pelo que se pode observar a sua evolução desde o braço até aos pulmões ("primeira passagem") com um máximo de contraste, visto

que ainda não existe qualquer actividade de fundo nas imagens da sequência.

No que diz respeito ao formato de aquisição dos exames, houve necessidade de algumas soluções de compromisso devido à própria natureza do problema e às características do sistema de aquisição. Note-se que se pretende a detecção de velocidades numa zona bastante pequena (artéria pulmonar: 5-6 cm de comprimento) e percorrida pela cabeça do traçador em tempos na ordem dos 0.5 segundos (11.3 cm/s). Para estes valores, imagens de 128 por 128 ou de 256 por 256 pixel's conduziriam a um baixo nível de contagens (radiação detectada) por pixel e consequentemente a um elevado ruído estatístico. Adquiriram-se então imagens de 64 por 64 pixel's para tempos de aquisição por imagem entre os 0.1 (limite inferior permitido pelo equipamento) e 0.3 segundos, resoluções que permitem ainda a determinação de velocidades na zona de interesse. A fig. 2.1 pretende realçar as características muito próprias destas imagens, com especial destaque para a pobre informação morfológica contida na sequência. A seta indica a zona da artéria pulmonar onde se pretende quantificar o movimento.

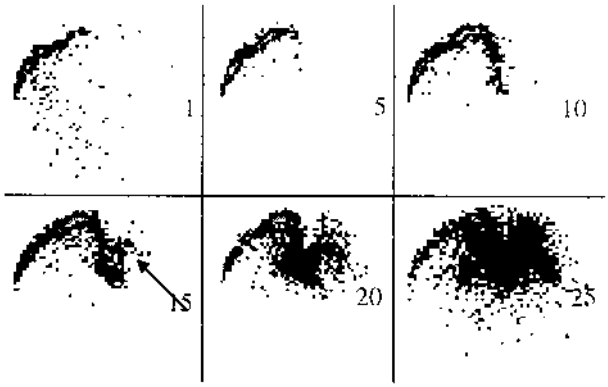


Fig. 2.1 - Sequência de imagens pertencentes a um exame real.

B. Imagem Funcional de Máximas Derivadas (IFMD)

A pobre resolução espacial destas imagens e o facto do objecto a identificar não manter uma forma constante no período de tempo em análise, dificulta de sobremaneira a aplicação de comuns técnicas de processamento de imagem para a detecção e quantificação de movimento. Várias experiências foram realizadas utilizando diferentes técnicas, tais como, diferenças de imagens acumuladas, segmentações, análises espectrais e filtragens espaciais, no entanto mostraram-se inconsequentes na quantificação do movimento do traçador [3]. Assim orientámos o trabalho sobre o aspecto funcional intrínseco a estas imagens, tendo-se realizado um conjunto de estudos funcionais, que visaram a detecção de algum padrão que pudesse conduzir à quantificação do movimento do traçador radioactivo [3].

A partir de certa altura resolvemos prestar uma particular atenção às curvas de actividade no tempo (CAT) para cada pixel. Vários estudos permitiram notar que os pixel's que se encontram no percurso feito pelo traçador radioactivo desde o braço até aos pulmões, apresentam tipicamente CAT's conforme o gráfico 2.1.

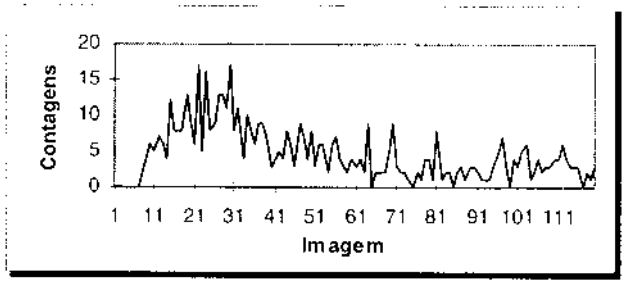


Gráfico 2.1 - Curva típica da actividade no tempo de um pixel (CAT).

Verifica-se portanto ao longo do tempo (i.e. das imagens), uma rápida subida nos valores de actividade seguindo-se uma gradual diminuição destes. Como o traçador se apresenta como um objecto divisível, onde a forma e a intensidade (contagens) variam de imagem para imagem, procuramos encontrar uma relação entre a "frente" ou "cabeça" do traçador e as CAT's dos pixel's. Verificou-se de facto que o instante de tempo correspondente à máxima variação positiva de actividade (MVPA) num pixel está relacionado com a passagem da "cabeça" do traçador [3]. Pode-se assim identificar a posição da cabeça do traçador para um determinado instante, procurando quais os pixel's cuja máxima variação positiva de actividade ocorreu no instante em causa.

No entanto, a determinação do instante de tempo correspondente à MVPA a partir de CAT's com tanta variabilidade (ruído) não se mostra muito fácil. Note-se que a detecção de contagens tem associada uma incerteza estatística que segue um processo de Poisson. A primeira solução utilizada [4], recorria a um ajuste polinomial de ordem 10 para as CAT's, sendo depois muito fácil obter-se o instante da MVPA, bastando procurar o valor máximo da derivada do polinómio. Nalguns exames disponibilizados mais tarde comprovou-se que esta ordem dos polinómios nem sempre conseguia atingir os objectivos. A presente metodologia aplica uma Transformada de Fourier sobre as CAT's, filtra os harmónicos de amplitude inferior a 10% do valor máximo, e em seguida calcula a Transformada inversa, mas da derivada da série de Fourier. O valor máximo desta curva indica imediatamente o instante de tempo correspondente à MVPA. Este método tem apresentado bons resultados em todos os exames. Os gráficos 2.2 e 2.3 apresentam como exemplo, o ajuste a uma CAT de um pixel, e a correspondente derivada.

Para representar a informação das MVPA's de todos os pixel's recorreu-se a uma imagem funcional (paramétrica) [5][6] da mesma dimensão das imagens dos exames, mas onde cada posição representará agora, não contagens, mas sim os instantes de tempo correspondentes às MVPA's dos respectivos pixel's. Daqui resultará uma representação espacio-temporal da cabeça do traçador radioactivo.

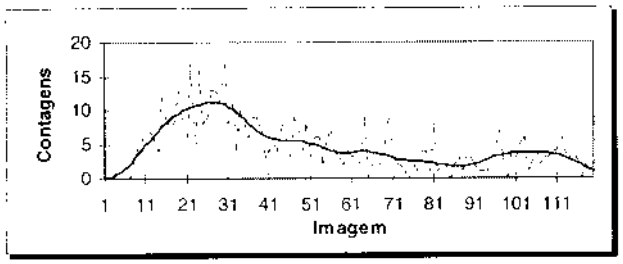


Gráfico 2.2 - CAT de um pixel, e o respectivo "ajuste" com traço contínuo.

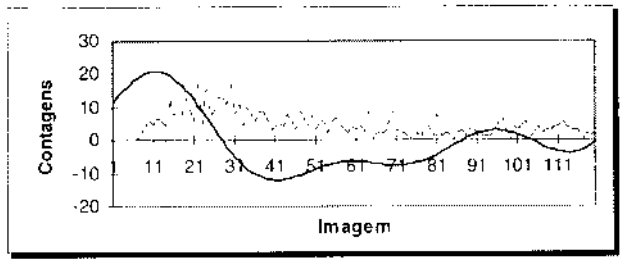


Gráfico 2.3 - CAT de um pixel, e a derivada da curva de "ajuste" com traço contínuo.

Fig. 2.2 - Imagem Funcional de Máximas Derivadas.

Para facilitar a visualização de cada instante temporal na imagem funcional, associou-se univocamente uma cor a cada instante. A fig. 2.2 apresenta uma imagem funcional de máximas derivadas (IFMD) obtida a partir de um exame de 120 imagens de 64 por 64 pixel's, e com um tempo de aquisição por imagem de 0.2 segundos. Esta imagem funcional apresenta grupos ou "regiões de instantes", que representam a posição da cabeça do traçador nesse particular instante. Uma análise atenta permite notar várias estruturas, como por exemplo, o percurso desde o braço até ao coração, a artéria pulmonar, os pulmões, parte de crossa da aorta e até o percurso descendente da aorta. Quanto à paleta de cores utilizada, esta poderá ser modificada permitindo uma melhor discriminação de estruturas.

C. Determinação de Velocidades

Para a determinação das velocidades, optámos pela substituição de cada "região temporal" por um só ponto. Desta forma cria-se uma outra imagem funcional, mas agora com uma sequência de pontos, onde cada um destes representa a posição espacio-temporal da cabeça do traçador no instante respectivo. O cálculo destes pontos pode ser feito de várias formas. No entanto o processo que conduziu a melhores resultados socorre-se de uma imagem soma, introduzindo no cálculo informação quantitativa da actividade, i.e., somam-se todas as imagens do exame obtendo-se uma imagem que fornece através das zonas de maior actividade o percurso seguido pelo traçador. De seguida calcula-se (1) o "centro de massa" [7] de cada "região temporal" que actua como uma máscara sobre a imagem soma.

$$x_{CM} = \frac{\sum_{i,j} \text{pixel}(i,j) * i}{\text{contagens}}$$
$$y_{CM} = \frac{\sum_{i,j} \text{pixel}(i,j) * j}{\text{contagens}} \qquad (1)$$

Obtida a imagem funcional com a sequência de "centros de massa" e como se conhecem as resoluções espacial e temporal das imagens, mais não há a fazer do que determinar a distância entre os sucessivos pontos e proceder aos respectivos cálculos de velocidades, que poderão ser facilmente visualizados graficamente.

III. RESULTADOS

Uma vez conseguido o desenvolvimento da metodologia, testámos a sua capacidade de medir velocidades. Apresenta-se de seguida os resultados da aplicação da metodologia sobre um modelo físico, desenvolvido exclusivamente para efeitos de teste, e também sobre um exame real.

A. Testes com Modelos Físicos

O primeiro modelo desenvolvido no Dep. Biofísica-IBILI em Coimbra, consiste essencialmente em dois tubos de diâmetros diferentes, percorridos por água, e onde por intermédio de uma seringa se introduz o traçador. Uma bomba eléctrica é responsável pela movimentação do líquido com uma velocidade constante. A fig. 3.1 apresenta parte do modelo visível pelo sistema detector de radiação.

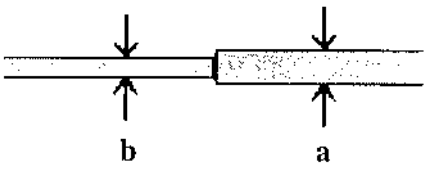


Fig. 3.1 - Parte do modelo físico construído para efeitos de testes visível pela câmara de raios gama.

Admitindo que o movimento do líquido é laminar (esquecendo a turbulência que existe na transição entre tubos), pode-se proceder ao cálculo da relação de velocidades (2) do líquido que será de esperar entre os dois tubos [8][9], uma vez que se conhecem as dimensões destes. Assim, teremos:

r_b = raio do tubo **a**
r_b = raio do tubo **b**
v_a = velocidade no tubo **a**
v_b = velocidade no tubo **b**

$$\frac{V_a}{V_b} = \left(\frac{r_b}{r_a}\right)^2 \qquad (2)$$

As dimensões dos tubos fazem esperar uma relação de velocidades da ordem de: $v_b = 2.8 * v_a$. Após a obtenção de um exame com este modelo, aplicou-se a metodologia descrita. A fig. 3.2 apresenta a correspondente imagem funcional dos centros de massa. O gráfico 3.1 apresenta as velocidades calculadas a partir da imagem funcional dos centros de massa.

Fig. 3.2 - Imagem funcional com uma sequência de centros de massa.

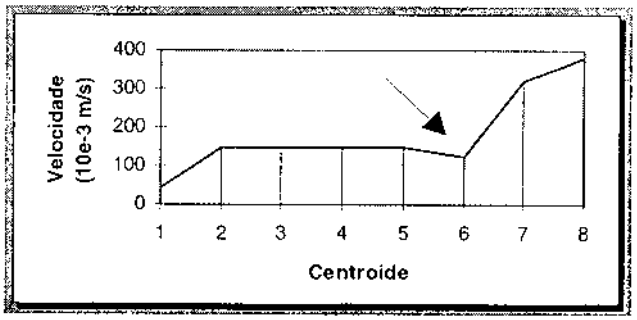


Gráfico 3.1 - Velocidade do traçador no percurso do modelo físico. A seta indica a velocidade na zona da perturbação entre ambos os tubos.

Foi assim possível chegar-se a uma relação de velocidades da ordem de: $v_b = 2.6 * v_a$.

Este valor, muito próximo da relação esperada, foi obtido realizando-se a média das velocidades entre centroides sucessivos e pertencentes ao mesmo tubo.

Desenvolveu-se ainda um segundo modelo, onde se acrescentou um terceiro tubo mais fino a seguir ao tubo **b** do primeiro modelo, tendo sido também introduzidas curvaturas no percurso. Após a observação de alguns exames, notou-se que o traçador passava no tubo mais fino apenas numa imagem, pelo que apenas foi possível o cálculo de velocidades nos tubos mais largos, tendo-se chegado aos mesmos resultados do modelo anterior.

B. Testes com Exames Reais

Os vários exames de "primeira passagem" disponibilizados desde o início da realização deste trabalho, foram obtidos a partir de pacientes que apresentam diferentes patologias daquela que tem sido objecto de referência neste artigo. Houve assim a necessidade de se proceder a uma selecção desses exames, por forma a que fosse possível visualizar-se a aplicação da metodologia sobre a artéria pulmonar, que é afinal a localização do percurso do traçador que mais nos interessa. Apresenta-se assim de seguida a aplicação da metodologia desenvolvida para o cálculo de velocidades, sobre um desses exames.

Fig. 3.2 - À esquerda a imagem funcional de máximas derivadas. À direita a correspondente imagem funcional de centros de massa.

Este exame foi obtido com o recurso a células marcadas, possui 120 imagens de 64 por 64 pixel's, e tem uma resolução temporal de 0.2 segundos. A fig. 3.2 apresenta a imagem funcional de máximas derivadas, e a respectiva imagem funcional dos centros de massa. Nesta última é possível notar-se uma inflexão no percurso indicada por **A**, inflexão essa que também é observável na sequência das imagens. Não se trata portanto de nenhum artefacto introduzido pelo método de determinação dos centroides. As referências **B** e **C** pretendem apenas chamar a atenção para o último ponto que se localiza já nas entradas das artérias pulmonares direita e esquerda.

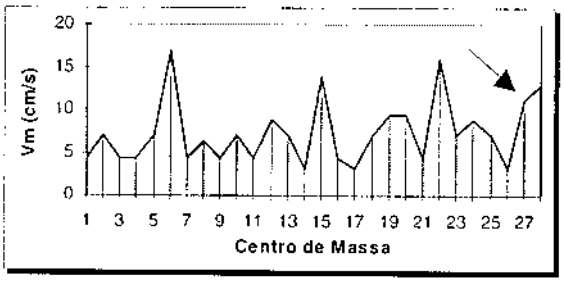


Gráfico 3.2 - Velocidade média do traçador no seu percurso desde o braço até a sua saída na artéria pulmonar. A seta indica a velocidade na artéria pulmonar.

O gráfico 3.2 apresenta também a evolução das velocidades, onde a seta indica o valor da velocidade na artéria pulmonar. Assim a velocidade obtida na região da artéria pulmonar ronda os 11 cm/s.

A consulta de algumas tabelas [10], permitiu verificar que a velocidade média do sangue na artéria pulmonar, ronda os 11.3 cm/s, isto para um indivíduo normal com uma artéria pulmonar com um diâmetro de 3 cm e um comprimento entre os 5 e os 6 cm. As velocidades médias nas cavidades do coração não são fáceis de obter, uma vez que o movimento do sangue nestas localizações é turbulento.

IV. DISCUSSÃO

A aplicação da metodologia desenvolvida de cálculo de velocidades, sobre alguns modelos físicos e exames reais, permitiu demonstrar que esta é capaz de determinar velocidades. O facto de em alguns dos exames estudados os valores das velocidades determinadas na artéria pulmonar serem da ordem de grandeza dos valores reais, mostra-se como uma boa indicação sobre o método.

Este método é então capaz de detectar movimento em imagens possuidoras de muito ruído e onde o objecto em estudo não preserve uma forma constante.

A etapa seguinte deste trabalho, passa pela análise de uma população de exames de indivíduos normais, e de uma outra de indivíduos com hiperpressão pulmonar, de preferência em diferentes estágios de evolução. A existência deste conjunto de exames permitirá a tentativa de validação clínica do método, ao proporcionar uma base de comparação. Concretizando-se a capacidade do método na medição de velocidades, e como se conhecem as características das várias zonas do percurso do traçador, pode-se partir para a determinação de outros parâmetros hemodinâmicos, tais como a aceleração, pressão ou caudal do sangue.

É importante referir ainda que a IFMD permite a fácil segmentação de algumas das estruturas presentes nas imagens de uma forma interactiva ou automática (fig. 4.1), sendo isso conseguido a partir de um histograma da IFMD.

Assim este método pode apresentar os resultados finais de uma forma absolutamente automática e sem necessidade de qualquer selecção de regiões de interesse.

Fig. 4.1 - Exemplo da segmentação de regiões de interesse, realizada a partir da imagem funcional de máximas derivadas.

Todos os estudos foram realizados numa Workstation Silicon Graphics, recorrendo à ferramenta "Explorer". Os algoritmos finais que implementam as funcionalidades descritas neste trabalho, foram integradas numa aplicação de Medicina Nuclear mais geral desenvolvida para ambiente Windows (fig. 4.2).

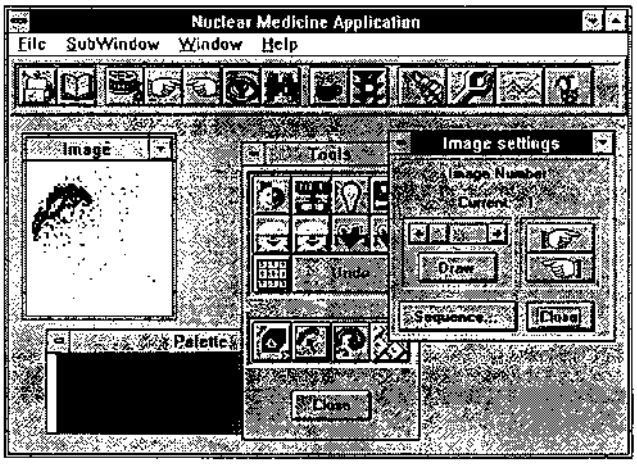


Fig. 4.2 - Aplicação de Medicina Nuclear desenvolvida para ambiente MS-Windows.

AGRADECIMENTOS

Este trabalho foi realizado com o apoio de uma Bolsa de Mestrado no âmbito do Programa CIÊNCIA. Gostaria assim de agradecer a todos aqueles que contribuíram para a realização deste trabalho, e ainda à JNICT, INESC, DETUA e IBILI os meios humanos e materiais disponibilizados.

NOTAÇÃO USADA

CAT = Curva de Actividade no Tempo
IFMD = Imagem Funcional de Máximas Derivadas
MVPA = Máxima Variação Positiva de Actividade

REFERÊNCIAS

[1] Harris P. and Heath D., "The Human Pulmonary Circulation - Its Form and Function in Health and Disease", 3.rd ed., Churchill Livingstone, Edinburgh, 1986.

[2] Altman P.L. and Dittmer D.S. (eds), "Respiration and Circulation" Biological Handbooks, Federation of American Societies for Experimental Biology, Bethesda, Maryland, 1971.

[3] Saro J.P. "Caracterização da Hemodinâmica Pulmonar através de Imagens Cintigráficas de Primeira Passagem", Tese de Mestrado em Eng.º Electrónica e Telecomunicações, Dezembro de 1994.

[4] Saro J.P., Rafael J.A., Lima J.J.P., Botelho M.F., "Estudos Funcionais com Sequências de Imagens de Primeira Passagem", V Congresso Nacional de Medicina Nuclear, Porto, Maio 1994.

[5] Goris M.; Pavel D., "Informatek Users Group - Functional Imaging" - N.º 1/1982 - January.

[6] Bacharach S.L., Green M.V., "A method for objective evaluation of functional images", The Journal of Nuclear Medicine, Vol. 23, Nº 4, 1982.

[7] Ferreira N.C., Lima J.J.P., "Correcção de Movimentos em Imagens de Medicina Nuclear Utilizando Informação de Centros de Massa", V Congresso Nacional de Medicina Nuclear, Porto, Maio 1994.

[8] Patel D.J. and Vaisnav R.N., "Basic Hemodynamics and Its Role in Disease Processes", University Park Press, 1980.

[9] Caro C.G., Pedley T.J., Schoster R.C. and Seed W.A., "The Mechanics of the Circulation", Oxford University Press, Oxford, 1978.

[10] Weibel E.R. et al, "The Lung: Scientific Foundations", Raven Press, Ltd, New York, 1991.

Volume Rendering Based on Oblique Projections

Hubert W.J. Borst Pauwels, Óscar Mealha, Beatriz Sousa Santos, José M.R. Nunes

Resumo - **Este artigo introduz um método novo para visualizar dados representados por voxels utilizando projecções oblíquas. Em vez da abordagem tradicional em que os objectos são rodados e depois ortogonalmente projectados, investigamos a possibilidade de de usar projecções oblíquas para visualizar um objecto de vários pontos do espaço com ganhos significativos em velocidade de cálculo. Provamos que projecções oblíquas conduzem a uma superfície igual à que é gerada quando o objecto é rodado no espaço 3-D em torno dos eixos Z- e X- seguido de projecção ortogonal. Quando desejado a distorção da superfície pode ser retirada com uma transformação geométrica sobre a imagem final. Indicamos como determinamos e exploramos projecções oblíquas específicas com a propriedade de projectar os vertices dos voxels exclusivamente em coordenadas discretas de forma a obter uma representação precisa da superfície.**

Abstract - **This paper introduces a new method for visualization of voxel represented data based on oblique projections. Instead of the traditional approach in which objects are rotated and then orthogonally projected, we investigated the possibility of using oblique projections to view an object under multiple different viewing positions with a speed advantage. We prove that oblique projections will lead to the same surface that is rendered when an object is rotated in 3-D space about Z- and X- axes followed by orthogonal projection. Moroever, when desired, the oblique distortion of the surface can be removed by a geometrical transformation of the final image. We will indicate how we can determine and exploit specific oblique projections with the property of mapping vertices of voxels exclusively on discrete coordinates in order to obtain very accurate rendering of surfaces.**

I. INTRODUCUCTION

Nowadays scientists can choose out of a range of different methods for visualizing 3-D data. According to the taxonomy of Elvins, [1] two main streams of algorithms can be distinguished, surface fitting algorithms and direct volume rendering algorithms.

Surface fitting algorithms detect surfaces in 3-D data and subsequently transform them in polygonal descriptions. The geometric primitives involved are usually triangles which can be used either to connect previously detected surface contours [2] or more directly, to describe surfaces within logical cubes [3]. In [4] an alternative approach is described as the cuberille model where surfaces are modeled with faces of cubes instead of triangles.

Direct volume rendering methods project surface elements directly into screen space without using geometric primitives as an intermediate representation. Surface elements can be projected either directly from the volume array into the image plane or in the reverse way by casting rays from an image plane into the volume array. An early direct volume rendering method that projects surface elements is described by Frieder et al. [5]. Their method traverses slices in back-to-front order whereby the image coordinates of voxels are calculated according to a 3-D rotation matrix and scaled to avoid artifact holes. More recently developed direct volume rendering methods that are based on orthogonal projections use so called splats as projection primitives [6], [7].

In this paper we describe a new direct volume rendering method based on oblique projections that can be used for back-to-front display of voxel based data. We prove that a surface obtained by rotating an object about Z- and X- axes in 3-D space, and then orthogonaly projected can also be obtained by an oblique projection followed by a 2-D transformation, essentially a 2-D scaling, which results in a speed advantage. Furthermore, we will indicate how we can determine and exploit specific oblique projections with the property of mapping vertices of voxels exclusively on discrete coordinates in order to obtain very accurate rendering of textures on objects.

II. OBLIQUE PROJECTIONS

An oblique projection of an object is obtained using projectors that are not perpendicular to the projecting plane [8]. We can describe an oblique projection as shown in figure 1, by the values α and f where f (the foreshortening factor) is the projection of the unit z-axis vector [001] on the projection plane (z=0) and α is the angle between this projection and the x-axis. Figure 1 also shows β the angle between the oblique projector (direction of projection) and the plane of projection, $\beta = \cot^{-1}(f)$. Representing points by row matrices and using homogeneous coordinates, the transformation matrix for producing any parallel projection

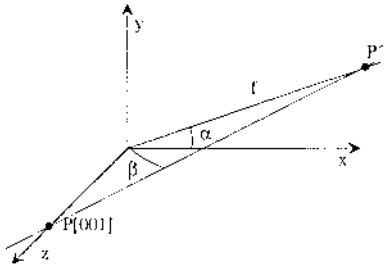


Fig. 1 - Oblique projection on projection plane z=0.

of 3D object on the projecting plane z=0 is defined as:

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -f\cos\alpha & -f\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

THEOREM 1

Let

1-X be a 3-D object

2-$[3DRPz]$ be the transformation matrix that rotates X an angle δ about the local z'-axis, followed by a rotation ϕ about the local x'-axis and subsequently followed by an orthogonal projection on the $z = 0$ plane Then, there exists an oblique projection $[POz]$ and a 2-D transformation $[2D]$ such that

$$[X][3DRPz] = [X][POz][2D]$$

end of theorem 1

For a proof of theorem 1 see appendix A.

The significance of theorem 1, for visualizing voxel-represented objects, is twofold. Provided the voxels are projected in the same order as if they were projected with the rotation matrix:

1. The same set of visible surface faces obtained after rotating an object about Z- and X- axes followed by orthogonal projection can be obtained by a specific oblique projection.
2. The same set of visible surface faces with the same geometrical proportions obtained after rotating an object about Z- and X- axes followed by orthogonal projection can be obtained by a specific oblique projection followed by a specific 2-D transformation.

III. THE DISCRETE COORDINATES PARADIGM

Important for the quality of a 3-D visualization method is the accuracy with which coordinates in 3-D space (object space) can be transformed in a discrete 2-D space (image space). In methods that use a 3-D rotation matrix we always encounter the problem that 3-D coordinates may be transformed into non-discrete coordinates, which will have to be rounded or truncated. In this section we will indicate how we can avoid the occurrence of non-discrete coordinates during a 3-D to 2-D transformation by using a set of specific oblique projections with the property of projecting vertices of voxels exclusively onto discrete coordinates. We will indicate how to choose particular values α and f to obtain this property by means of the following theorem.

THEOREM 2

Let S_1 be an axis aligned square of size R with its left bottom vertex positioned on point (0,0) (figure 2).

Let S_2 be a square obtained from S_1 using a translation vector (T_x, T_y).

Let C be an axis aligned cube of size R, with one of its vertices on the origin and the others with positive coordinates (figure 2).

Then an oblique projection defined by:

$$\alpha = \arctan\frac{T_y}{T_x} \quad (2)$$

and

$$f = \frac{T_x}{-R\cos\alpha} \quad (3)$$

projects the vertices of C onto the vertices of S_1 and S_2

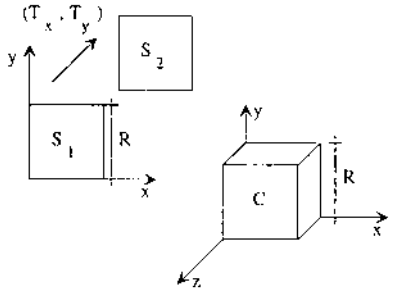


Fig. 2 - An oblique projection of a cube can be described as a translation of a square.

end of theorem 2.

For a proof of theorem 2 see appendix B.

With the help of theorem 2, we can now establish a set of oblique projections with the property of projecting voxel vertices exclusively on discrete coordinates. The procedure is simple. Draw an oblique projected voxel of size R on a grid using a translation vector (T_x, T_y), using integer values for R, T_x and T_y, calculate the oblique parameters α and f with equations 2 and 3. According to theorem 1, we can subsequently calculate the associated 3-D rotation angles with equations 17 and 18, as described in appendix A.

For example, from an oblique voxel defined by $T_x = T_y = -1$ and $R = 1$ we can calculate the oblique parameters α and f and therefore the associated 3-D rotation angles, which in this case will lead to an isometric projection. In contrast with this oblique projection an isometric projection does not exclusively map voxel vertices onto discrete coordinates.

IV. THE OBLIQUE VOXEL METHOD

The oblique voxel method traverses a 3-D volume slice by slice starting with the slice furthest away from the observer. Voxels in each slice are traversed in back-to-front order guided as in case the object was rotated by the associated 3-D rotation matrix. We can use different discrete projecting voxels as projection primitives as can be identified with theorem 2. In order to avoid artifact holes each voxel is scan converted on its bounding rectangular area as depicted in the example of figure 3.

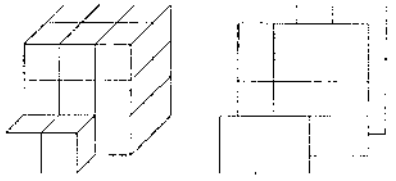


Fig. 3 - Back-to-front display of oblique projected voxels. Each voxel is scan-converted on its bounding rectangular area to avoid artifact holes.

Using one specific oblique transformation or projection primitive, 24 different projections can be obtained by combining both traversal and projecting order of voxels. Spe-

cifically, projection of slices can take place in 3 different ways as shown in figure 4.

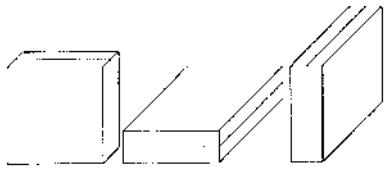


Fig. 4 - Three different ways of projecting oblique slices in back-to-front order.

Traversal of slices can take place either in first-to-last or last-to-first order and traversal of voxels within each slice can take place in 4 ways by alternating the start point of reading at one of the four corners of a slice.

The final step in the method consists of a 2-D transformation, a 2D warping post-process, of the image in order to remove the oblique distortion. We use the transformation matrix $[2D]$ of equation 4 with matrix parameters as can be calculated with equations 11 and 12 as described in appendix A.

V. COMPUTATIONAL COMPLEXITY OF OBLIQUE VOXEL METHOD

The computational complexity of any direct volume rendering method that projects voxels in back to front order without pre-processing of the volume data can be expressed as a time function $O(f(n^3))$ where n stands for the maximal dimension of the volume data and f is the time function expressing the computational effort to calculate the image coordinates of a voxel and to scan-convert its associated projection primitive on a display.

In the oblique voxel method, calculation of image coordinates is a relatively inexpensive procedure since no 3-D matrix is required. Image coordinates of voxels are determined while traversing the volume data by tracking two pointers yielding the projected coordinates of the currently accessed voxel. The bottle neck in the computational complexity of the method can be found when it is applied with oblique projections leading to large dimensions for the rectangular projection primitives (see theorem 2). However, significant optimisation for these critical oblique projections could be obtained when the display system supports either software or hardware for fast scan-conversion of rectangles.

VI. RESULTS

We tested the method's expected abilities of rendering surfaces with voxel precision by applying it on several objects. For example, we created a 64x64x64 cube covered with a texture of numbers (figure 5), on each of its faces.

A distance image was produced with the oblique voxel method using the following parameters $T_x = T_y = -1$ and $R = 1$, from which we can calculate the oblique parameters α and according to theorem 2. By means of equations (17) and (18) the associated 3-D rotation angles can be calculated, which in this case will lead to an isometric projection. The oblique distortion can be removed by 2-D transformation with a scaling factor and rotation angle as can be calculated with equations (12) and (17). Figure 6 shows the

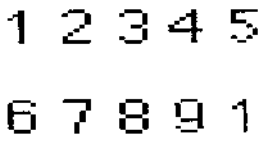


Fig. 5 - Image displaying numbers to be used as a texture.

oblique distance image processed with an image space shading operator [9].

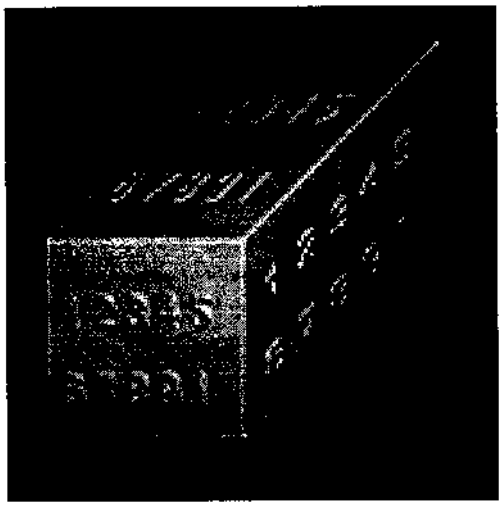


Fig. 6 - An oblique projection of a 64x64x64 cube with a texture of numbers

The oblique distortion was removed by a transformation using a bi-cubic interpolation of the oblique image leading to an isometric projection of the cube as displayed in figure 7.

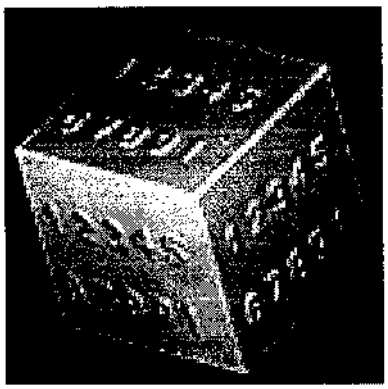


Fig. 7 - Resulting image after transforming the oblique projection into an isometric one by means of a 2-D transformation based on a bi-cubic interpolation.

Figure 8 shows the same isometric projection of the cube obtained after applying a direct volume rendering method that uses a 3-D rotation matrix and projects voxels as pixels. The resulting z-buffer image was processed with the same image space shading operator as in the case of the oblique

image displayed in figure 6. The significant loss in texture information, which is clearly visible at all three faces of the cube, can be explained by the fact that pixels are not very suitable projection primitives for the hexagonal shape of rotated and orthogonally projected voxels and by the fact that the image coordinates of the rotated voxels may have to be rounded.

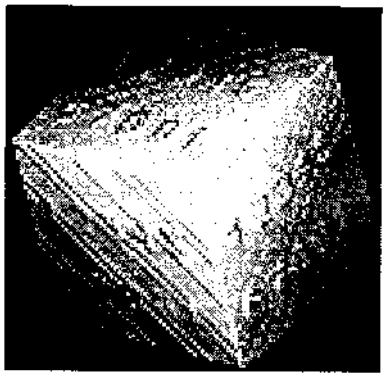


Fig. 8 - Isometric projection obtained with a method that uses a 3-D rotation matrix and projects voxels as pixels.

Figure 9 shows the same isometric projection of the cube obtained after applying a ray casting method that uses nearest neighbour interpolation. The ray caster is capable of preserving the texture information but some irregularities are visible, for example in the boundary of the numbers and the "false" edges on the originally smooth surface parts of the cube.

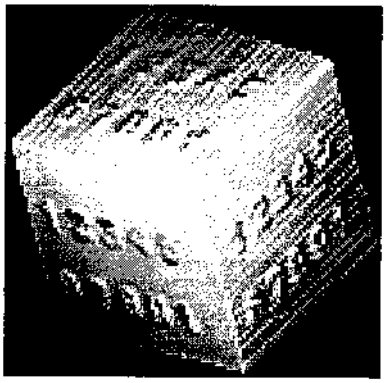


Fig. 9 - Isometric projection obtained with a ray caster that uses nearest neighbour interpolation.

A second test object consisted of a 256x256x129 volume containing data of a human thorax (obtained by Computarised Axial Tomography). A distance image was produced with the oblique voxel method using the following parameters, $T_x = T_y = -1$ and $R = 1$ which, as described above, is equivalent to an isometric projection. Figure 10 shows the oblique distance image after being processed with the image space shading operator.

The oblique distortion was removed by transformations based on a bi-cubic interpolation and a bi-linear interpolation of the original oblique image. The resulting image (figure 11) is far more sharper than the slightly blurred image resulting from transformation based on bi-linear interpolation (figure 12).

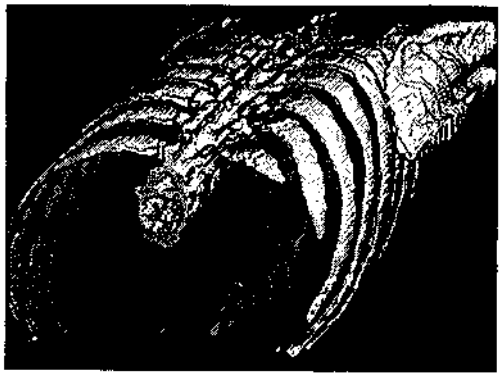


Fig. 10 - An oblique projection of a 256x256x129 volume containing CAT data of a human thorax

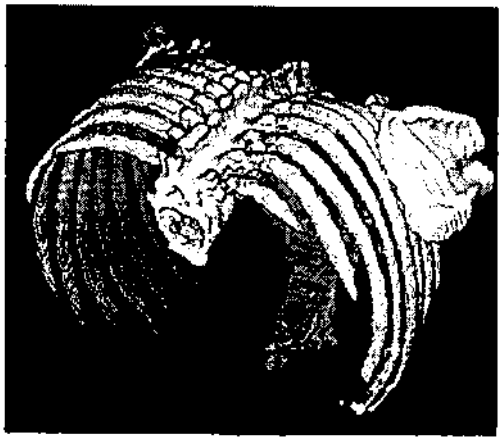


Fig. 11 - Image resulting from transforming the oblique projection into an isometric one using a 2-D transformation based on a bi-cubic interpolation.

However, in contrast with the transformation based on bi-linear interpolation, the transformation based on bi-cubic interpolation leads to some "false" dark spots or pixels at the edges of the ribs due to interpolation with the background colour of the image. In comparison with the image produced by the direct volume rendering method that uses a 3-D rotation matrix and projects voxels as pixels (figure 13) some differences with the oblique voxel method are visible especially in the texture of the shoulder blade and spine.

The image produced by the ray casting method that uses nearest neighbour interpolation (figure 14) is comparable in quality to the images produced by the oblique voxel method.

All the methods described in this section were implemented in the C language on a Silicon Graphics, Iris Indigo Elan 4000 with 80MByte of memory. We have measured the average time of 10 trials for each method in order to produce a distance image of the thorax after the entire volume was read into memory. The oblique voxel method used 1.2 seconds, the direct volume rendering method which projects voxels as pixels used 3,3 seconds and finally the ray caster that uses nearest neighbour interpolation used 166.5 seconds. Additionally, we measured the time needed by the oblique voxel method in order to generate a distance image of the same volume but now with oblique parameters $T_x = T_y = -1$ and $R = 2$ which can be associated with a rotation of 45 degrees about both z and x axis. As expected, due to the larger dimension for the rectangular projection primitives, we

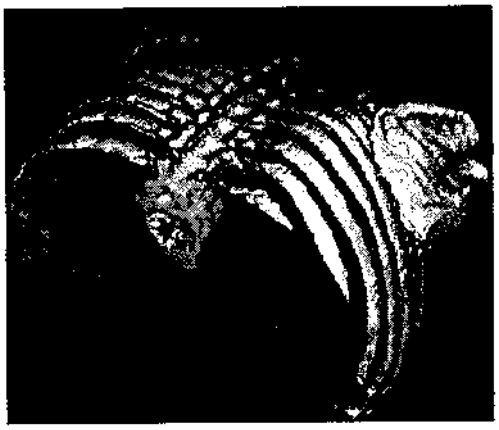


Fig. 12 - Resulting image after transforming the oblique projection into an isometric one using a 2-D transformation based on a bi-linear interpolation.

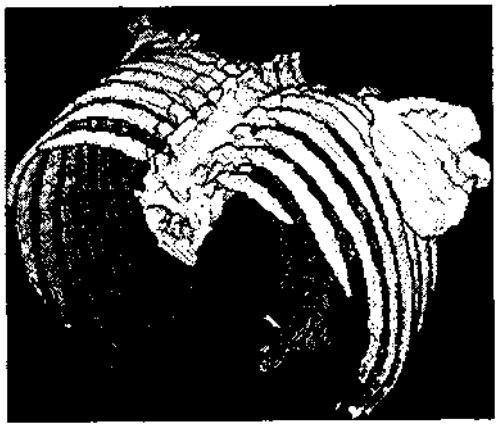


Fig. 13 - Isometric projection obtained with a method that uses a 3-D rotation matrix and projects voxels as pixels.

measured a slightly larger time of 1.3 seconds.

In relation to the results we refer briefly to the discussion in [10] describing the specific advantages and disadvantages of oblique projections in comparison with orthogonal projections. Oblique projections may be particularly suitable for objects with much detail or irregular shapes on one principal face since they have the property of displaying the exact shape of one face of an object. For example, see the front face of the oblique projected cube in figure 6 which displays the numbers with preservation of angles and lengths. A disadvantage of oblique projections is the typical oblique distortion which may lead to an unrealistic perception of the object, however our 2D warping post-process eliminates this problem. For example the oblique projected cube in figure 6 appears to be a parallelepiped object instead of a cube as is perceived correctly in the orthogonal projection in figure 7.

VII. Discussion

In addition to the current spectrum of volume rendering methods, oblique projections can be helpful to render voxel data with voxel precision. We gave the mathematical foundations for a direct volume rendering method allows a 3-D object to be viewed under different viewing positions without using a 3-D rotation matrix. We have indicated how

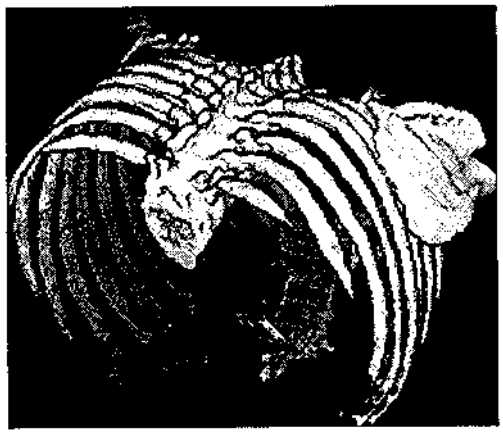


Fig. 14 - Isometric projection obtained with a ray caster that uses nearest-neighbour interpolationray caster that uses nearest-neighbour interpolation.

we can use specific oblique projections, that project vertices of voxels exclusively onto discrete coordinates, in order to avoid the introduction of rounding errors of coordinates while the surface is generated. As a consequence, surfaces are rendered with hardly any loss in texture information. A unique property of the method is that it produces two images displaying the same set of points of an object, with both an oblique and an orthogonal projection with the advantage that each projection has its specific characteristics.

We conclude that the oblique voxel method is capable of very fast and accurate volume rendering with the constraint that it allows an object only to be viewed under a limited set of different rotation angles corresponding to specific oblique projections. As future work we intent to investigate the possibility of a more general projection method that allows the user to view the volume from any angle, allowing for example, quick rendering of images or real time animation sequences.

Acknowledgements

This work was partially funded by the Portuguese Agency for Cientific Research and Technology (JNICT) with a scholarship BD/1733/91-IA.

References

[1] T. Todd Elvins, "A survey of algorithms for volume visualization", *Computer Graphics*, vol. 26, no. 3, pp. 194–201, Aug. 1992.

[2] H. Fuchs, Z. M. Kedem, and S. P. Uselton, "Optimal surface reconstruction from planar contours", *Communications of the ACM*, vol. 20, no. 10, pp. 693–702, Oct. 1977.

[3] William E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm", *ACM Computer Graphics*, vol. 21, no. 4, pp. 163–169, July 1987.

[4] Gabor T. Herman and Hsun Kao Liu, "Three-dimensional display of human organs from computed tomograms", *Computer Graphics and Image Processing*, vol. 9, pp. 1–21, 1979.

[5] Gideon Frieder, Dan Gordon, and R. Anthony Reynolds, "Back-to-front display of voxel-based objects", *IEEE Computer Graphics & Applications*, pp. 52–60, Feb. 1985.

[6] J. Wilhelms and A.V. Gelder, "A coherent projection approach for direct volume rendering", *Computer Graphics*, vol. 25, no. 4, pp. 275–284, 1991.

[7] Lee Westover, "Footprint evaluation for volume rendering", *ACM Computer Graphics*, vol. 24, no. 4, pp. 367–376, Aug. 1990.

[8] Ludwig Adams, Werner Krybus, Dietrich Meyer-Ebrecht, Rainer Rueger, Joachim M. Gilsbach, Ralph Moesges, and George Schloendorff, "Computer-assisted surgery", *IEEE Computer Graphics & Applications*, pp. 43–51, May 1990.

[9] Dan Gordon and R. Anthony Reynolds, "Image space shading of 3-dimensional objects", *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 361–376, 1985.

[10] I. Carlbom and J. Paciorek, "Planar geometric projections and viewing transformations", *Computing Surveys*, vol. 10, no. 4, pp. 465–502, 1978.

APPENDIX A

Proof of theorem 1

Consider X to be oblique projected with $[POz]$ followed by a rotation about the z axis with an angle β, subsequently followed by a scaling along the y axis with a scaling factor S_y and finally followed by a translation along the x axis and y axis with factors T_x and T_y, respectively. The combined transformation is then described as:

$$[POz2D] = [POz]\,[2D] \tag{4}$$

where

$$[POz2D] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -f\cos\alpha & -f\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} \cos\beta & S_y\sin\beta & 0 & 0 \\ -\sin\beta & S_y\cos\beta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ T_x & T_y & 0 & 1 \end{bmatrix} \tag{5}$$

(Note that although $[2D]$ describes a 2-D transformation it is written in the form of a 3-D transformation matrix) for which we calculate

$$[POz2D] = \begin{bmatrix} \cos\beta & S_y\sin\beta & 0 & 0 \\ -\sin\beta & S_y\cos\beta & 0 & 0 \\ A & S_y.B & 0 & 0 \\ T_x & T_y & 0 & 1 \end{bmatrix} \tag{6}$$

where

$$A = -f\cos\alpha\cos\beta + f\sin\alpha\sin\beta$$

and

$$B = -f\cos\alpha\sin\beta - f\sin\alpha\cos\beta$$

Now consider X to be rotated about a local axis system x'y'z' with its origin on C. Where $C = (C_x, C_y, C_z)$ is the centroid of X. A rotation of X about the local z' axis with angle δ followed by a rotation about the local x' axis with angle ϕ can be performed by translating X to make C coincident to that of the origin of the global axis system, susequently followed by the required rotations and finally by translating X back with C to its original position. After rotation, X will then be projected orthogonally onto the $z = 0$ plane. The combined transformation is then described as:

$$[3DRPz] = [3DR]\,[Pz] \tag{7}$$

where

$$[3DR] = [TR]\,[Rz]\,[Rx]\,[TR]^{-1}$$

specifically

$$[3DR] = \begin{bmatrix} \cos\delta & \cos\phi\sin\delta & \sin\phi\sin\delta & 0 \\ -\sin\delta & \cos\phi\cos\delta & \sin\phi\cos\delta & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ C & D & E & 1 \end{bmatrix} \tag{8}$$

where

$$C = -C_x\cos\delta + C_y\sin\delta + C_x$$

$$D = \cos\phi(-C_x\sin\delta - C_y\cos\delta) + C_z\sin\phi + C_y$$

and

$$E = \sin\phi(-C_y\cos\delta - C_x\sin\delta) - C_z\cos\phi + C_z$$

$$[3DRPz] = \begin{bmatrix} \cos\delta & \cos\phi\sin\delta & \sin\phi\sin\delta & 0 \\ -\sin\delta & \cos\phi\cos\delta & \sin\phi\cos\delta & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ C & D & E & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

from which we calculate

$$[3DRPz] = \begin{bmatrix} \cos\delta & \cos\phi\sin\delta & 0 & 0 \\ -\sin\delta & \cos\phi\cos\delta & 0 & 0 \\ 0 & -\sin\phi & 0 & 0 \\ C & D & 0 & 1 \end{bmatrix} \tag{10}$$

The equation $[POz2D] = [3DRPz]$ will hold a solution if:

$$\beta = \delta \tag{11}$$

$$S_y = \cos\phi \tag{12}$$

$$T_x = -C_x\cos\delta + C_y\sin\delta + C_x \tag{13}$$

$$T_y = \cos\phi(-C_x\sin\delta - C_y\cos\delta) + C_z\sin\phi + C_y \tag{14}$$

$$-f\cos\alpha\cos\beta + f\sin\alpha\sin\beta = 0 \tag{15}$$

$$S_y(-f\cos\alpha\sin\beta - f\sin\alpha\cos\beta) = -\sin\phi \quad (16)$$

Applying a fixed α and f we can derive β, δ and ϕ. Since $f \neq 0$ equation 15 yields

$$\cos\alpha\cos\beta = \sin\alpha\sin\beta$$

Using the identities $\frac{\sin\alpha}{\cos\alpha} = \tan\alpha$ and $\frac{\sin\beta}{\cos\beta} = \tan\beta$ yields

$$\tan\alpha\tan\beta = 1$$

From which we derive

$$\delta = \beta = \frac{\pi}{2} - \alpha \quad (17)$$

From equation 16 we can derive ϕ
Applying equation 12 in 16 yields

$$\cos\phi(-f\cos\alpha\sin\beta - f\sin\alpha\cos\beta) = -\sin\phi$$

Using the identity $\frac{\sin\phi}{\cos\phi} = \tan\phi$ yields

$$-f\cos\alpha\sin\beta - f\sin\alpha\cos\beta = -\tan\phi$$

From which we derive

$$\phi = \arctan(f\cos\alpha\sin\beta + f\sin\alpha\cos\beta)$$

Using the identities $\sin\beta = \cos\alpha$ and $\cos\beta = \sin\alpha$ conform equation 17 yields

$$\phi = \arctan(f(\cos^2\alpha + \sin^2\beta))$$

Using the identity $\cos^2\alpha + \sin^2\beta = 1$ yields

$$\phi = \arctan(f) \quad (18)$$

Applying a fixed δ and ϕ we derive α and f

$$\alpha = \arctan(\frac{1}{\tan\delta}) \quad (19)$$

$$f = \tan\phi \quad (20)$$

APPENDIX B

Proof of theorem 2
See figure 2 for symbols.
Let S_1 be defined by the set $\{(0,0)(0,R)(R,R)(R,0)\}$ and S_2 defined by the set $\{(T_x,T_y)(T_x,R+T_y)(R+T_x,R+T_y)(R+T_x,T_y)\}$
Let C be defined by the coordinates formed by the union of C_1 and C_2. Where $C_1 = \{(0,0,0)(0,R,0)(R,R,0)(R,0,0)\}$ and $C_2 = \{(0,0,R)(0,R,R)(R,R,R)(R,0,R)\}$.
Given $\alpha = \arctan(\frac{T_y}{T_x})$ and $f = \frac{T_x}{-R\cos\alpha}$
Applying equation 1, the coordinates of any oblique projected point can be expressed as:

$$x_p = x - zf\cos\alpha \quad (21)$$

$$y_p = y - zf\sin\alpha \quad (22)$$

Applying equation (21 and equation (22) with $z = 0$ yields

$$x_p = x \quad (23)$$

$$y_p = y \quad (24)$$

Equations (21) and (22) will transform C_1 in S_1
Applying equation (21) with $z = R$, $\alpha = \arctan(\frac{T_y}{T_x})$ and $f = \frac{T_x}{-R\cos\alpha}$ will give

$$x_p = x - R(\frac{T_x}{-R\cos\alpha})\cos\alpha$$

which yields

$$x_p = x + T_x \quad (25)$$

Applying equation (22) with $z = R$ and $f = \frac{T_x}{-R\cos\alpha}$ will give

$$y_p = y - R(\frac{T_x}{-R\cos\alpha})\sin\alpha$$

which yields

$$y_p = y + T_x\frac{\sin\alpha}{\cos\alpha}$$

Using the identity $\frac{\sin\alpha}{\cos\alpha} = \tan\alpha$ yields $y_p = y + T_x\tan\alpha$
Applied with $\alpha = \arctan(\frac{T_y}{T_x})$ will give $y_p = y + T_x\frac{T_y}{T_X}$ which yields

$$y_p = y + Ty \quad (26)$$

Equation (25) and (26) will transform C_2 into S_2

Software Gráfico Orientado por Objectos para Aplicações Multimédia

Silvério Neves, Luís Almeida

Resumo- **Neste artigo descrevemos o desenvolvimento de um ambiente integrado de alto nível para PCs com suporte de programação gráfica.**

O desenvolvimento foi efectuado em ambiente Windows, em linguagem C++ e com ferramentas multimédia. O resultado deste desenvolvimento foi a criação de uma classe de objectos gráficos com características dinâmicas e estáticas e um editor gráfico com suporte de animação gráfica ("MoveIt v1.0").

Abstract- **This paper describes the development of a high-level integrated environment for PCs with graphics programming support.**

The development was carried out on the Windows™ environment, with C++ language and multimedia tools. The result of this development was the creation of a graphic objects class with static and dynamic features and a graphic editor with graphic animation support ("MoveIt v1.0").

I. INTRODUÇÃO*

Nos dias de hoje, falarmos de computadores, telecomunicações ou formação faz surgir uma palavra mágica: *multimédia*. O seu significado, sendo vasto, insere-se essêncialmente na conjugação de texto, som e imagem num mesmo meio de comunicação. O crescimento exponencial do uso da multimédia no nosso dia-a-dia deve-se ao aparecimento de ferramentas (melhores processadores, placas de som, placas de vídeo, redes de comunicações de alta capacidade,...) com um progressivo baixo custo para o cidadão comum.

Inserido nesta filosofia, o projecto que desenvolvemos tinha como finalidade a criação de um ambiente integrado de alto nível para PCs com suporte de programação gráfica. O seu desenvolvimento dividiu-se essêncialmente em três partes :

A primeira centrava-se na elaboração de um classe de objectos gráficos com características estáticas (posição, tamanho, cor, plano,...) e dinâmicas (direcção de movimento, distância de movimento, tipo de movimento, velocidade de movimento,...).

Na segunda parte criou-se um editor gráfico com suporte de animação gráfica para a referida classe de objectos gráficos.

Para a terceira parte reservou-se a inserção de características multimédia (som, vídeo,...).

* Trabalho realizado no âmbito da disciplina de Projecto.

II. FERRAMENTAS UTILIZADAS

Ao nível do sistema operativo optamos pelo ambiente Windows por ser o mais utilizado nos PCs e com mais probabilidades de o continuar a ser no futuro.

Software utilizado :

Win32s - a arquitectura de 32 bits tem muitas vantagens em relação à de 16 bits: é usada nos novos processadores, os novos sistemas operativos são de raíz de 32 bits, a gestão de memória torna-se muito mais flexível, etc.

Linguagem C++ - uma linguagem extremante flexível e apropriada para o ambiente Windows cujo modelo se baseia numa estructura orientada por objectos [1] [2].

Borland C++ - um compilador que possuiu uma vasta livraria para o ambiente Windows (Object Windows Libray 2.0) [3] permitindo ainda compilar para Win32s.

Video for Windows Development Kit 1.1 - fornece as ferramentas necessárias para manipular ficheiros vídeo (*.AVI - Audio Video Inteleaved) [8].

Para fazer uso deste software trabalhamos num PC com processador 486DX4, 8Mbytes de RAM e placa de som Sound Blaster 16.

III. IMPLEMENTAÇÃO

A. Classe de Objectos gráficos

A classe de objectos gráficos por nós implementada assenta numa classe principal Tobject. Todos os objectos particulares são derivados desta classe como mostra a figura 1.

A class TObject possuiu as linhas de orientação para as classes derivadas. As suas funções membro controlam as operações do rato, o desenho do objecto, a leitura e escrita

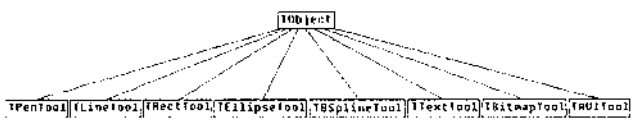


Fig. 1 - Classe de Objectos gráficos

num ficheiro e a manutenção de uma tabela de movimentos e sons. Contendo informação importante relativa a cada objecto como :

- tipo do objecto (elipse, rectangulo, etc...),
- cor,
- espessura de linha,
- tamanho do objecto,
- plano,
- tabela de movimento (MotionTable array),
- etc.

Em cada classe derivada definem-se propriedades própias dos diferentes objectos gráficos, reescrevendo se necessário funções da classe Tobject.

A tabela de movimento é um array de classes ObjMotion ordenado pela varivel de tempo de cada elemento. A classe ObjMotion possuiu as seguintes variáveis : *Time*, *Position*, *Motion*, *RefPoint* e *Wave*.

Definindo a que tempo *Time* o objecto se situa na posição *Position*, o tipo de movimento *Motion* que efectua para chegar à posição *Position* e o som que deve ser tocado no tempo *Time*.

B. Estructura de ligação Tstate

Para fornecer uma eficaz ligação entre a nossa aplicação e a classe de Objectos gráficos implementou-se a estructura Tstate. Esta estructura possuiu as seguintes variáveis :

- tipo de objecto seleccionado,
- variável de cada tipo de Objecto gráfico,
- plano disponivel para desenhar,
- espessura da caneta seleccionada,
- cor seleccionada,
- tipo de letra seleccionada,
- posição do rato,
- tempo actual da cena,
- se algum objecto está seleccionado,
- tipo de movimento seleccionado,
- palete da aplicação,

C. Duplo Buffer

Essencialmente há três técnicas de animação, a aplicação lógica XOR, a troca de página video e o duplo buffer [9].

A primeira consite na escrita directa de um sprite (conjunto de bits que descrevem um objecto) na memória video através da operação lógica XOR. Esta técnica tem um grande inconveniente, só funciona com fundos sólidos.

A troca de página video consiste em dividir a memória video em diferentes páginas (caso o modo gráfico o permita) e activar só uma delas. Numa página não activa escreve-se a cena para o instante de tempo que se segue, actualizando-se o ecrã pela activação dessa mesma página.

O buffer duplo é conceptualmente similar à anterior, sómente que se usa um buffer em vez de uma página video. Esta técnica tem um senão, é a cópia da memória em massa para o ecrã.

A nossa escolha recaiu no buffer duplo porque as duas primeiras técnicas lidam directamente com o hardware do adaptador de video. O ambiente Windows é independente do hardware (uso de drivers), tornando-se inconcebível a aplicação lógica XOR ou a troca de página video como técnica de animação.

D. Palete

A gestão da palete no ambiente Windows é efectuada através de uma lookup table indexada [9]. Isto é, temos uma palete de hardware e uma palete lógica. A palete de hardware possuiu o numero de cores que a memória do adaptador video permitir, mas a palete lógica possuiu um conjunto de cores indexadas com numero superior ou igual à palette de hardware. As cores disponiveis para o ecrã são as presentes na palete lógica. Como ilustra a figura 2, esta gestão de cores pode tornar-se morosa.

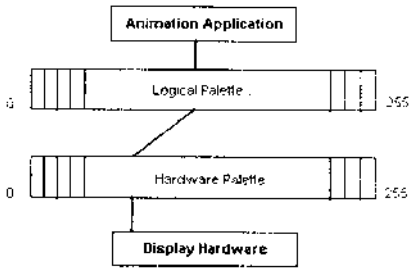


Fig. 2 - Gestão da palete

Para uma melhor performance da animação e uma uniformização de cores durante a mesma é conveniente criar uma palete identidade. Este procedimento em termos práticos "cola" a palete lógica à palete de hardware.

E. Timer

No coração do velho IBM PC, e ainda na maior parte dos PCs de hoje, há um timer de hardware que origina uma interrupção em cada 55 milisegundos. Significando pulsos de relógio 18,21 vezes por segundo [10].

Para o ambiente Windows, este timer é uma limitação para aplicações multimédia, visto que, as mensagens do Windows não são atendidas da fila de espera mais rápido do que 18,21 vezes por segundo.

Na nossa aplicação usamos um timer de 10 vezes por segundo. Esta opção aliada a uma boa estrutura da nossa aplicação mostrou-se boa, contando que durante a animação praticamente a única mensagem necessária é a do timer.

IV. MOVEIT V1.0

Como resultado do nosso desenvolvimento surgiu a aplicação multimédia "MoveIt v1.0". É uma aplicação MDI (Multiple Document Interface), em que cada janela MDI Child representa uma cena de animação. A janela principal possuiu vários elementos de interacção com o utilizador: menu, control bar, status bar, tools windows, animate window e time bar como poderemos observar na figura 3.

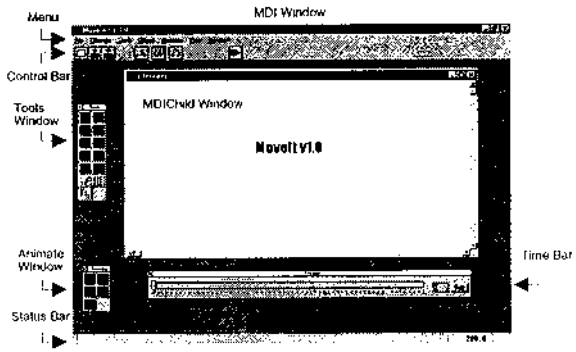


Fig. 3 - Janela principal da aplicação MoveIt v1.0

A. Tools Window

A Tools Window permite-nos lidar com os objectos gráficos por nós criados. A figura 4 mostra-nos esta janela.

1 - Modo de selecção.
2 - Desenho livre.
3 - Linha.
4 - Rectangulo.
5 - Elipse.
6 - Curva B-Spline.
7 - Texto.
8 - Bitmap.
9 - AVI.
10 - Côr de fundo.
11 - Côr da linha.
12 - Espessura da linha.
13 - Tipo de letra.

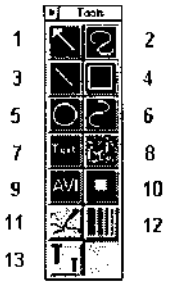


Fig. 4 - Tools Window

B. Animate Window

Para atribuição do tipo de movimento ao objecto gráfico usamos a Animate Window como mostra a figura 5.

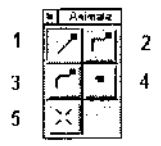


Fig. 5 - Animate Window

1 - Movimento directo.
2 - Movimento com quebra.
3 - Movimento curvo.
4 - Parado.
5 - Desaparecer.

C. Control Bar

O Control Bar como mostra a figura 6 possuiu 7 botões com as seguintes funções :

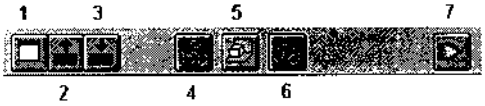


Fig. 6 - Control Bar

1 - Criar um novo ficheiro MoveIt.
2 - Abrir um ficheiro MoveIt.
3 - Gravar a cena corrente.
4 - Define-se o tempo da cena.
5 - Permite-nos escolher o fundo da cena. Pode ser uma côr ou um bitmap.
6 - Selecciona um ficheiro de som Wave para tocar em associação com o objecto seleccionado.
7 - Este importante comando executa a animação que está criada na cena corrente.

D. Menu

Todas os comandos do menu existem em icons na Tools Window, Animate Window e Control Bar menos os do grupo Object (figura 7).

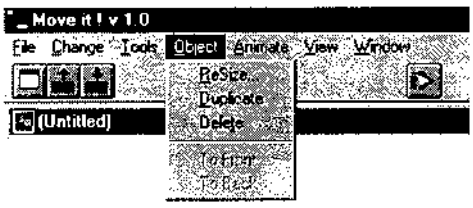


Fig. 7 - Object pull-down menu do menu principal

Este grupo do menu possibilita-nos redimensionar o tamanho do objecto, duplicá-lo ou apagá-lo. Estas acções sobre os objectos só têm efeito quando seleccionados.

E. Time Bar

A figura 8 mostra-nos o Time Bar.

Fig. 8 - Time Bar

O Time Bar é um elemento fundamental de interacção para a concepção da cena de animação. Ao deslocarmos o cursor ao longo do Time Bar estamos a situar a cena temporalmente, permitindo inserir objectos gráficos no tempo desejado como visualizar a posição dos objectos ao longo do tempo. Esta funcionalidade do Time Bar dá-nos a possibilidades de conceber uma animação com precisão em "Draw Time" de um segundo.

V. SHOWIT V1.0

Desenvolveu-se outra aplicação denominada ShowIt v1.0. Embora no MoveIt possamos correr a animação da cena da janela MDI Child activa, esta aplicação permite-nos executá-la separadamente.

A figura 9 mostra-nos a interface gráfica desta aplicação.

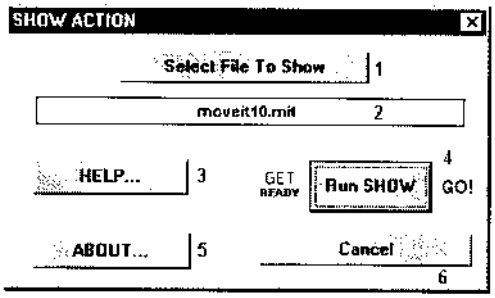


Fig. 9 - Diálogo inicial com o utilizador da aplicação ShowIt v1.0

1- Seleccionar a animação pretendida.
2- Mostra o ficheiro seleccionado.
3- Ajuda da aplicação.
4- Corre a animação.
5- Janela About.
6- Cancelar a selecção.

Durante a animação com a ajuda de algumas teclas podemos gerar algumas acções. A tabela 1 mostra as teclas especiais e as suas respectivas acções.

Tabela 1

Tecla	Acção
P	Pausa na animação
G	Eliminação da acção Pausa
B	Ir para o inicio da animação
E	Ir para o fim da animação
N	Seleccionar nova animação
Esc	Sair da aplicação

VI. FUTUROS MELHORAMENTOS

A aplicação MoveIt pode ser melhorada num futuro próximo em alguns aspectos :

- Implementação de botões interactivos.
- Elaboração de um ficheiro projecto, que possa englobar várias cenas de animação.
- Novos tipos de movimento.
- Sistema de "help on-line".
- Impressão.
- Introdução de objectos gráficos tri-dimensionais.

Numa perspectiva mais vasta, podemos inserir a classe de objectos gráficos numa Dynamic Link Library para uso público.

VII. CONCLUSÕES

No fim do desenvolvimento deste projecto ficamos ainda mais cientes das potencialidades que a multimédia nos oferece. Especialmente no ambiente Windows, poderemos afirmar que com uma linguagem tão versátil como o C++ e com as ferramentas de software e hardware que estão ao nosso dispôr nos dias de hoje, podemos desenvolver um excelente ambiente multimédia sobre ele.

Em relação à aplicação desenvolvida, "MoveIt v1.0", sendo um bom meio de efectuar apresentações a sua área de aplicação torna-se bastante vasta. A implementação dos botões interactivos referidos nos futuros melhoramentos alargam ainda mais essa área permitindo, por exemplo, situar-se na área da educação servindo de editor de tutoriais e de enciclopédias interactivas.

VIII. AGRADECIMENTOS

Agradecemos o Prof. Dr. V. Sklyarov a orientação do projecto.

REFERÊNCIAS

[1] Borland International, "Borland C++ for Windows - Programmer's Guide", 1993

[2] Borland International, "Borland Object Windows for C++ - Programmer's Guide", 1993

[3] Borland International, "Borland Object Windows for C++ - Reference Guide", 1993

[4] Anthony Porter, "C++ Programming for Windows", Osborne, McGraw-Hill, 1993

[5] Namir Clement Shammas, "What every Borland C++ 4 programmer should know", Sams Publishing, 1994

[6] Ori Gurawich and Nathan Gurewich, "Borland C++ Multimedia Programming", SYBEX, 1994

[7] Michael J. Young, "Windows Animation Programming with C++", AP Professional, 1994

[8] Paul DiLascia, "Exploit Video for Windows 1.1", Microsoft Systems Journal, Janeiro 1995

[9] James Finnegan, "Implementing Games for Windows", Microsoft Systems Journal, Janeiro 1995

[10] Rick Grehan, "The Software Stopwatch", BYTE - The Magazine of Technology Integration, Abril 1995

Palmtop Based System for Vocational Integration of Intellectually Disabled People: Preliminary Definition

Nelson Pacheco da Rocha[1], Bernardo Cunha[1], Giulio Lancioni[2], Mark O'Reilly[3], Anthony Montgomery[3], Philip Seedhouse[4], Fred Furniss[5], Pedro Morato[6]

[1]DETUA/INESC, [2]University of Leiden, [3]University College Dublin, [4]Arden Computing, [5]University of Leicester, [6]Faculdade de Motricidade Humana

***Resumo*- O projecto VICAID (Vocational Integration through Computer Assistance for Intellectually Disabled People) insere-se no âmbito do programa TIDE (Technology Initiative for Disabled and Elderly People) e tem como objectivos desenvolver e avaliar um sistema baseado num palmtop para apoio a pessoas com deficiências intelectuais profundas na realização de tarefas complexas em locais de trabalho integrados. O desenvolvimento do sistema será baseado num modelo de referência, o qual deverá considerar o inter-relacionamento entre factores humanos e limitações e possibilidades técnicas. Esta comunicação apresenta e analisa um conjunto de requisitos que está a ser utilizado para a definição do modelo de referência.**

***Abstract*- VICAID (Vocational Integration through Computer Assistance for Intellectually Disabled People) is a project within the TIDE programme (Technology Initiative for Disabled and Elderly People) that aims to develop and evaluate a palmtop based system to support people with severe intellectual disabilities to perform complex work routines in integrated work settings. The development of the system will be based in a reference model, which should consider the interrelationships between human factors and technical constraints and possibilities. This paper presents and analyses the set of requirements that is being used for the definition of the reference model.**

I. INTRODUCTION

Severe learning disability is a common form of life-long disability frequently associated with long-term unemployment. In recent years, employment within the competitive job market has become a meaningful option with severe learning disabilities [1]. Supported employment has offered an effective approach to assist persons with disabilities to secure paying jobs and can be considered as new opportunities for a large and disadvantaged section of the community. In this supported employment model, the individual with a severe disability is first assisted to find a job in the competitive market and then accompanied to the job by a work supervisor who provides direct individualised on-the-job training and support for the disabled worker.

The limited application of the supported employment model for persons with disabilities can be directly linked to two related phenomena: the first of these is the high initial costs of one to one work supervisor involvement while the second is the length of time for which work supervisor input is required, not only to initially teach the disabled person work routines, but also to maintain an acceptable level of performance over time.

One approach to the problem of maintaining consistent task performance has been to recruit the disabled person's fellow-workers to provide support to the disabled person. However, such interventions require greater-than-normal input from co-workers, focus mainly on limited target behaviours rather than on complex work routines, and are only effective if followed by intensive input from the work supervisor.

An alternative approach is the use of computer-aided programmes to facilitate occupational engagement in individuals with severe or profound disabilities [2]. Despite the effectiveness of these programmes one major drawback has been envisaged [3]: the lack of portability of complex equipment. However, the recent developments in consumer electronics have included the widespread availability of palmtop computer at relatively low prices. The VICAID (Vocational Integration through Computer Assistance for Intellectually Disabled People) is a project within the TIDE programme (Technology Initiative for Disabled and Elderly People) that aims to develop and evaluate a comprehensive system based around the use of palmtop microcomputers as prosthetic, teaching and support aids. This system will be used to support people with severe intellectual disabilities to perform complex work routines (tasks) in integrated work settings by providing a sequence of instructions as response to users keyed entries.

II. TECHNICAL APPROACH

The challenge of designing a system that should be at the same time light and small to be easily carried around,

simple enough to be operated by any impaired or non-impaired user, sufficiently robust to endure hard treatment by people with sensory or motor problems, easily expandable to accommodate different interaction devices, self-powered and autonomous for long periods of time, able to communicate or tele-communicate with other systems, powerful in terms of computation and with enough resources to support intelligent software capable of provide an adaptive prompting system is by no means a simple task that can be achieved by a simple engineering error and trial approach. A *Reference model* resulting from the merging and intersection of different requirements is a fundamental basic structure needed to perceive the best possible solution out of the ideal one as well as to design a feasible prototype in a pre-defined scheduled time. Therefore, the first aim of the engineering component of the project is to provide a common *Reference model* gathering all human factors and establishing their interrelationships with the technical constraints and possibilities. In order to do this we have structured our approach to the definition of that *Reference model* in the following three steps:

- Try to establish the major groups of requirements gathered according to their nature. The way this division is established must also imply that each of these groups of requirements may be examined and discussed separately from each other. By doing this we can envisage the different global solutions that could solve the problem when seen simply from that particular group of requirements.
- Perform a first independent analysis of each group of requirements, providing as a result the set of hardware and software prerequisites needed to answer those needs. This phase provides a first in-depth perception of the technical constraints and possibilities resulting from each requirement, its feasibility, and the level of its importance by comparing its practical need with the ease of implementation with current state of the art technology.
- Determine a *Reference model* by intercepting the different groups of requirements and their resulting hardware/software prerequisites. The interception volume will provide the guidelines for the best possible solution, eventually trimming the general ideal solution to a more simple but consistent and feasible one.

III. MAJOR GROUPS OF REQUIREMENTS

Following the approach just described, we have started by trying to identify those basic groups of requirements according to the specifications given above: they should be self-contained as much as possible, gather requirements that can be associated under a same common name and be as much orthogonal with each other as possible. This means that each group can be analysed independently from each other but that, in the end, at the hardware and software support layer, there will be a common intersection volume able to supply an answer to most of each group's needs. As a result of this top-down approach, three major groups of requirements have been identified: *General requirements*, *User Interface requirements* and *Programmability requirements*.

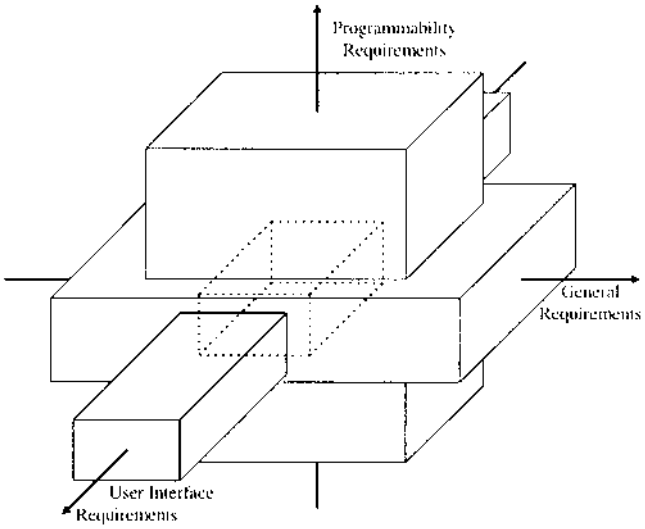


Fig. 1 - The *Reference model* definition approach.

The above described approach is depicted in Fig. 1, where each one of the requirement groups are identified as a volume centred on the origin of a three axis system. Each group, on the other hand, will grow according to one of the axis directions. Some requirements will stretch the corresponding volume away from the centre therefore reducing the intersection volume that will define the set of technical constraints of the project. By evaluating the importance and precedence of each requirement, and removing or redefining some of them, this intersection volume can be trimmed to the best compromise possible. A short description of these groups, and the current state of their structural analysis is further presented.

A. General Requirements

This first group of requirements results from previous knowledge of the problem, from the proposition of the project itself and, in general terms, from the common sense collected from the analysis of the typical constraints imposed by the virtual model of a mental impaired user working in an unidentified indoor environment. It was already clear from the project proposition that the equipment should be small and light to be easily carried around, easy to operate, self powered and robust. From the software point of view, the system should also have the capability to be adapted to new working situations, that is, it must be re-programmable and flexible enough in terms of processing power and architecture specifications to support software with different levels of complexity. The following set of requirements have therefore been systematised under this *General requirements* group,

followed by a discussion of the possible technical solutions to answer those needs:

- *Adaptability*; the system should be designed for people whose only severe impairment is cognitive. Nevertheless, it should also take into account that a substantial minority of these persons also have moderate to medium hearing losses, long or short sight that may not be properly corrected by glasses, and some problems with fine motor co-ordination. Provision for ease of adaptation to those particular cases must therefore be taken into consideration.
- *Small size (volume)*; this was a necessity from the start of the project, and the use of palmtop or alike computers has always been specified as a target objective. This requirement will naturally impose clear limitations to the integration of new, specific designed hardware, due to physical limitation of space, and will require a redesign of the traditional palmtop architecture.
- *Portability*; the equipment must be taken by the user as he/she moves around a non confined area. This also implies small size, but, most of all, it means that the equipment must not be heavy. This, again, imposes limitations in terms of inclusion of new features. We have also concluded that careful redesign of the equipment may compensate a small increase in weight by means of a more efficient way of handling it.
- *Autonomy*; the equipment must be fully operational at all times, that is, its power should not be turned on/off while moving around at the work site. Hence, solutions based on permanent attachment of the equipment to power outlets may not be considered. Ideally, the equipment should also be able to work uninterruptedly for at least eight hours (a full working day). Current state of the art palmtops support continuos work for over 60 hours. This will give us an overhead in terms of available power to integrate well designed, low power, devices.
- *Robustness*; careless treatment can most probably be expected by people with cognitive impairment. On the other hand, and at least in some cases, the environment itself may very well turn out to be hostile to the computer due to the nature of the work. Therefore, the equipment should be able to support mechanical impacts with no damage and possibly other aggressions such as liquid spill or dusty environments. The system should be strong enough to cope with a careless treatment and should present improved characteristics in aspects such as physical impact response, resistance and durability of movable parts.
- *Simplicity of operation*; being targeted at people with cognitive impairment, simplicity of use is naturally a fundamental requirement for the equipment. Furthermore, the system should not present all sorts of user's options, but a very easy user interface suitable for an interaction with a wide range of different users. The simplest of the solutions will comprise a graphic display and some conceptual keyboard made from a small number of keys. This conceptual keyboard must take into account the lower level of recognition ability and manual precision of the potential users. On the other hand, and in order to increase its usefulness, the equipment must also be fully operational without requiring more than minor changes in the working environment
- *Expandability*; the need for interactive adaptation of software, both to different user needs and to different applications, may require different levels of hardware support, either at the CPU performance level and at the memory storage capabilities. The use of an IBM PC compatible platform has already been considered as the best approach, as this will ensure a higher degree of independence to technology obsolescence. This will also provide the simplest development path due to the large availability of low cost widely used PC compatible computers and components. The selected platform should also provide means to interface to other devices, either through serial or parallel ports, or through more powerful PCMCIA plug and play interface ports.
- *I/O devices*; A set of I/O devices independent from the standard PC compatible platform must be considered in order to enhance, complement or even substitute the hardware platform native ones by providing alternate forms of interaction both in the direction from the user to the system and in opposite direction. The aforementioned I/O devices, apart from their functional aspects, must provide an easy integration to the prototype at the hardware level.

B. User Interface Requirements

As previously stated, the potential users of computer assisted instruction and maintenance who suffers from intellectual disabilities will most probably have some kind of additional disabilities, such as some sort of physical or sensory impairments. Therefore, and due to the individualistic nature of the person involved, the design of the system should be broad enough to interact with a mixed variety of people. One should not, on the other hand, be tempted by the illusion that a universal system can be developed to answer all kinds of user needs and requirements. A well-balanced solution using a modular approach and capable of integrating different I/O devices and supporting adaptive software modules may therefore be the best approach to this question. This will imply, among other things, that an appropriately delimited target group of users should be defined, and their most significant characteristics, from the interaction point of view, be established. This specification work, as it deals with the technical integration, performed by engineering people, of the specific *User Interface requirements*, is only possible by means of a multidisciplinary team composed by engineers, psychologists, sociologists, special education professionals and social careers.

Therefore, a multidisciplinary team involving the different partners of the VICAID consortium has performed an assessment of the technical constraints, by evaluating practical options for the user/equipment interaction through the analysis of different possible technical solutions, both at hardware and software levels. As a result, several requirements were also defined for the input and output interface.

Regarding the input devices:

- There should be a total avoidance of any device that requires motor dexterity.
- The users will be allowed to interact with the system through a conceptual keyboard composed by a small number of physical keys (with a pre-defined meaning for each of them).
- The conceptual keyboard should require less manual precision and less recognition ability than a standard keyboard.
- Keys should be large and strong enough to support handling from heavy handed users.
- Both tactile and auditive feedback should be considered for the action upon the keys.
- The users must be able to provide the system feedback related with their own actions.

Regarding the output devices:

- The use of a colour display has been considered an important issue as a way of improving the capability of keeping the attention of at least some of the users.
- Sound, including voice music and audio signalling, either synthesised or recorded, should also be considered, as well as other forms of catching the user attention, such as flickering lights.
- As a consequence of the limited capacity of the user for self-decision making in situations of difficulty (when something unexpected happens or something goes wrong), the equipment must also contemplate some form of communication with the co-worker, so that he/she can be promptly notified, either automatically or by user initiative, of those critical situations.

C. *Programmability Requirements*

Programmability requirements emerge from two distinct needs. The first is the need to assure a modular software approach, able to be easily adapted, reviewed or updated. The second is the need to provide, at a higher level, the mechanisms that will allow the co-workers to develop the different prompt sequences, as well as to set the parameters and rules associated to each one of them. From the first of those needs, the following requirements emerge:

- Processing power must be enough to assure a response in real time to all predictable user actions. Otherwise, any delay may result in erratic behaviour of the user, due to its inability to understand why the machine does not react as he/she expects. This requirement will have a great impact in the processor selection.
- Memory capacity must also be big enough to support all the software, and should not contribute as a bottleneck to the equipment development. This means that it should be either over-dimensioned during early project phases or the hardware platform should support memory expansion capabilities.
- Due to the resilient nature of PC based architecture, and wide availability of well-established development tools both at hardware and software level, the selection of a PC compatible infra-structure is also and clearly a requirement.

As to the second of the above referred needs, task programmability by the co-workers may very well need to be performed in a different kind of equipment. This is due to the fact that the palmtop, with its conceptual keyboard, will not provide the necessary resources to be easily used as a programming device. Therefore, task development software will be run in standard desktop PC computers, and downloaded afterwards to the palmtop. On the other hand, evaluation of both user and software performance will also imply that recorded data on task evolution will need to be transferred to a remote computer. Therefore, as a result from these questions, the palmtop based system should be able to support some form of communication with the outside world.

IV. INSTRUCTIONAL MODEL

As a mean of consolidating the main characteristics of the user interface, the above *User Interface requirements* were evaluated using a top-down approach by cross-correlating them with an *Instructional model* which is also under development [4]. The *Instructional model* defines the way the prompting sequences will be delivered to the user and how each one of the prompts will relate both to the undergoing task and to the user performance. This model includes the following basic elements:

- Instructions.
- Reinforcement.
- Instructional re-adjustment.
- Automatic prompting.
- Corrective feedback.

A. *Instructions*

The instructions are to be delivered to the user as a sequence of prompts. Typically each prompt, with its associated icon, will have a direct connection with a specific step of the task (e.g., "open the dishwasher door"). Prompt presentation will be made using a visual and, possibly, an auditory mode:

- Visual instructions will consist of simple pictorial images with colour elements.
- Auditory instructions will be in the form of verbal statements.

In terms of the user interaction, the user will only need a single step command to require the next instruction. This step will take one of two possible forms:

- The users will be expected to push a key to ask for the next instruction in the sequence.
- The user will be instructed to pick up some kind of small device associated with the task step he/she has completed, and to provide it afterwards to the palmtop.

B. Reinforcement

At pre-set intervals, requiring the next instruction would not show the next prompt of the job task but rather a prompt indicating that the user should contact a co-worker (the work supervisor or somebody else) to have a check on his/her work or to have reinforcement (a positive remark, a sticker or some other form of pleasant event).

When a reinforcement instruction occurs the system should:

- Present an image of the co-worker that the user should contact.
- Present complementary information such as flickering lights or sound pattern.
- Interact with the co-worker in order to inform him that a reinforcement action should take place. This interaction could be performed by either an auditory unit (through a sound pattern) or a communication link established by a radio transmitter.

C. Instructional Re-adjustment

Instructional re-adjustment aims to reduce the number of performance errors by re-organising the sequence of instructions. This results from the fact that some of the steps in the prompt sequence may be performed in one of several ways, as long as all the steps are completed. Lets take the example of figure 2. If, for some reason the user, after completing step 1, executes step 4 instead of step 2 as it should (Fig 2.a), the system may still be able to re-organise the sequence successfully by prompting steps 3 and 2 sequentially after step 4, as in fig 2.b.

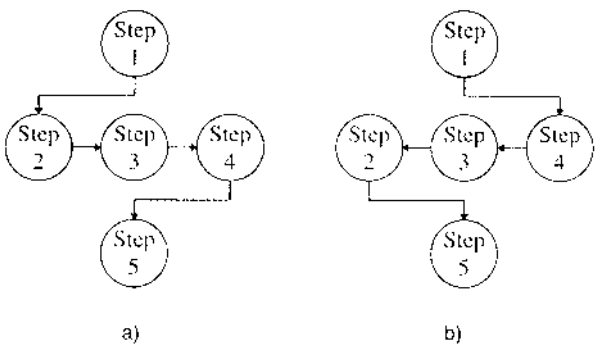


Fig. 2 - Instructional Re-adjustment.
a) Normal sequence. b) Re-organised sequence.
Steps shown in parallel can be re-organised in any order.

However, and to be aware of what have been done by the user, the system must keep track of his/her performance. This requires a dedicated mechanism that can be implemented either by using technology related with intelligent buildings or, in an ad-hoc way, by using sensors connected to the computer by some sort of radio link and differently coded for the various task steps. Both of these solutions are clearly complex and would not satisfy one of the general requirements of the system, which is to support disabled people on work settings with minor changes in the environment.

Being impossible to have a complete control of the users task performance, we should therefore consider different options where the users themselves provide the system with feedback related with their own actions. In this way the system will have a way to find out if the instruction sequence is correctly performed. In terms of user interaction this means that the user, after reading and performing the instruction, will inform the system of what he/she had done instead of asking for the next instruction. This is illustrated in the following sequence:

- The user reads the instruction from the system.
- The user performs the action.
- The user informs the system that the action has been performed.

We are aware that, by using this kind of mechanism, it is not possible to have non vulnerable communication between the user and the computer. Therefore, and to ensure the maximum success rate, special care should be taken while designing the user interface system. This system should provide not only the best possible communication, but also be very easy to learn by the user. To achieve these goals, and for the time being, the following three options have been considered:

- The user provides the computer with little self-identifiable chips related to the each one of the task steps. In this solution, providing the chip to the computer will be integrated as part of the user automatic response. The user, after performing an action, has to look for the chip related with that action. If the chip corresponds to the step that the computer has illustrated, the conclusion is that the user responded correctly. Thus the computer will proceed to the next instruction that may be a next step or a reinforcement instruction. If the chip does not correspond to the step that the computer has illustrated, the conclusion is that the user had produced an incorrect response. If such a response does not preclude the continuation of the task, the computer will provide the necessary reactions such as the re-organisation of the instruction sequence (instructional readjustment). This kind of solution requires a device connected to the palmtop able to receive and process the chips (chip reader).
- The user provides the computer with the same little self-identifiable chips but, in this case, only when instructed to do it by the prompt sequence. In this case the feedback related to the each one of the task

steps will be integrated as part of the task itself in the form of control points. This solution will simplify the practical constraints related to the strategic placement of the chips but reduces, on the other hand, the point by point control provided by the previous option with consequences at the instructional re-adjustment level.

- The user provides input to the system by selecting one out of two or three keys. It is clear, however, that a severely mental impaired user will not be able to tell the system, through variable and abstracts symbols, which activity he/she has done, as their ability to perform discrimination is highly reduced. This means that this mental model should be somehow simplified. One way of doing this is to consider that the selection will be done by means of comparison between different colours, shape or tactile characteristics associated with each one of the keys and one symbol that is drawn in the object that the user must manipulate to perform the action. In order to illustrate this let us consider the task "open the door". The user must follow several steps:
  - He/she reads from the system the action that he/she has to perform (to open the door).
  - He/she performs the action.
  - He/she finds which one of the three selection keys is associated with the symbol in the door by means of its colour, drawing, shape or tactile consistency.
  - He/she presses the appropriate key of the conceptual keyboard.

D. Corrective Feedback

The corrective feedback will alert the work supervisor whenever the user is not in the position to proceed in the task (he/she has made a mistake that can not be solved by instructional re-adjustment). This element can be implemented only if the system is aware of the users performance, which is a similar situation to the one described for the instructional re-adjustment.

The work supervisor can be alerted by the two following forms:

- The radio transmitter already introduced for the reinforcement, should also be considered to establish a communication link with the work supervisor for actions related with the corrective feedback.
- The auditory unity can be used to signal the situation (beep).

E. Automatic Prompting

The interval between two user requests (the period of time to perform a particular instruction) should not be smaller than a pre-set minimum execution time neither longer than a pre-set maximum execution time. Two different warnings should be therefore provided:

- If the interval between two user requests is smaller than a pre-set time for the particular instruction the system should not move to the next instruction but it should alert the co-worker (considering the short interval a sign of inappropriate performance).
- If the interval between two user requests exceed the maximum time allowed for that particular instruction, the system should alert the user and also the co-worker.

The first warning can be performed by using a radio transmitter, while the second one can be performed by using a radio transmitter together with the auditory unit (beep).

F. Consolidated View

Table I presents a summary of the intersection of the *Instructional model* with the *User Interface requirements* at the Input/Output devices level. Each one of these devices is evaluated in terms of need for each kind of *Instructional model* element. Its degree of importance will also be inferred in a later phase of the project to further enhance the proposed *Reference model*.

	Instructions	Reinforcement	Instruction Re-adjustment	Corrective Feedback	Automatic Prompting
Colour Display	•	•	•	•	•
Maximum number of symbols in the display	1	1	1	1	1
Next instruction key	•				
Selection keys			•	•	
Radio Transmitter		•		•	•
Beep				•	•
Sound patterns	•	•			
Flickering lights		•			•
Chip reader	•		•		

Table I - Consolidated view.

V. CONCLUSIONS

In this paper we have identified the technical approach for a definition of a *Reference model* that will support the development of a palmtop based system for vocational integration of intellectually disabled people. By using a volume intersection model and a systematic process analysis the major requirements of the system were identified. These requirements have been gathered into three different groups: *General requirements*, *User Interface requirements* and *Programmability requirements*. An independent analysis of the *User Interface requirements* group has been performed by cross-correlating them with an *Instructional model*, which is also under development.

Based on the presented work, several technical solutions were identified and are currently under discussion, which will lead to the implemetation of a first prototype of the system. This prototype will be evaluated in a laboratory environment (University of Leiden) during the second trimester of the next year. The result of this evaluation will be used to trimm both the prototype and the *Instructional model* resulting in the final system, which will be introduced in real life situations both at Leicester (Work-Place of Leicester) and Lisbon (Faculdade de Motricidade Humana) during the year of 1997.

REFERENCES

[1] F. R. Rusch, T. K. Morgan, J. E. Martin, M. Riva, M. Argan, "Competitive employement: Teaching mentally retarded employees self-instructional strategies". Applied Research in Mental Retardation, 6, 389-407, 1985.

[2] G. E. Lancioni, "Procedures for promoting independent activity in people with Severe and Profound Learning Disability", Mental Handicap Research, 7, 237-255, 1994.

[3] G. E Lancioni, D Oliva, "A computer-aided programme for promoting unsupervised activities for multihandicapped adolescents", Journal of mental Deficiency Research, 33, 313-322, 1988.

[4] Anthony J. Montgomery, Mark O'Reilly, G. E. Lancioni, "Decision models for Instructional Strategies for Learning Disabled Persons", VICAID (TIDE 1199) Deliverable, 1995.

ISDNLINK - Uma biblioteca de classes para RDIS

Osvaldo A. Santos, Fernando M. S. Ramos

Resumo- **Um programa genérico que utilize RDIS tem à sua disposição uma API que implementa apenas os protocolos de baixo nível da camada OSI. Este artigo descreve uma biblioteca de classes para Windows 3.x que implementa os serviços de alto nível geralmente utilizados em aplicações finais, tais como transferência de ficheiros e blocos de dados. Foi dada especial atenção à interface entre este módulo e o programa cliente, utilizando a estratégia método-evento, facilitada pelas potencialidades da linguagem C++, nomeadamente derivação de classes e métodos virtuais.**

A performance e flexibilidade foram outros aspectos tidos em conta, bem como aspectos de *multitasking* e operação em *background*.

Abstract- **A generic program which uses ISDN is confronted with an API that implements the OSI layer lower protocols only. This paper describes a Windows 3.1x class library, that implements the high level services usually required in final applications, such as file and memory block transfer. A special attention was taken concerning the aspects related to the interface between this library and the client application, using mainly the event-method strategie, made easy by using C++ programming language, namely inheritance and virtual functions.**

Performance and flexibility were another focused aspects, as well as multitasking and background operation aspects.

I. INTRODUÇÃO

O acesso básico RDIS[1] 2B+D disponibiliza dois canais B de 64 Kbit/s e um canal D de 16 Kbit/s, tornando-o atraente para aplicações de Televigilância, Telemedicina, Videoconferência, acesso a redes locais, etc [1].

A carta PCBIT, desenvolvida pelo INESC, permite aceder aos recursos deste tipo de acesso, incluindo os serviços de voz. A comunicação entre as aplicações e a carta PCBIT é feita através de uma API[2] fornecida pelo fabricante, que na prática é uma DLL[3] chamada ISDNBIOS.DLL, que implementa as primitivas básicas de comunicação. Estas primitivas são: REGISTER, RELEASE, PUTMESSAGE, GETMESSAGE, SETSIGNAL, DEINSTALL, INISDNBIOS, GETMANUFACTURER, GETVERSION, GETSERIALNUMBER e MANUFACTURER. Todas elas têm associadas as suas respectivas estruturas de dados, que permitem a troca da informação necessária a cada tipo de primitiva.

Uma vez que a API implementa apenas os protocolos de baixo nível do modelo OSI (até à camada 3), é implicitamente pouco aconselhável o seu uso directo em aplicações finais, uma vez que o programador teria que implementar os protocolos de alto nível do módulo de comunicações.

II. SERVIÇOS DE ALTO NÍVEL QUE UMA APLICAÇÃO GENÉRICA NECESSITA.

Básicamente, os serviços que uma aplicação genérica necessita são os seguintes: estabelecimento de ligações no canal B, terminação de ligação, transmissão e recepção de mensagens de controlo, transmissão e recepção de blocos de dados de tamanho variável e transmissão e recepção de ficheiros.

Na construção desta biblioteca de classes foi procurado um equilíbrio entre flexibilidade, simplicidade, independência e performance, vectores estes que nem sempre se podem conjugar facilmente:

-Flexibilidade, para permitir um conjunto diverso de serviços, que eventualmente serão usados concorrentemente e em *background*.

-Simplicidade, traduzida na escrita do menor número possível de linhas de código no programa cliente, para usar determinado serviço RDIS.

-Independência, de maneira a tornar o seu uso geral, independentemente do programa cliente.

-Performance, para aproveitar ao máximo a largura de banda oferecida pela RDIS e ocupar ao mínimo o CPU da máquina, permitindo assim ao programa cliente usufruir do máximo de recursos da máquina.

Nos próximos pontos será mostrado como cada um destes objectivos foi atingido, respeitando sempre os outros vectores.

III. *MULTITASKING* COOPERATIVO.

O sistema operativo Windows 3.1x permite às aplicações correrem concorrentemente apenas em *multitasking* cooperativo (fig. 1): quando uma aplicação tem o controlo do CPU as outras aplicações param, a menos que tenham rotinas de serviço a interrupção,

[1] Rede Digital com Integração de Serviços

[2] *Aplication Programming Interface*

[3] *Dynamic Link Library*

Assim, uma aplicação só tem acesso ao CPU quando as outras aplicações deixarem.

Uma aplicação típica recebe mensagens do sistema operativo, processa cada uma dessas mensagens o mais rapidamente possível e devolve em seguida o controlo ao sistema operativo. Na maior parte do tempo só a aplicação activa recebe mensagens, que correspondem geralmente a eventos gerados pelo utilizador através da interface gráfica.

Assim, uma aplicação em *background* arrisca-se a não ter tempo de CPU de maneira a poder efectuar uma tarefa em tempo-real. Uma maneira de obter regularmente tempo de CPU é utilizar um *timer*. Esta facilidade permite que o *kernel* do sistema envie regularmente uma mensagem para a aplicação, mesmo quando esta se encontra em *background*, sendo o intervalo de tempo programável. No entanto, isto só é possível se as outras aplicações forem bem comportadas.

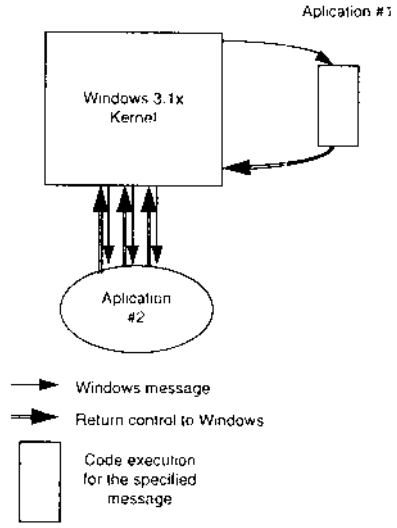


Fig. 1 - *multitasking* cooperativo no sistema operativo Windows 3.x..

ou seja, processarem rápidamente as mensagens e devolverem o controlo ao sistema.

IV. A ESTRATÉGIA MÉTODO-EVENTO.

Uma das estratégias mais simples de interagir com um módulo de software é a política método-evento (fig 2), utilizada com grande sucesso, por exemplo, no Microsoft Visual Basic. Através desta política, além de ser possível utilizar os serviços de determinado objecto, esse mesmo objecto pode ter capacidades de notificação de eventos, que permitem chamar funções do programa cliente sempre que seja necessário.

Assim, o programa cliente apenas tem que construir o código adequado para cada um dos eventos gerados pelo objecto, tal como no Visual Basic. Esta estratégia revelou-se de uma extrema simplicidade e flexibilidade, tal como será demonstrado.

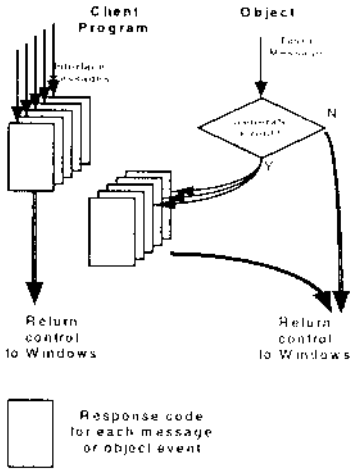


Fig. 2 - estratégia método-evento

V. VANTAGENS DA UTILIZAÇÃO DE C++.

Além das vantagens naturais de organização de software, tais como encapsulamento de dados, maior facilidade de manutenção ou capacidade de herança, a grande vantagem aplicada neste módulo é a capacidade de redefinir métodos virtuais da classe base. Esta capacidade é utilizada para permitir à classe base notificar as suas classes derivadas da existência de eventos. Como os métodos da classe derivada são definidos na aplicação cliente, o código a executar para cada evento é específico de cada aplicação final que use a biblioteca.

VI. DIAGRAMA DE CLASSES.

A fig. 3 mostra a estrutura de classes da biblioteca, onde se podem ver as diferentes relações entre classes: herança ou uso. As classes virtuais ISDNLINK e LINK servem básicamente para permitir a interface entre as classes base PBITLINK e DATALINK e as suas respectivas classes derivadas.

As classes actuam como processos virtuais, uma vez que

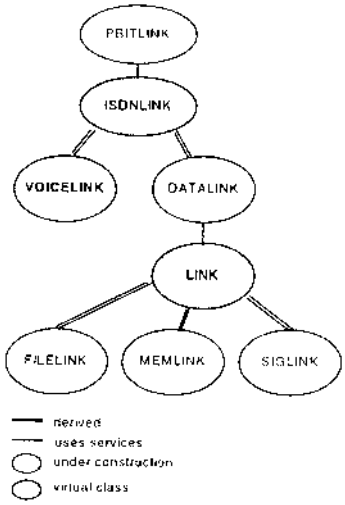


Fig. 3- diagrama de classes

obtêm regularmente tempo de CPU. Essa atribuição de tempo de CPU é feita chamando regularmente o método TimeSlice() de cada classe.

Para evitar que seja a aplicação cliente a fazer pooling a esses objectos, foi implementado um timer que chama o método TimeSlice() da classe PBITLINK em cada 20 ms (fig. 4). Os objectos da classe ISDNLINK, ou de classes derivadas desta, registam-se na classe PBITLINK quando são criados, ou seja, a classe PBITLINK sabe o endereço desses objectos. Então, basta chamar o método TimeSlice() de cada um destes objectos para garantir que todos eles têm tempo de CPU regularmente, comportando-se assim como processos virtuais, permitindo a operação em background.

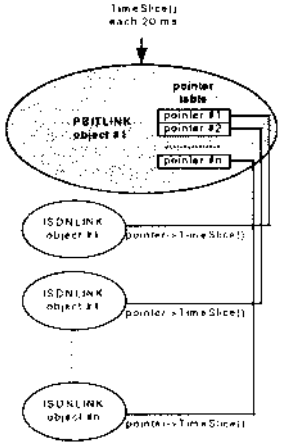


Fig. 4 - distribuição de tempo de CPU

VII. A CLASSE PCBITLINK.

Esta classe tem duas grandes funções: fazer a interface com a API e distribuir o tempo de CPU pelos objectos ISDNLINK. A distribuição de tempo de CPU já foi analisada anteriormente. Quanto à interface com a API, resume-se à distribuição de mensagens no sentido API -> objectos ISDNLINK, uma vez que estes objectos têm capacidade de usar directamente a API.

As mensagens podem ser caracterizadas em dois tipos de acordo com o seu destino: mensagens destinadas a um determinado objecto ISDNLINK, por exemplo as mensagens utilizadas durante a transferência de ficheiros, e mensagens com destino a um novo objecto ISDNLINK, por exemplo as mensagens recebidas durante a fase de estabelecimento de ligação. As mensagens são encaminhadas de acordo com o seu identificador de ligação, que é processado pela API. Quando as mensagens não têm identificador, isso significa que são mensagens de estabelecimento de ligação, e neste caso são direccionadas para o primeiro objecto ISDNLINK livre. A fig. 5 descreve o algoritmo de encaminhamento de mensagens.

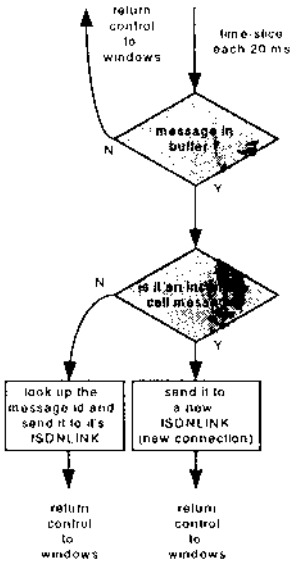


Fig. 5 -distribuição de mensagens na classe PBITLINK

VIII. A CLASSE BASE ISDNLINK.

A classe ISDNLINK tem como função a interface entre a classe PBITLINK que distribui as mensagens vindas da API e as classes DATALINK e VOICELINK que são as responsáveis respectivamente pelas ligações de voz e dados.

Segue-se a declaração da classe:

```
#ifdef ISDN_DLL
 #define TIPO_CLASSE _export
#else
 #define TIPO_CLASSE _import
#endif

class TIPO_CLASSE ISDNLINK
{
private:
   WORD ID;
   DWORD Father;
   WORD wRC; //API connection identifier.
   BYTE Uso; //Object state: FREE ou BUSY
   BYTE Estado;

public:

   far pascal _export ISDNLINK();
   far pascal _export ~ISDNLINK();
   SetLowerLevelLink(DWORD llink,WORD id);

   virtual WORD TimeSlice(void);
   virtual LowerLayerIsDisconnecting();
   virtual GetEstado(void);
   virtual SetLowerLevelLink();
   virtual RemoveLowerLevelLink();
```

```
    virtual IL_ConnectInd(LPBYTE ptr);
    virtual IL_EstablishLapDInd(LPBYTE ptr);
    virtual IL_DataLapDInd(LPBYTE ptr);
    virtual IL_DisconnectInd(LPBYTE ptr);
    virtual IL_InfoInd(LPBYTE ptr);
    virtual IL_ConnectActiveInd(LPBYTE ptr);
    virtual IL_ConnectActiveConf(LPBYTE ptr);
    virtual IL_SelectProtocolConf(LPBYTE ptr);
    virtual IL_ReleaseLapDConf(LPBYTE ptr);
    virtual IL_DisconnectConf(LPBYTE ptr);
    virtual IL_ConnectConf(LPBYTE ptr);
    virtual IL_EstablishLapDConf(LPBYTE ptr);
    virtual IL_DataLapDConf(LPBYTE ptr);
};
```

Analisando o protótipo da classe, é muito importante referir que TIPO_CLASSE é <_export> ou <_import> conforme se está a compilar a biblioteca ou o programa cliente, permitindo assim exportar ou importar a classe. Esta particularidade é implementada em todas as classes. De salientar que nem todos os compiladores permitem importar classes de DLL's; foi usado o compilador Borland C++ 4.0 que é um dos que permitem esta facilidade. Os métodos SetLowerLevelLink(), LowerLayerIsDisconnecting() e RemoveLowerLevelLink() têm como função a ligação à classe de nível inferior, PBITLINK, e permitem a essa classe gerir a tabela de ISDNLINKs. O método TimeSlice(), presente em todas as classes, é utilizado para obter tempo de CPU, sendo chamado regularmente pela classe hierarquicamente inferior.

Os métodos virtuais IL_xxxxx correspondem a mensagens vindas da API, e são chamados pela classe PBITLINK quando faz a distribuição de mensagens. Esta distribuição tem em conta o identificador de ligação, a variável wRC. Como estes métodos são virtuais, são as classes derivadas de ISDNLINK que vão processar cada uma destas mensagens.

IX. A CLASSE DATALINK.

Esta classe faz a gestão de chamadas de dados sobre o canal B (fig. 6). As suas responsabilidades são: estabelecimento e terminação de ligações de dados incluindo a montagem dos protocolos adequados das camadas 2 e 3 do modelo OSI e suporte e gestão de objectos de nível superior, que correspondem às classes do tipo LINK. Os serviços que esta classe fornece aos objectos LINK são basicamente transmissão e recepção de pacotes.

O estabelecimento de uma ligação de dados em modo circuito é feito através da troca de mensagens apropriadas (figs.6, 7 e 8).

Os métodos e variáveis mais importantes desta classe são:

```
class TIPO_CLASSE DATALINK:public ISDNLINK
{
  private:

    //Callbacks
    virtual ConnectActiveIndCallback(char far
*address,char far *subaddress,
IEUserToUserSignalling UTUS);
 virtual ConnectActiveConfCallback(void);
 virtual DisconnectIndCallback(void);
 virtual DisconnectConfCallback(void);
 virtual ConnectConfCallback(void);
 virtual ConnectActiveProtocolsCallback(void);

  public:

 //Gestao de Links de nivel superior
 LINK *LinkTable[MAXIMO_LINKS];
 WORD LinksUsed;

 ConnectRequest(char far *Number,char far
*SubAddress=""); //Vai para a API
```

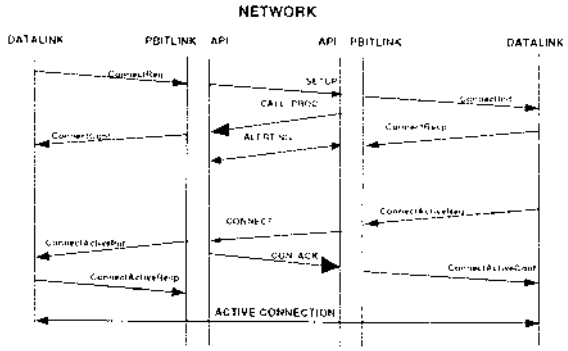


Fig. 6 - Sinalização no canal D necessária para o estabelecimento de uma ligação em comutação de circuitos no canal B.

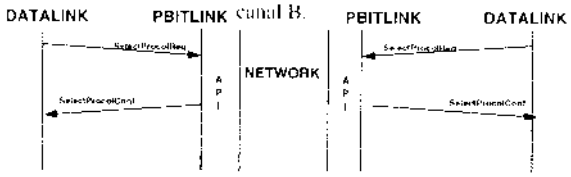


Fig. 7 - Selecção do protocolo do nível 2 da pilha de protocolos do utilizador.

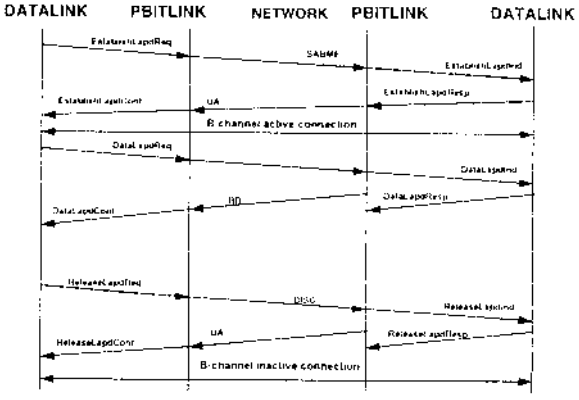


Fig. 8 - Mensagens (canal B) necessárias para o estabelecimento de uma ligação LAPD no canal B.

```
SendData(LPBYTE Data,WORD Len);
IL_ConnectInd(LPBYTE ptr);
IL_EstablishLapDInd(LPBYTE ptr);
IL_DataLapDInd(LPBYTE ptr);
IL_DisconnectInd(LPBYTE ptr);
IL_InfoInd(LPBYTE ptr);
IL_ConnectActiveInd(LPBYTE ptr);

IL_ConnectActiveConf(LPBYTE ptr);
IL_SelectProtocolConf(LPBYTE ptr);
IL_ReleaseLapDConf(LPBYTE ptr);
IL_DisconnectConf(LPBYTE ptr);
IL_ConnectConf(LPBYTE ptr);
IL_EstablishLapDConf(LPBYTE ptr);
IL_DataLapDConf(LPBYTE ptr);

SetLink(LINK *HLink,WORD Id);
TimeSlice();
} ;
```

Os métodos virtuais xxxxCallback() são usados para notificar a aplicação final do progresso da fase de estabelecimento e terminação de ligação. De modo a usar esta facilidade, a aplicação final terá que construir uma classe própria, derivada de DATALINK.

A tabela de ponteiros de LINKS, LinkTable, serve para fazer a gestão dos LINKs de nível superior, que vão usar esta classe como servidor de pacotes. Como foi utilizado um array de ponteiros para uma classe virtual, facilmente se implementam classes de alto nível que possam usar a classe DATALINK de um modo transparente. Um bom exemplo disto é a classe SIGLINK, que foi desenvolvida já depois desta biblioteca estar terminada. O tempo de desenvolvimento e integração foi de cerca de uma hora !

Os métodos IL_xxxxx são a implementação dos métodos virtuais da classe base ISDNLINK, e como cada um deles corresponde a uma mensagem trocada com a API, estes contêm o código resposta a essa mensagem. Estes métodos são chamados pela classe PBITLINK, de acordo com a política de distribuição de mensagens explicada anteriormente.

Assim, uma aplicação final terá que incluir a biblioteca de classes, e conter o seguinte código:

Exemplo 1: sem notificação.

```
DATALINK Channel1;

void ExemploEstabelecerLigacaoDados(void)
 {
   Channel1.ConnectRequest("370563","0");
 }

void ExemploTerminarLigacoes(void)
 {
   Channel1.DisconnectRequest();
 }
```

Exemplo 2: com notificação da aplicação cliente.

```
//prototyping

//criação de uma nova classe derivada de
DATALINK.
class B1:public DATALINK
{
  public:
      //métodos redefinidos
      WORD FAR PASCAL  export
   DisconnectIndCallback();
      WORD FAR PASCAL _export
   ConnectActiveProtocolsCallback(void);
   };

//main program

B1 Channel1;

void ExemploEstabelecerLigacaoDados(void)
 {
   Channel1.ConnectRequest("370563","0");
 }

void ExemploTerminarLigacoes(void)
 {
   Channel1.DisconnectRequest();
 }

//redefinição do método virtual
//callback de ligação activa, é chamada quando
os protocolos estão montados.
B1::ConnectActiveProtocolsCallback(void)
 {
  interface("Ligação activa");

 }

//redefinição do método virtual
//callback de terminação de ligação, é chamada
quando a ligação é terminada
B1::DisconnectIndCallback()
 {
  Interface("Ligação terminada");
 }
```

X. A CLASSE BASE LINK.

Esta classe virtual tem como função a interface entre o DATALINK e as classes de nível superior FILELINK, MEMLINK, SIGLINK e outras que poderão ser implementadas fácilmente sobre esta estrutura.

Cada LINK (fig. 9) comporta-se como um canal de dados virtual, de largura de banda variável, que pode chegar até muito próximo dos 64 Kbit/s se, por exemplo, todos os outros LINK's estiverem inactivos. Se vários LINKs

estiverem activos, a largura de banda é distribuída equitativamente, permitindo assim vários canais de transmissão de dados paralelos e independentes. Esta facilidade é importante, por exemplo, para a troca de mensagens de controlo em simultâneo com a transmissão de um ou mais ficheiros.

Analisando o protótipo da classe, facilmente é verificado o carácter de servidor de pacotes da classe DATALINK. De facto, o método PacketReception() da classe LINK é chamado da classe DATALINK sempre que a esta chega um pacote com esse destino. Além de servidora de pacotes, a classe DATALINK também atribui tempo de CPU para que cada objecto LINK envie pacotes, caso necessite. Esta atribuição de tempo de CPU permite, por exemplo, enviar ficheiros em *background*.

```
class TIPO_CLASSE LINK
{
  private:
   WORD ID;
   DWORD Father;

  public:
   far pascal _export LINK();
   far pascal _export ~LINK();

SetLowerLevelLink(DWORD llink,WORD Id);
        virtual WORD far pascal _export
PacketReception(LPBYTE Data,WORD Len);
        virtual WORD far pascal _export
TimeSlice(void);
        virtual WORD far pascal _export
LowerLayerIsDisconnecting();
};
```

Os LINKs são reconhecidos pelo seu identificador que não passa de um simples número. Assim, as mensagens vindas do LINK com o identificador 2 numa aplicação são enviadas pela classe DATALINK para o LINK com o identificador 2 na outra aplicação. O número máximo de LINKs é de 256 por cada DATALINK, permitindo assim, por exemplo, a transferência de 256 ficheiros simultâneamente, a uma taxa média de 250 bits/s. O número do LINK é atribuído no programa cliente quando é registado num DATALINK. Exemplo:

```
DATALINK Channel1;
FILELINK FileService;

//Aqui regista-se um LINK de transmissão de
ficheiros sobre o DATALINK Channel1 com o
identificador 5.

FileService.Init(&Channel1,5);
```

Para a classe DATALINK poder gerir o destino dos pacotes, foi introduzido o identificador do LINK no primeiro byte de cada pacote de dados. Os bytes seguintes são bytes de controlo específicos de cada tipo de LINK.

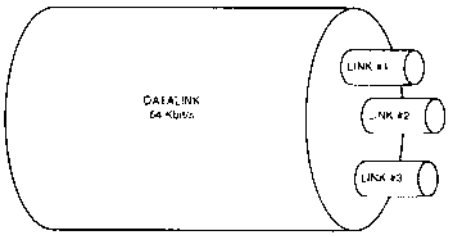


Fig. 9 - canais de dados virtuais sobre um DATALINK

XI. A CLASSE FILELINK.

Esta classe, derivada da classe LINK, é responsável pelo envio e recepção de ficheiros. Ambas as operações são efectuadas em *background*, deixando o processador livre para quaisquer outras operações. Os métodos principais desta classe são:

```
class TIPO_CLASSE FILELINK:LINK
 {
   ...
   public:

     WORD Init(DATALINK *LowLayer,WORD Id);
     WORD SendFile(char far *Name);

     ...
  virtual TxFileStart(char *Name);
  virtual TxFileEnd(WORD flag);
  virtual TxFileProgress(DWORD Bytes,DWORD
 Total);
   virtual RxFileStart(char *Name);
virtual RxFileEnd(WORD flag);
virtual  RxFileProgress(DWORD  Bytes,DWORD
Total);
 };
```

Para poder utilizar um objecto desta classe, é necessária a sua correcta inicialização sobre um objecto DATALINK. Para enviar um ficheiro basta chamar o método SendFile() com o nome do ficheiro como parâmetro. A recepção de ficheiros é feita automáticamente, bastando inicializar correctamente o objecto FILELINK.

Os métodos virtuais TxFile___ e RxFile___ têm como função a notificação da aplicação cliente do progresso da transmissão e recepção de ficheiros. Para utilizar esta facilidade basta criar uma nova classe na aplicação cliente, derivada de FILELINK, e redefinir estes métodos, escrevendo o código resposta a cada um deles.

Exemplo de utilização, pressupondo ligação estabelecida:

```
********* programa transmissor ********

DATALINK Channel1;
FILELINK FileService;
```

```
//Vai utilizar o identificador
//de link número 3 por exemplo

FileService.Init(&Channel1,3);
FileService.SendFile("teste.bmp");


********* programa receptor ********

DATALINK Channel1;
FILELINK FileService;

//Vai utilizar o identificador
//de link número 3

FileService.Init(&Channel1,3);

********* programa receptor com notificação
********

   //prototyping

   class FILET:FILELINK
    {
        RxFileStart(char *Name);
        RxFileEnd(WORD flag);
        RxFileProgress(DWORD Bytes,DWORD
   Total);
  };

//main program

DATALINK Channel1;
FILET FileService;

//Vai utilizar o identificador
//de link número 3

FileService.Init(&Channel1,3);


//redefinição do método virtual
FILET::RxFileStart(char *Name)
{
  Interface("Recebendo o ficheiro %s",Name);
}

//redefinição do método virtual
FILET::RxFileEnd(char *Name)
{
  Interface("Ficheiro %s transferido",Name);
}

//redefinição do método virtual
FILET::RxFileProgress(DWORD Bytes,DWORD Total)
{
       Interface("Recebendo %ld bytes de
%ld",Bytes,Total);
}
```

XII. A CLASSE MEMLINK.

Esta classe é similar à classe FILELINK, excepto no tipo de informação transmitida: blocos de memória. Existem dois métodos de envio destes blocos:

```
class TIPO_CLASSE MEMLINK:LINK
 {
   ...
  public:
   SendData(HGLOBAL Hmem);
  SendData(HPBYTE Data,DWORD Size,WORD
Mode=MEM_NORMAL);
   ...
 }
```

O primeiro método é passar simplesmente ao objecto o *handle*[4] do bloco de memória desejado. O segundo método requer o ponteiro para o bloco e o tamanho do bloco. Este segundo método permite dois modos de operação: normal e modo cópia. No modo cópia o bloco é copiado, garantindo assim que os dados não são alterados durante a transmissão.

Na parte do receptor é alocado um bloco de memória com o tamanho necessário, sendo o *handle* desse bloco passado à aplicação cliente quando a transferência chega ao fim.

XIII. A CLASSE SIGLINK.

Esta classe fornece serviços de troca de mensagens de controlo, tipicamente com apenas alguns bytes de comprimento. O tamanho da mensagem foi limitado ao tamanho máximo de um pacote, que é 1920 bytes. Exemplo de utilização:

```
********* programa transmissor ********

DATALINK Channel1;
SIGLINK ControloCamara;

//Vai utilizar o identificador
//de link número 7 por exemplo

ControloCamara.Init(&Channel1,7);
...
ControloCamara.SendMessage(MOVE_RIGHT);
...
ControloCamara.SendMessage(STOP);
```

[4] identificador de um bloco de memória protegida, alocada e gerida pelo sistema operativo.

```
********* programa receptor ********

//prototyping

class CAMCONTROL:SIGLINK
 {
   public:
    ReceiveMessage(LPBYTE Data,WORD Len);
 };

//main program

DATALINK Channel1;
CAMCONTROL ControloCamara;

//Vai utilizar o identificador
//de link número 7 por exemplo

ControloCamara.Init(&Channel1,7);

//redefinição do método virtual
CAMCONTROL::ReceiveMessage(LPBYTE Data,WORD Len)
 {
   ...
    //processamento das mensagens
   ...

 };
```

XIV. ANÁLISE DE DESEMPENHO.

Para avaliar o desempenho da biblioteca foi feito um conjunto de testes, envolvendo medida de tempos de estabelecimento e terminação de ligação, e transferência de ficheiros e blocos de memória.

O primeiro teste consistiu em medir os tempos de estabelecimento e terminação de ligação. Os testes foram feitos entre dois PC's 486DX2 66MHz, usando a rede pública. Num dos PC's foi medido o tempo gasto, utilizando o relógio interno da máquina.

Connect Time (D channel)	Connect Time (D+B channels)	Disconnect Time (B+D channels)
3.07 s	4.12 s	220 ms

Tabela 1 - tempos de estabelecimento e terminação de ligação, usando a rede pública.

O segundo teste foi a transferência de um único ficheiro, medindo o tempo gasto nessa transferência. Analisando a tabela 2 conclui-se que a taxa de transferência foi 62.3 Kbit/s, com uma eficiência de 62.3/64.00 = 97.3%. Não foi atingida uma eficácia de 100% devido aos tempos de leitura e escrita no disco duro, bem como ao *overhead* de controlo presente em todos os pacotes enviados para a rede.

	Size (bytes)	Time spent (s)	Transfer rate (Kbit/s)	Efficiency (%)
File	2,698,537	346.3	62.3	97.3
Memory Block	384,000	48,3	63.6	99.4

Tabela 2 - desempenho da transferência de ficheiros e blocos de memória

O terceiro teste consistiu em transferir um bloco de memória, medindo o tempo de transmissão. Analisando a tabela 2, verifica-se um aumento da eficácia para quase 100%, devido à inexistência de operações de leitura ou escrita em disco, sendo o *overhead* de controlo dos pacotes o único responsável pelos 0,6% de desperdício da largura de banda disponível no canal B.

XV. CONCLUSÕES.

Os objectivos iniciais foram cumpridos: foi construída uma biblioteca de classes que permite integrar facilmente comunicações RDIS em aplicações finais, de uma forma rápida e flexível. Apesar de ser implementada numa linguagem orientada a objectos, não se registaram efeitos negativos a nível de velocidade de execução.

De momento esta biblioteca está a ser usada no projecto Televigilância para Windows e Sistema de Videotelefonia, estando prevista a sua extensão aos projectos Teleradiologia e Teletrabalho.

A evolução desta biblioteca prevê um módulo de compressão de dados em tempo real, o que permitirá um aumento da largura de banda virtual média para cerca de 128 Kbit/s, se considerarmos uma taxa de compressão média de 2:1. Um outro aspecto importante será a possibilidade de atribuição de prioridades aos LINK's, permitindo à aplicação final definir quais as operações mais importantes.

REFERÊNCIAS

[1] Mário Serafim Nunes, Augusto Júlio Casaca, "Redes Digitais com Integração de Serviços", Editorial Presença, 1992.

Ajudas Computacionais para a Tele-operação de um Manipulador Robótico

Filipe M. Silva, Francisco Vaz*, João G.M. Gonçalves**

*Departamento de Electrónica e Telecomunicações, Universidade de Aveiro
**Comissão das Comunidades Europeias, JRC - Ispra (ITÁLIA)

Resumo - **Neste trabalho apresenta-se uma experiência da utilização de tecnologias robóticas na realização de operações de verificação remota em áreas de armazenamento de materiais físseis. Neste sentido, faz-se uma descrição dos aspectos funcionais do sistema e das ajudas computacionais desenvolvidas. O sistema é constituído por um braço manipulador operado remotamente a partir de uma estação local de controlo. A operação remota pressupõem a necessidade de cooperação entre o homem e a máquina baseada em estruturas gráficas apropriadas. A *interface* homem-máquina implementada apresenta novos conceitos em relação às arquitecturas clássicas na área, devido à integração num único écran de toda a informação relevante para o operador do sistema: *interfaces* gráficas de controlo, visualização de sinais vídeo e de ambientes virtuais.**

Abstract - **This work presents an experience in using robotics technologies to perform remote verification tasks inside a fissile material storage area. In this sense, it is described the functional aspects of the system and the developed computer aids. The system is composed of a manipulator arm remotely operated from a local system's operator console. The remote operation assumes the need of cooperation between man and machine based on appropriated graphical structures. The human-machine interface presents new concepts in comparison with more classical techniques in the area, due to the complete integration into a single screen of all relevant information needed by the system's operator: control interfaces, visualisation of video signal and virtual environments.**

I. INTRODUÇÃO

A necessidade de armazenamento de grandes quantidades de material físsil, associado ao desenvolvimento da tecnologia electrónica e dos computadores, veio justificar os esforços efectuados para a aplicação de tecnologias robóticas na assistência de operações remotas de verificação e inspecção. O projecto SAOV [1,2,3] visa o desenvolvimento de um sistema integrado de robótico móvel capaz de realizar autonomamente tarefas de verificação (e.g., inspecção visual, medições e leituras) remota em áreas de armazenamento de materiais radiactivos. O sistema robótico pode ser dividido em duas partes distintas: (1) a parte móvel que opera remotamente, constituída por um veículo móvel que suporta um braço articulado, sensores (e.g., câmaras TV, ultra-sons, *laser range finder*) e outro equipamento associado (e.g., unidades de *pan-and-tilt*, lentes motorizadas de *zoom* e focagem); e (2) a estação gráfica de controlo localizada fora da área de armazenamento.

Este artigo descreve o trabalho relacionado com o desenvolvimento das ajudas computacionais para a operação remota do braço manipulador, com especial destaque para os esquemas de controlo do manipulador e para o desenvolvimento da *interface* homem-máquina. O sistema manipulador será operado manualmente com base na informação visual presente na estação local. Procura-se demonstrar a combinação de um sistema de manipulação multi-sensorial com uma *interface* homem-máquina que permita obter um desempenho aceitável dadas a especificidade da aplicação. O desenvolvimento deste trabalho é baseado na utilização de *standards* industriais, tal como linguagem C, Unix, X-Window (X11R5+PEX), Motif, PHIGS. Para além de uma razão de custos, o uso de *standards* tem a vantagem de encorajar futuros desenvolvimentos e facilitar substancialmente a portabilidade do sistema.

II. ARQUITECTURA DO SISTEMA

O sistema de manipulação é constituído pelos seguintes módulos funcionais:

- a consola de operação:
  - estação gráfica SunSPARC10/41 (Unix, NFS, X11, Motif, PHIGS).
- o braço manipulador:
  - 6 graus de liberdade (Robosoft GT-6A), pinça, codificadores ópticos incrementais (um para cada eixo), computador 68020 num sistema VME, sistema operativo tempo real - Albatros.
- o sistema vídeo:
  - duas câmaras TV, unidades de *pan-and-tilt* e lentes motorizadas de *zoom* e focagem.

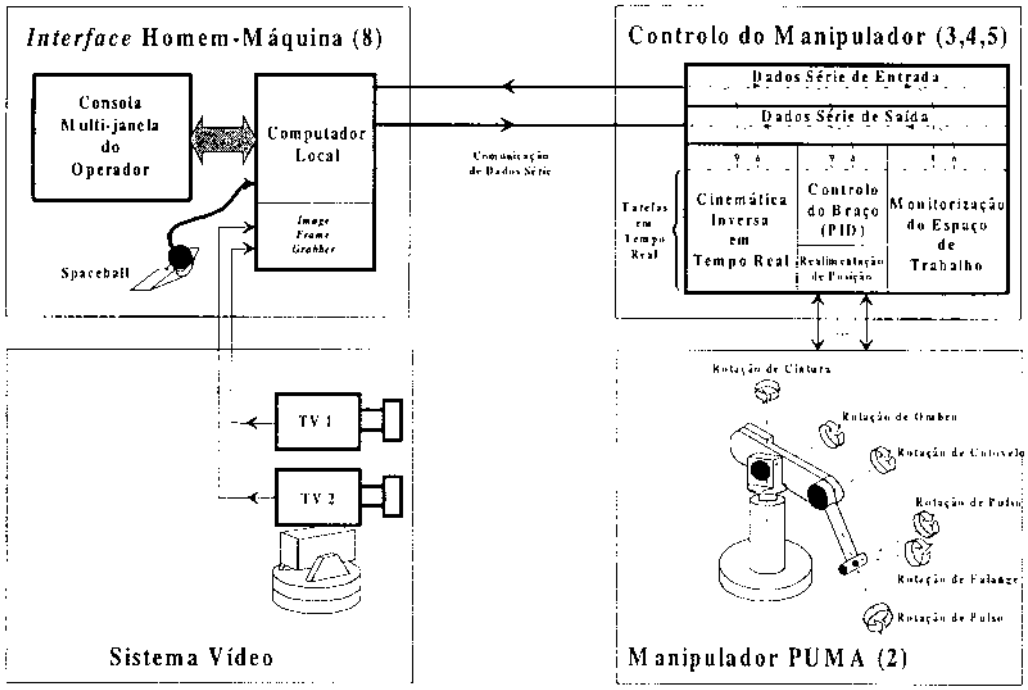


Fig. 1 - Arquitectura do Sistema.

A arquitectura do sistema está representada na figura 1. O sistema foi projectado assumindo que todas as operações a realizar são directamente controladas pelo operador com base em realimentação visual sempre disponível. Isto significa que o operador pode considerar-se incluído dentro do ciclo de controlo (*Man-in-the-Loop*). Numa fase posterior será de esperar a inclusão de operações semi-automáticas e automáticas. Neste contexto, autonomia significa que o ciclo de controlo se fecha através do computador do sistema remoto, dos seus sensores e actuadores. Em qualquer das situações é importante o conceito de controlo supervisionado em que o operador continuamente monitora e interactivamente modifica a operação do sistema, permitindo a intervenção humana em tempo real.

A. Características Geométricas e Mecânicas do Braço

O manipulador robótico é constituído por diversos elos mecânicos, supostamente rígidos, interligados por articulações rotacionais (proporcionam movimentos de rotação em torno do seu eixo - símbolo R). No tocante ao número de articulações, a sua estrutura integra seis eixos através de dois subsistemas principais: o braço (três primeiros eixos) e o punho (três eixos finais). Neste caso, o braço permite aceder a qualquer ponto do espaço operacional, enquanto que o punho permite orientar a mão. A figura 1 ilustra a geometria do manipulador utilizado neste trabalho. A estrutura adoptada, do tipo 6R, revela-se a estrutura com mais potencialidades e que conduz a um volume de trabalho máximo. As juntas articuladas são actuadas por motores DC (tensão nominal de 48V) acoplados a redutores de engrenagens.

B. Sistema Controlador

O controlador robótico consiste num computador VME equipado com um processador 68020 correndo Albatros, um sistema operativo multi-tarefa e tempo real. O esquema de controlo implementado no manipulador é o método proporcional-integral-derivativo (controlador PID) de eixos individuais. Esta técnica de controlo refere-se ao espaço das juntas articuladas e não toma em consideração aspectos dinâmicos do braço manipulador. A um nível mais alto, o sistema operativo proporciona o controlo independente da posição e velocidade de cada junta articulada.

C. Sensores

O sistema robótico incorpora os seguintes sensores:

- odômetros: cada junta articulada tem acoplado um codificador óptico incremental. Os codificadores e os contadores associados constituem a base dos odômetros. Os dados odométricos, que consistem na quantificação dos deslocamentos angulares dos seis eixos, estão disponíveis na execução dos processos internos de controlo e como informação de realimentação na reconstrução da sua representação virtual na estação local.
- câmaras TV: duas câmaras montadas sobre unidades de *pan-and-tilt* estão equipadas com lentes motorizadas para focagem e *zoom* automático. Torna-se possível obter uma visão global da operação do sistema, sem perda de detalhes importantes.

D. Interface Homem-Máquina

O operador humano está envolvido no controlo e supervisão do sistema remoto assumindo a existência de realimentação visual. Esta característica torna a *interface* homem-máquina uma peça fundamental do sistema. Deste modo, a *interface* homem-máquina foi preparada com o objectivo de proporcionar um controlo supervisionado realista e intuitivo, e que permita ao operador:

- monitorar continuamente a operação usando visualização de sinais vídeo ou ambientes virtuais.
- interactivamente intervir e modificar os objectivos pela aplicação de comandos de controlo, quer operando a Spaceball quer usando o rato na *interface* gráfica.

Todos os esforços foram concentrados na integração num único écran de toda a informação relevante ao operador (e.g., informação visual das câmaras TV, módulos gráficos de controlo, modelos virtuais 3D animados). Os objectivos principais a alcançar com esta implementação são:

- procurar que o próprio operador se sinta dentro do ambiente remoto (tele-presença);
- dar uma visão integrada do estado dos diferentes dispositivos e sensores remotos;
- permitir que o operador em cada momento seleccione os módulos que julgue relevantes.
- maximizar a participação do operador no controlo e monitorização do sistema remoto.

E. Arquitectura do Software

A descrição da arquitectura do sistema torna claro que algumas das tarefas são executadas na estação central enquanto outras usam o computador do braço robótico. Esta arquitectura a dois processadores leva a questionar a sua eficiência e quais as estratégias a adoptar com vista à sua optimização. Foram quatro os critérios usados para estabelecer o modo de agrupar e localizar os processos nos dois sistemas físicos [2]:

1) as tarefas orientadas para o *hardware* ou próximas dos sensores executam no computador do robot, enquanto as aplicações orientadas para a aplicação, incluindo a *interface* homem-máquina, correm na estação de operação local.
2) minimizar o tráfico através do canal de comunicações;
3) minimizar o atraso no ciclo de controlo;
4) proporcionar um nível de modularidade e expansibilidade de acordo com futuras exigências.

Estes critérios deram origem à arquitectura representada no diagrama de blocos da figura 2. Uma tarefa que pode parecer excepção é a que se relaciona com os processos de cinemática que, embora sejam orientados para a aplicação, são muito influenciados pelos sensores e pela interacção com o operador. Quando executados na estação central, estas características dão origem a um grande tráfico no canal de comunicações, com consequentes atrasos no ciclo de controlo e fortes limitações no desempenho global do sistema.

Toda a comunicação entre a consola do operador e o braço robótico é obtida através do Communications Server [4]. Este processo é composto de duas tarefas: uma correndo ao nível da estação local e a outra ao nível do sistema remoto de manipulação. Para além disso, existem tarefas em *background* que monitorizam o estado das comunicações entre o robot e a estação local. Quando ocorre uma perda de comunicação (e.g., devido a interferência rádio), o sistema remoto pára imediatamente e entra num estado de *standby*. Em tal situação, o operador do sistema é imediatamente informado.

Do lado da estação local, o operador do sistema usa um dispositivo de entrada com seis graus de liberdade, a Spaceball [5], para controlar o braço manipulador baseando-se somente em informação de realimentação visual - conceito de *Man-in-the-Loop*. Os comandos entre a estação de operação e o computador do braço são transmitidos via ligação RS232 (cabo ou rádio). Estes comandos são interceptados pelo Communications Server, a correr no computador do robot, que os redirecciona para a tarefa de destino. Os processo no sistema robótico são executadas concorrentemente pelo Albatros, o sistema operativo em tempo real [6].

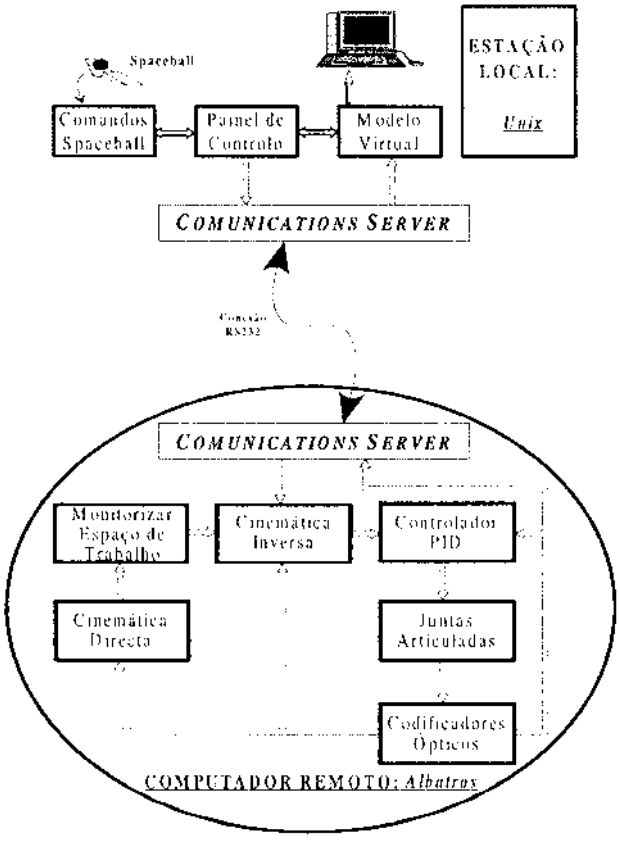


Fig. 2 - Diagrama de blocos do sistema robótico

Os comandos da Spaceball são transformados em coordenadas do braço robótico e aplicados à tarefa de cinemática inversa que corre no sistema computacional multi-tarefa do manipulador. O algoritmo de cinemática inversa calcula os deslocamentos de cada junta articulada que correspondem ao movimento desejado para a extremidade do braço. Estes deslocamentos são os dados de entrada para os seis controladores de PID (um para cada junta articulada), forçando o deslocamento de cada junta articulada para a posição/velocidade desejada. O sinal de realimentação usado ao nível do controlo dos PID é obtido por meio de codificadores ópticos incrementais acoplados aos servo-motores em cada junta articulada. Periodicamente (todos os 250 ms), a estação central inquire o sistema robótico e actualiza as coordenadas posicionais. As ferramentas gráficas usam estas coordenadas para modificarem dinamicamente a representação gráfica 3D do braço manipulador.

III. CINEMÁTICA DO MANIPULADOR ROBÓTICO

A cinemática de braços robóticos trata do estudo analítico da geometria do movimento, em particular, as relações entre as coordenadas no espaço das juntas articuladas $q = [q_1,...,q_n]$ e as coordenadas no espaço operacional $r = [r_1,...,r_m]$. As relações matemáticas que se podem estabelecer entre os dois sistemas, consoante o sentido da transformação de coordenadas, tomam o nome de cinemática directa e inversa. A cinemática directa corresponde a uma transformação não linear do tipo:

$$r = f(q) \tag{3.1}$$

a qual tem sempre uma solução, e conduz à relação diferencial:

$$\dot{r} = \left(\frac{\partial f}{\partial q}\right) \cdot \dot{q} \tag{3.2}$$

Por outro lado, a cinemática inversa consiste no estabelecimento das relações inversas da forma:

$$q = f^{-1}(r) \tag{3.3}$$

Esta equação pode ter ou não solução e, no caso afirmativo, existem ainda duas possibilidades a saber: uma única solução ou uma infinidade de soluções. As relações diferenciais são também mais complexas que no caso anterior e resultam:

$$\dot{q} = \left(\frac{\partial f}{\partial t}\right)^{-1} \cdot \dot{r} \tag{3.4}$$

Pelo facto das últimas três juntas articuladas do robot partilharem a mesma origem, o movimento cartesiano pode ser separado em movimento de translação e rotação, simplificando os cálculos envolvidos [7]. Deste modo, o problema da cinemática é dividido em dois subproblemas de menor dimensão ($n = m = 3$): (1) análise dos três primeiros eixos que constituem o braço e que permitem posicionar a mão no espaço tridimensional; e (2) análise dos três últimos eixos que constituem o punho e permitem orientar a mão [8]. A arquitectura do sistema robótico, ao nível mais baixo, proporciona o controlo independente da posição e velocidade de cada junta articulada. O propósito do controlador de posição é actuar os diferentes motores de modo que o deslocamento angular da junta siga o deslocamento angular desejado. Este tipo de movimento discreto entre dois pontos no espaço cartesiano é conhecido por movimento ponto-a-ponto, e é usado para colocar a ponta do braço (ou *end-effector*) numa posição e orientação arbitrária dentro do espaço de trabalho. Quando as juntas do manipulador são actuadas em velocidade o manipulador é capaz de executar movimentos contínuos ao longo de um dado caminho no espaço cartesiano. Este tipo de controlo, movimento diferencial, é usado para implementar o modo interactivo privilegiado de operação em que o utilizador do sistema, actuando sobre a Spaceball, manipula o robot ao longo do espaço de trabalho. Em qualquer dos casos, as transformações entre o espaço cartesiano e o espaço das juntas consomem uma parte considerável do tempo de processamento. Contudo, o controlo de velocidade exige transformações em tempo real, tornando esta tarefa computacionalmente muito intensiva e dando origem a maiores intervalos de controlo.

A. Redundância

Com base na geometria do braço, podem ser identificadas várias configurações para o manipulador. Para o manipulador tipo PUMA, com seis eixos, existem quatro soluções possíveis para a configuração dos primeiros três eixos (figura 3) e para cada uma destas soluções existem mais duas soluções possíveis para os três últimos eixos (figura 4). Manipuladores com seis graus de liberdade apresentam, frequentemente, redundância visto a tarefa particular a ser realizada requerer menos graus de liberdade do que os disponíveis no manipulador.

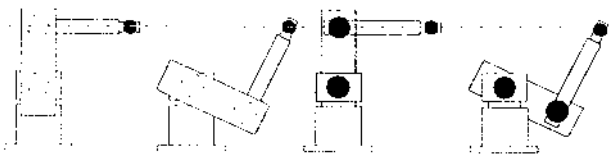


Fig. 3 - Configurações redundantes do braço.

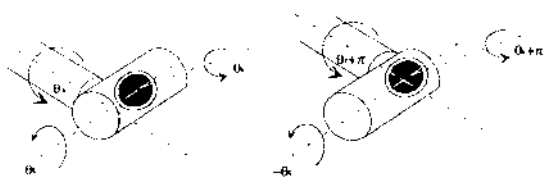


Fig. 4 - Configurações redundantes do pulso.

B. Singularidades

As singularidades cinemáticas ou configurações singulares surgem quando a matriz jacobiana do manipulador $J_r(q)$, que relaciona as velocidades das juntas com a velocidade do *end-effector*, é singular. Nestas configurações, o robot degenera e perde um ou mais graus de liberdade, isto é, existe uma redução no número de linhas e colunas linearmente independentes da matriz. Em termos práticos, o mecanismo robótico não permite velocidades do *end-effector* nas chamadas direcções singulares. Isto embaraça severamente o controlo da trajectória desejada porque velocidades arbitrárias do *end-effector* não podem ser asseguradas. Trajectórias que passem perto de singularidades cinemáticas são também difíceis de efectuar porque o mau condicionamento da matriz jacobiana origina velocidades para as juntas articuladas que não são fisicamente realizáveis (embora de valor limitado).

Para o manipulador em estudo, com pulso esférico, as configurações singulares podem ser classificadas em duas partes: singularidade fronteira e singularidade interior. As singularidades fronteira podem ser evitadas restringindo a área de trabalho. Por sua vez, as singularidades interiores podem ocorrer virtualmente em qualquer lugar dentro do espaço de trabalho quando dois ou mais eixos do robot são colineares. Dada a estrutura cinemática do manipulador, todas as suas configurações singulares podem ser encontradas resolvendo a equação $\det[J_r(q)]=0$ [7]. O manipulador tipo PUMA tem três singularidades (ver figura 5): (1) a singularidade de alinhamento, quando o pulso está tão próximo do eixo da primeira junta quanto é possível; (2) a singularidade de cotovelo, quando este se encontra completamente estendido; e (3) a singularidade de pulso, quando os eixos quatro e seis estão alinhados.

C. Arquitectura do Sistema Robótico

Todas as tarefas relacionadas com o braço manipulador são executadas no computador do robot equipado com um processador 68020. As tarefas são executadas concorrentemente sobre Albatros, um sistema operativo multi-tarefa em tempo real. O sistema operativo implementa um *scheduler* de prioridades com um período de amostragem de 7.5 *ms*. Os processos do sistema são tarefas de baixo nível geridas pelo *kernel* do Albatros, às quais estão reservadas as prioridades superiores. A tarefa

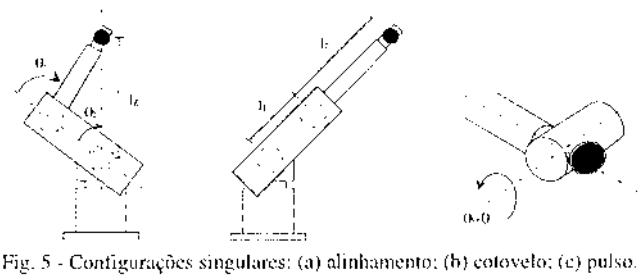


Fig. 5 - Configurações singulares: (a) alinhamento; (b) cotovelo; (c) pulso.

associada aos controladores de PID compreende um gerador de trajectórias de referência e os ciclos individuais de controlo de baixo nível. O gerador de trajectória calcula os pontos intermédios das juntas articuladas de modo a sincronizar os seis eixos. A saída dos controladores de PID comandam os actuadores, forçando o deslocamento de cada junta articulada. A realimentação necessária ao controlo do movimento é obtido por meio de codificadores ópticos incrementais acoplados aos servo-motores em cada eixo individual. Uma placa controladora de eixos (*bus* VME) lê estes dados que permanecem disponíveis para os outros processos sob pedido (e.g., as tarefas de cinemática directa e inversa).

IV. *INTERFACE* HOMEM-MÁQUINA

O operador do sistema está envolvido no controlo e supervisão do sistema através da consola de operação e da correspondente *interface* homem-máquina. A principal característica da *interface* desenvolvida é a integração de toda a informação necessária à operação do sistema num único écran, incluindo a exibição de imagens vídeo, modelos virtuais, e *interfaces* gráficas para o controlo (figura 6). A *interface* homem-máquina foi preparada com o objectivo de proporcionar um controlo supervisionado realista e intuitivo que permita:

- monitorar continuamente a operação via *displays* baseados em vídeo ou ambientes virtuais.
- interactivamente intervir e modificar os objectivos pela aplicação de comandos de controlo enquanto opera a Spaceball ou o rato na *interface* gráfica.

A informação visual de realimentação que permite conduzir e monitorar a operação do sistema é fornecida pelo sistema vídeo e pelo exibição visual do ambiente virtual a três dimensões. Um sistema robótico baseado em ambientes virtuais gerados por computador é uma alternativa a *displays* baseados em vídeo. Neste caso, um sinal de menor largura de banda, que consiste na informação dos ângulos das juntas articuladas, é enviado para o sistema local que constrói a representação virtual. O Modelo Virtual representa o sistema robótico a três dimensões. Os dados de posição necessários para a actualização visual são fornecidos pelo sistema sensorial do manipulador. Estes dados permanecem disponíveis ao nível do Communications Server, sendo inquiridos todos os 250 *ms*. Esta taxa de actualização representa o compromisso entre a taxa de *display* e a sua correspondência física, e o tempo de atraso relacionado com as comunicações (comandos do operador do sistema e dados posicionais de realimentação), assegurando deste modo a efectividade da representação. A velocidade computacional e de interpretação do sistema gráfico determinam esta taxa de modificação [9].

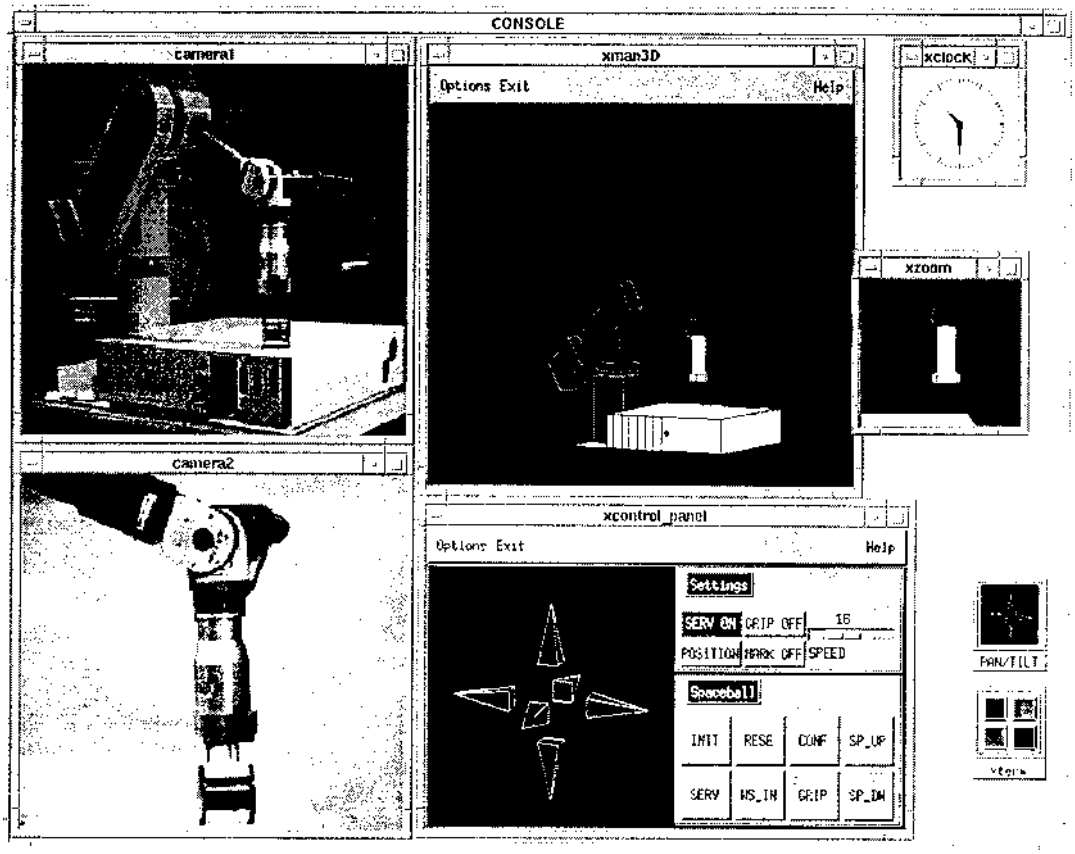


Fig. 6 - Vista integrada da *interface* gráfica do utilizador.

O uso de duas câmaras TV proporciona a realimentação visual para o operador do sistema, complementando a animação 3D em tempo real do braço manipulador. As duas câmaras são montadas em duas unidades de *pan-and-tilt* controladas por computador. Uma das câmaras tem uma lente grande angular permitindo uma vista global do ambiente de operação. A segunda câmara está equipada com lentes motorizadas com *zoom* e focagem automática favorecendo a exibição de detalhes. Uma descrição completa do sistema de visão pode ser encontrada em [10].

O operador interactivamente intervém no controlo do sistema pela aplicação de comandos quer operando a Spaceball quer usando o rato na *interface* Painel de Controlo. O módulo Painel de Controlo é uma janela interactiva que implementa facilidades de controlo para o braço manipulador. Esta *interface* gráfica proporciona os meios para activar as funções de controlo do robot e monitorar o seu estado corrente. Em adição, o painel de controlo funciona como *interface* de entrada dos comandos de movimento gerados pela Spaceball. Foram implementados dois modos de controlo do braço:

- modo *target* - o utilizador interactivamente selecciona a posição e/ou orientação final do *end-effector* (movimento ponto-a-ponto).
- modo interactivo - o operador usa um dispositivo de entrada, Spaceball ou rato, para controlar interactivamente os movimentos do *end-effector* (movimento diferencial).

V. DISCUSSÃO

O trabalho descrito não pode ser considerado completo, sendo antes um primeiro passo no desenvolvimento de um sistema de manipulação para a realização de operações de verificação remota. Nesta perspectiva, os objectivos futuros passam por duas fases importantes: (1) a integração no sistema robótico global (robot móvel e braço manipulador); e (2) o teste e avaliação a efectuar pelos utilizadores finais. O sistema é baseado em conceitos de modularidade e expansibilidade e, por isso, não são esperadas grandes dificuldades para a integração completa [2]. Por outro lado, a reacção dos utilizadores finais é difícil de prever, nomeadamente ao nível da *interface* homem-máquina. Na verdade, o desempenho do sistema depende, principalmente, da sincronização do ciclo de operação, incluindo o operador.

O controlador PID mostra-se adequado para o tipo de operações pretendidas: movimentos suaves e sem restrições de tempo. Contudo, em combinação com o sistema de engrenagens, apresenta uma reduzida velocidade de resposta, originando vibrações desnecessárias e limitando a precisão. Assim, o movimento da extremidade do braço é afectado pelas repetidas acelerações e desacelerações à volta do caminho seleccionado e por pequenas vibrações residuais que produzem oscilações.

Aumentar a potência de cálculo é fundamental para deixar mais tempo computacional para outros algoritmos, como por exemplo, planeamento de trajectórias, detecção de obstáculos para evitar colisões, e etc. O operador humano encontra-se dentro do ciclo de realimentação a um nível muito alto mas com pequena largura de banda, enquanto o ciclo sensorial de baixo nível está perto do robot com grande largura de banda. Assim, o objectivo futuro é usar uma técnica de controlo supervisionado que permita passar cada vez mais autonomia para o robot, sem alterar a estrutura básica e permitindo a intervenção humana em tempo real.

VI. CONCLUSÕES

A arquitectura do sistema manipulador, tanto ao nível do *hardware* como do *software*, é bastante flexível para os tipos de operações envolvidas: movimento suave e sem restrições no tempo de execução. A filosofia de sistema aberto, com ênfase no uso de standards industriais (X11R5, Motif, PHIGS), é uma vantagem para a portabilidade do *software* para outros sistemas. A arquitectura a dois processadores permite uma distribuição flexível das tarefas de acordo com as suas características intrínsecas. Assim, as aplicações baseadas nos sensores ou orientadas para o *hardware* são executadas no computador do robot, enquanto as aplicações do utilizador orientadas para o controlo do sistema são executadas na estação gráfica em ambiente Unix.

Pode dizer-se que a *interface* gráfica é fácil de usar, proporcionando ao operador toda a informação necessária para a tele-operação do braço manipulador. A integração de várias janelas num único écran apresenta vantagens em relação a sistemas clássicos (em geral, incluem um grande número de monitores e consolas de operação), a principal das quais a de estimular a participação do operador dum modo mais activo.

A utilização de modelos virtuais permite, para além da complementaridade de informação, a implementação dum sistema de controlo local, isto é, em vez dos comandos serem transmitidos para o sistema remoto são usados por um modelo local. A informação visual adquirida poderá então ser usada para tarefas de treino de operadores ou como sistema de predição antes de efectuar uma dada tarefa.

REFERÊNCIAS

[1] J.G.M. Gonçalves et. al., "Computer Aided Tele-Operation Applied to Safeguards", Proceedings of the 31st. Annual Meeting of the INMM, pg. 566-571, 1990.

[2] J.G.M. Gonçalves et. al., "Mobile Robotics for the Surveillance of Fissile Materials Storage Areas: Sensors and Data Fusion", no livro *Data Fusion Applications*, editado por S. Pfleger, J.G.M. Gonçalves e D. Vernon, pg. 214-245, Springer-Verlag, 1993.

[3] J.G.M. Gonçalves et. al., "Mobile Robotics Applied to the Remote Verification of Storage Areas", 15th ESARDA Symposium on Safeguards and Nuclear Material Management, pg. 817-827, 1993.

[4] V. Sequeira, J.G.M. Gonçalves, "Communications Server for a Mobile Robotic System", JRC Technical Note, 1993.

[5] *Spaceball: Application Developers' Reference*, Spatial Systems Inc, 1989.

[6] *Albatros Reference Manual*, Robosoft Inc, 1989.

[7] K.S. Fu, R.C. Gonzalez and C.S.G. Lee, *Robotics, Control, Sensing, Vision, and Intelligence*, McGraw-Hill, 1987.

[8] F.M. Silva, "Sistema de Tele-operação dum Manipulador Robótico", Tese de Mestrado, 1995.

[9] F.M. Silva, J.G.M. Gonçalves, "A Graphical Tool for the Tele-operation of a Manipulator Arm", Proc. of COMPUGRAPHICS'92, pg. 346-354, 1992.

[10] V. Sequeira, J.G.M. Gonçalves, "Control and Navigation of a Mobile Robot", JRC Technical Note, 1993.

ICAP - Interface de Comunicação para µAutómato Programável

Fernando J. F. C. de Sousa, Máximo A. S. de Oliveira, José Luis Azevedo, J. P. Estima de Oliveira

***Resumo*- Apresenta-se neste trabalho uma ferramenta de software, desenvolvida para computador pessoal, que substitui a consola de programação de um autómato programável.**
A ferramenta desenvolvida apresenta-se ao utilizador sobre uma interface gráfica em ambiente Windows.

***Abstract*- This work presents a software tool, developed for a personal computer, that can replace the programming console of a programmable logic controller (PLC).**
To the user, the tool appears as a graphic interface under the Windows environment.

1. INTRODUÇÃO

Hoje em dia as mais diversas tarefas tendem a ser controladas e/ou automatizadas, obrigando, em cada caso, ao estudo do processo que se pretende automatizar e ao desenvolvimento de um sistema de controlo adequado. Em muitas situações, os autómatos programáveis, mercê da sua versatilidade, expansibilidade e robustez, são os controladores de eleição.

Os autómatos programáveis são constituídos essencialmente por uma unidade de processamento, ligada a dispositivos de entrada/saída com isolamento galvânico, aos quais é possível ligar sensores, detectores, fins-de-curso, actuadores, etc, [1]. Nas configurações mais simples as entradas e as saídas são puramente digitais.

Um dispositivo ligado ao autómato, capaz de lhe fornecer um sinal, é um dispositivo de entrada. O ponto físico onde esse dispositivo é ligado ao autómato é genericamente designado por *entrada*. A cada entrada está associada uma posição e memória que contém o respectivo estado (ON ou OFF). Esta posição de memória é normalmente apelidada de *input bit*.

Na memória de um autómato existem também *output bits* que guardam os estados das saídas físicas, através das quais são actuados os dispositivos de saída.

Em qualquer modelo de autómato programável está implementado um conjunto de funções básicas. Por um lado existem funções booleanas como AND, OR e NOT, por outro encontram-se temporizadores, contadores e, por vezes, registos de deslocamento. Também é vulgar encontrarem-se intruções aritméticas, de manipulação de bits, bytes ou words, e instruções de controle do programa. Nas máquinas de gamas mais elevadas (que, muitas vezes, incluem entradas e saídas analógicas) encontra-se uma grande diversidade de outras instruções especiais.

Facilmente se conclui, assim, que um autómato pode substituir relés, temporizadores, contadores e outros dispositivos. O facto de ser controlado por "software" e, portanto, também programável garante-lhe uma grande versatilidade, mantendo no entanto a ideia do acesso discreto a cada um dos dispositivos de entrada/saída.

As linguagens de programação suportadas pelos autómatos dependem do modelo escolhido, podendo ir das simples listas de mnemónicas (idiossincráticas de família para família) e/ou diagramas de escada ("ladder logic") [2], até linguagens mais sofisticadas como os diagramas funcionais e o Grafcet [3], ou mesmo C. Hoje em dia, todas as linguagens suportadas pelos fabricantes tendem a obedecer, *grosso modo*, à norma IEC 1131-3 [4].

Para programar o autómato costuma utilizar-se uma consola que permite escrever e transferir programas para o autómato, monitorizar e alterar dados na memória. Nos modelos das gamas média e alta, os fabricantes disponibilizam igualmente ferramentas de programação utilizáveis em computadores possoais.

Quando este trabalho foi desenvolvido, os modelos SP-10, 16 e 20 da Omron eram apenas operáveis a partir de consola própria. É essa consola, que se pretende substituir com o ICAP - Interface de Comunicação para µAutómato Programável. Para isso, estabeleceram-se os seguintes objectivos:

- Caracterizar o protocolo de comunicação entre a consola e o autómato, já que este tipo de informação não é facultado pelo fabricante.

- Caracterizar o formato das diferentes instruções, tarefa necessária pelas mesmas razões do ponto anterior.

- Desenvolver uma ferramenta de "software", que proceda à validação sintáctica e lexical de um ficheiro contendo a sequência de instruções que formam um programa.

- Desenvolver uma ferramenta de "software", que codifique as diferentes instruções de um qualquer programa, em código inteligível pelo autómato.

- Desenvolver a ferramenta que, sobre um ambiente gráfico, permita a edição, validação, armazenamento, envio e recepção de programas para o autómato.

- Dotar a interface gráfica da possibilidade de monitorização e edição das várias zonas de memória.

II. A CONSOLA E O AUTÓMATO

A. Consola

A consola utilizada possui um teclado que permite aceder às instruções do autómato e a funções específicas de monitorização [5]. Tem ainda funções de "upload" e "download", podendo armazenar programas em cartões de memória de 16 Kbytes. Está dotada de um monitor de cristais líquidos onde são visualizáveis as instruções digitadas.

Com a consola é possível seleccionar dois modos de funcionamento: prgm e run. O primeiro permite edição, "upload" e "download" de programas. O segundo faz correr o programa existente na memória do autómato.

B. Autómato

O autómato utilizado foi um SP20 da OMRON [5]. Possui 12 entradas e 8 saídas. Tem um conjunto de 38 instruções e contém memória que permite construir programas até 240 instruções. Está dotado de 2 *analog timers*, 1 *reversible drum counter*, 16 *timers/counters*, 1 *shift register* de 16 bits e as instruções lógicas são igualmente de 16 bits. É programado por intermédio da consola descrita anteriormente.

C. Memória do Autómato

A memória do autómato está distribuída por sete áreas distintas:

12	*input bits* para receber sinais externos.
8	*output bits* para enviar sinais para o exterior.
204	*work bits* que, num programa, podem guardar os diferentes estados.
69	*bits* dedicados a funções específicas.
256	*data retention bits* onde pode ser guardada informação permanentemente (esta zona é alimentada por uma pilha).
128	*link relay*, zona usada para transferências de dados com outros autómatos.
16	*timers/counters* para definir temporizadores e contadores.

III. COMUNICAÇÃO

Dada a ausência de informacão, foi necessário determinar o protocolo usado na comunicação consola-autómato. Concluiu-se que a comunicação se processava nos seguintes termos:

9600 baud - 8 data bits, 1 start, 1 stop, even parity.

Determinou-se também o formato da trama de comunicação:

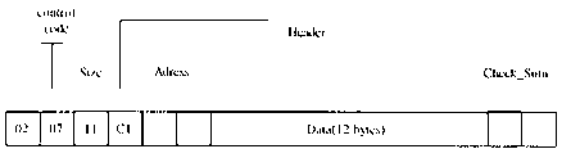


Fig. 1 - Trama de comunicação.

Estrutura da trama

02 início de trama.

Control code:

0F comunicação da consola para o autómato.

07 comunicação do autómato para a consola.

Size: comprimento da trama.

Header:

C1 a informação enviada pela consola destina-se a ser gravada na memória do autómato.

C3 a informação enviada não se destina a ser gravada na memória do autómato.

Address: Endereço destino da informação da trama.

Data: 12 bytes para transporte de informação.

Checksum: checksum da trama.

IV. ROTINAS DE BASE DO PROGRAMA ICAP

O programa ICAP foi desenvolvido de modo a possuir dois níveis distintos: um ao nível do utilizador, que consiste numa interface gráfica gerada em Visual Basic, e outro de mais baixo nível, que consiste em rotinas de comunicação, monitorização, interpretação e geração de código (Fig.3). Estas rotinas de base, que serão apresentadas de seguida, foram desenvolvidas em C++, recorrendo-se ainda ao uso das ferramentas *Lex* e *Yacc* para a construção de um tradutor.

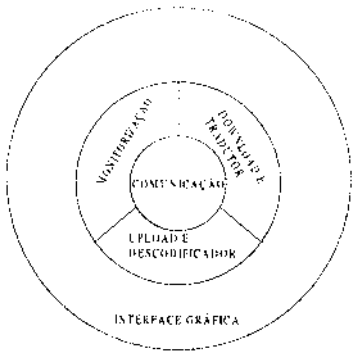


Fig. 2 - Diagrama da estrutura do programa ICAP.

A. Tradutor

Pretende-se transformar um conjunto de mnemónicas organizadas num ficheiro (programa), em código legível pelo autómato. Para tal recorreu-se a dois utilitários destinados a escrever compiladores - *Lex* e *Yacc* [6].

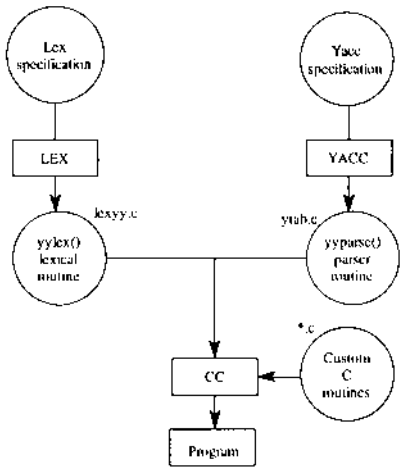


Fig. 3- Construção do tradutor.

Estes, como se ilustra na figura 3, aceitam um ficheiro de especificações e geram, respectivamente, uma rotina lexical-yylex() e uma rotina *parser*-yyparse(), que, quando compiladas com código construído pelo programador, constroem o tradutor desejado.

B. Descodificador

O descodificador lê a informação contida num ficheiro, tenta identificá-la com o código de instruções conhecidas e, caso o faça com sucesso, escreve-a noutro ficheiro.

Quando o código lido não é identificável com uma instrução (situação pouco provável, já que na recepção é verificado o *checksum*), a rotina pára a sua execução e devolve uma mensagem de erro.

Para proceder à interpretação do código armazenado no ficheiro, este é inicialmente carregado na memória do PC e é inicializado um ponteiro (sinalizador de início de instrução) para essa área da memória.

O método desenvolvido para identificar uma instrução (sinalizada pelo ponteiro), consiste numa comparação exaustiva da memória com o formato das várias instruções conhecidas. Como o código das diferentes instruções varia em tamanho e tem *bytes* variáveis, dependendo do tipo de operandos e do respectivo *range*, foi desenvolvida uma função especial para efectuar essa comparação.

Especificações da função:

- número de parâmetros de entrada variável.
- o primeiro parâmetro é uma *string* cujo formato é discriminatório dos parâmetros seguintes.
- formato do primeiro parâmetro:

'*'- qualquer byte (não necessita de mais informação)

'd'- um byte (discriminado nos parâmetros da função)

'[]'- um byte descrito bit a bit, podendo estes serem variáveis 'x' - não necessita de informação complementar (Ex: [01x00xx1]).

Exemplo elucidativo da acção desta função

inf. na memória 4F 01 FF 49 00
↑
ponteiro

func_comparação("*dd[0x001001]*",0x01,0xff);

O primeiro byte (endereçado na memória pelo ponteiro) pode ser qualquer um, já que está especificado com um '*' (não necessita de mais informação); o segundo (ponteiro+1), como está especificado com um 'd', é obrigatoriamente o valor contido no segundo argumento da função(0x01); o mesmo acontece com o terceiro (pos+3) (0xff); o quarto terá o formato especificado entre []; finalmente, o quinto obedece ao procedimento do primeiro.

Se o conteúdo da posição de memória seguinte ao ponteiro, obedecer ao formato descrito no primeiro argumento da função e retantes argumentos, se existirem, esta função retorna 1, se não devolve 0.

A comparação com as diferentes instruções é executada com o seguinte procedimento:

Se(= A) Gera instrução A;
Se não Se(= B) Gera instrução B;
Se não Se(= C) Gera instrução C;
.....

sendo: A,B,C,... os formatos respectivos das várias instruções.
'Gera instrução *x*' função que escreve em ASCII a instrução detectada, e que actualiza o ponteiro (sinalizando o início de nova instrução).

O procedimento descrito é executado até ser encontrada a instrução 'END'(Fig.4).

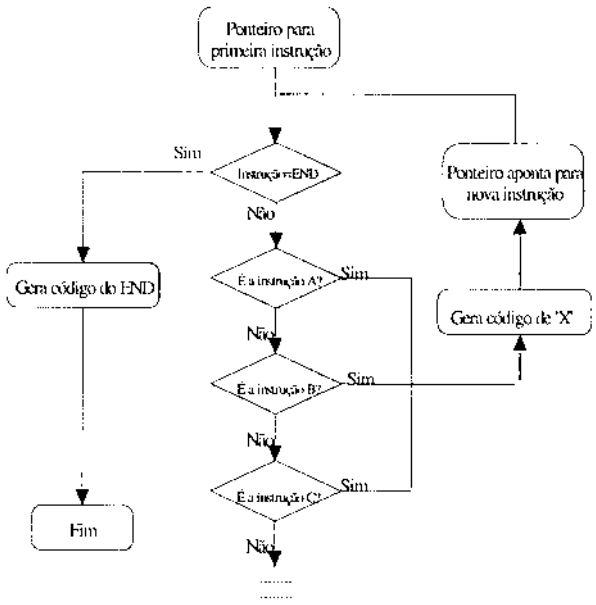


Fig. 4 - Descodificação das instruções.

A rotina 'Geração de instrução' é chamada com um identificador da instrução detectada na memória. Através deste, selecciona um bloco que vai ser o responsável pela transformação para ASCII da instrução e pela actualização do ponteiro para a instrução seguinte. A transformação resultante (ASCII) é escrita num ficheiro. No final, quando a instrução 'END' é detectada, o ficheiro contém o texto de todo o programa.

C. Download

O envio de um programa para o autómato é feito através de 61 tramas: 58 tramas de dados, correspondentes a toda a memória do autómato desde o endereço 0xFB00, até 0xFDB8 (com o incremento de 12 por trama) e 3 tramas finais de controlo.

Inicialmente é construída uma estrutura que vai conter as 61 tramas atrás descritas (Fig.5).

De seguida, as tramas de dados são preenchidas com o código gerado pelo compilador, tendo cada uma capacidade para 12 bytes. O processo repete-se até ser encontrada uma instrução de *end*.

No final, na terceira trama de controlo, é escrito o *checksum* (2 bytes) de todos os dados transportados pelas tramas de *data*.

Resta agora o envio de cada uma das tramas para o autómato. Começa-se por inicializar a porta série para uma comunicação com *baudrate* de 9600, 1 bit de paridade (par), 1 *start bit* e 1 *stop bit*.

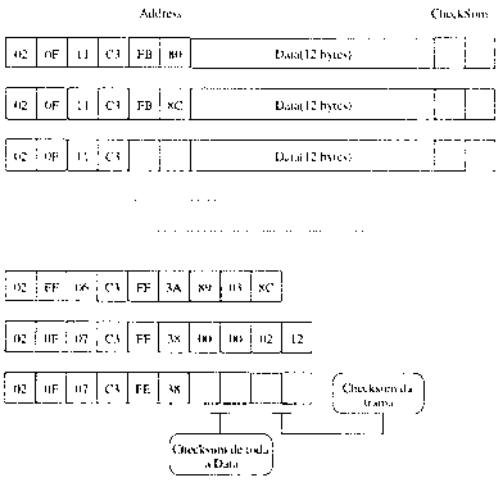
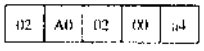


Fig. 5 - Estrutura de tramas do programa do autómato.

Em seguida, são enviadas as tramas previamente preenchidas com os dados do programa, intercalando entre cada uma delas uma trama de sincronismo:



Entre cada byte enviado, é imposto um *delay* de 10ms. Este valor foi encontrado experimentalmente, como sendo óptimo para o autómato receber o programa sem erros.

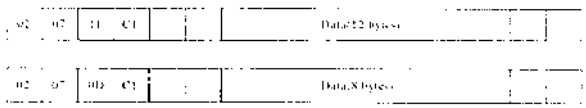
Após o envio de cada trama, aguarda-se a resposta do autómato durante 500 ms. Esgotado este período de tempo é de novo enviada a trama, repetindo-se este procedimento até à sexta tentativa. Falhadas todas as tentativas, é devolvida uma mensagem de erro e a execução do programa termina.

O fluxograma da Fig.6 exemplifica este procedimento.

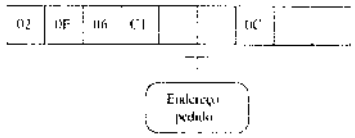
D. Upload

Estando disponíveis as ferramentas de edição, compilação e transferência de um programa para o autómato, surge a necessidade de receber no PC um programa vindo do autómato. Nesta secção vamos mostrar o procedimento para ler um programa contido na memória do autómato.

A recepção de um programa no PC, é feita através de 58 tramas de dados.

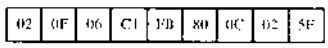


Para obter uma determinada trama é feito o pedido correspondente.

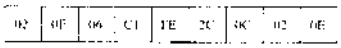


Ao envio de uma trama de pedido, o autómato responde com a trama de *data* correspondente ao endereço pedido.

Assim, fazendo pedidos desde o endereço 0xFB00



até ao endereço 0xFE2C



e com incremento de endereço 12, obtemos toda a área de memória de programa.

Tal como no caso da transmissão, também aqui é devolvida uma mensagem de erro, se se esgotarem as 6 tentativas de recepção de uma trama (Fig.7).

A cada trama recebida, é extraída a informação e escrita num ficheiro. Terminada com sucesso a execução do programa de recepção, obtém-se um ficheiro que contém o código do programa residente na memória do autómato.

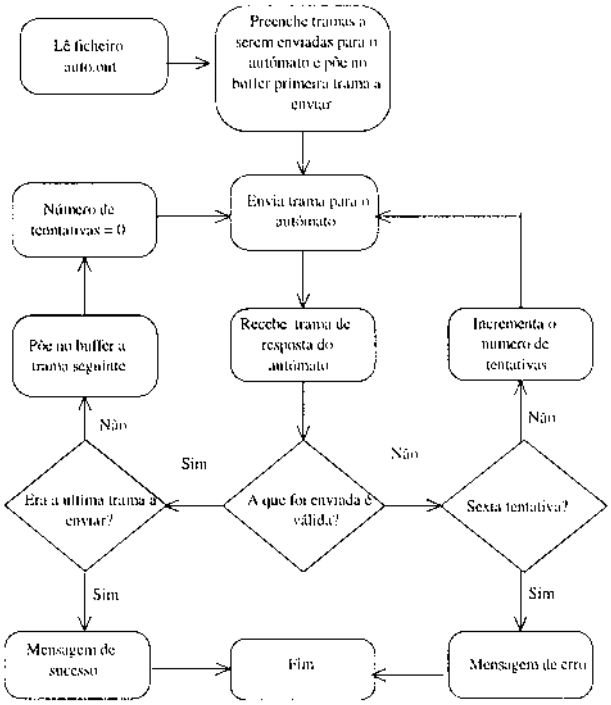


Fig. 6 - Envio de um programa para o autómato.

V. MONITORIZAÇÃO

A monitorização do autómato tem como objectivo conhecer o seu estado interno como, por exemplo, saber quais os *timers/counters* usados, qual o conteúdo de certas zonas da memória, ou qual o estado das entradas e das saídas.

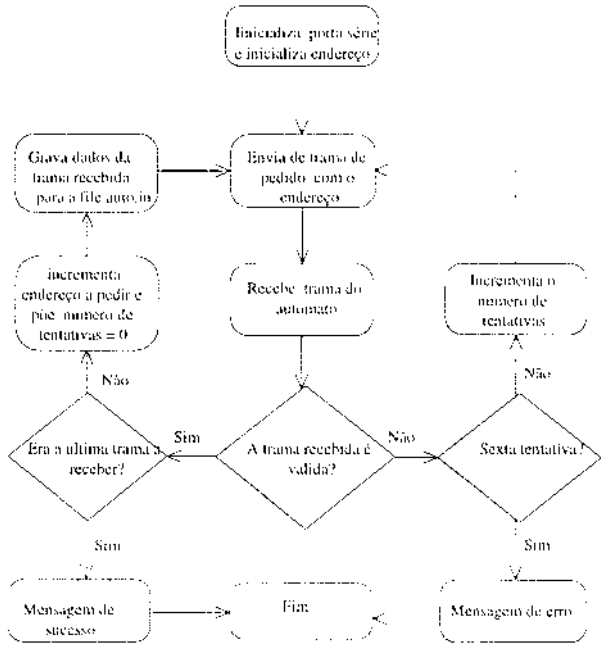
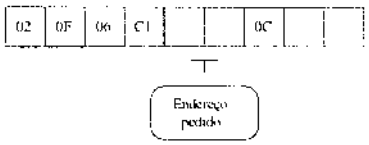


Fig. 7 - Recepção de um programa do autómato.

Toda esta informação pode ser obtida enviando para o autómato "tramas-pedido" adequadas. Essas tramas indicam o endereço da memória do autómato sobre a qual se pretende obter informação.

Por exemplo, para se receber o estado da área de memória DR são enviadas para o autómato tramas cujo endereço varia desde 0xFF40 até 0xFF58, com incremento de 8:



Por cada trama pedido enviada, o autómato devolve o conteúdo do endereço correspondente, em tramas do tipo:

VI. INTERFACE GRÁFICA

O ambiente gráfico foi desenvolvido com base no Microsoft Visual basic 3.0, com o intuito de fornecer ao utilizador um *feedeback* fiável e preciso sobre a tarefa em execução.

Possui funcionalidades de edição, monitorização e invocação de tarefas necessárias ao sistema.

Num primeiro contacto, a interface apresenta-se ao utilizador conforme a Fig.8, onde se destaca o editor que possibilita a edição e visualização de programas, possuindo facilidades de *copy & paste*, informação sobre o número da linha em edição e nome do ficheiro em uso.

Está disponível uma barra de botões associada ao editor, possibilitando a edição rápida de várias funcionalidades. A identificação das mesmas surge momentaneamente no ecrã, quando o rato passa por cima do botão respectivo. Quando um dos botões é premido surge, no editor, o texto correspondente à instrução.

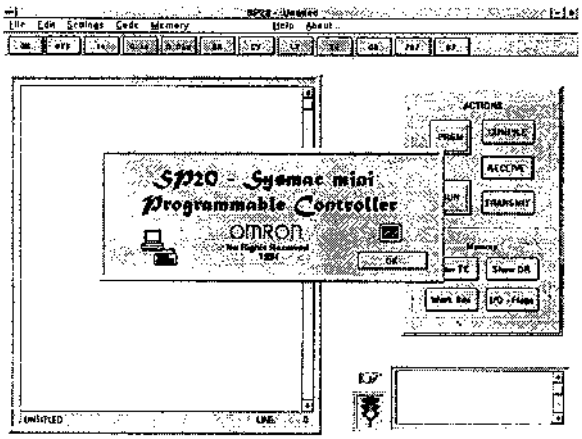


Fig. 8 - Aspecto da interface de entrada

A. Monitorização

A monitorização do estado interno do autómato é feito através de janelas, as quias permitem visualizar o conteúdo de áreas específicas da memória do autómato.

A invocação das janelas respectivas é feita, como atrás se descreveu, usando o painel de comandos, os menus ou combinações de teclas.

B. Temporizadores/Contadores (TC)

A janela apresentada na figura 9 permite ao utilizador monitorizar o estado de todos os TC's (*timers* e *counters*). O botão LOAD executa uma rotina que lê do autómato a informação necessária e apresenta-a ao utilizador numa forma clara.

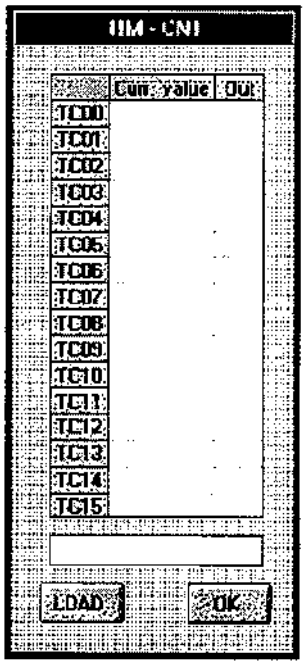


Fig. 9 - Monitorização dos *timers/counters*

Esta janela mostra uma grelha onde são assinalados, através de um símbolo (neste caso, uma lâmpada) na coluna *out*, todos os TC's que estão a ser usados pelo programa a correr no autómato. Se um dado TC está, num dado momento, activo, a lâmpada apresenta a cor amarela, e vermelha no caso contrário. A coluna *Curr. value* apresenta o valor actual de cada TC.

C. Memória DR

A importância desta área de memória foi já referida. Além das facilidades de monitorização foram também criadas facilidades de edição, com as opções de envio para o autómato ou de gravação em ficheiro.

A janela, apresentada na figura 10, ilustra uma grelha que permite a visualização dos 256 bits da área DR, organizada em palavras de 16 bits. A grelha possui ainda uma coluna onde é escrito, em hexadecimal, o valor de cada palavra.

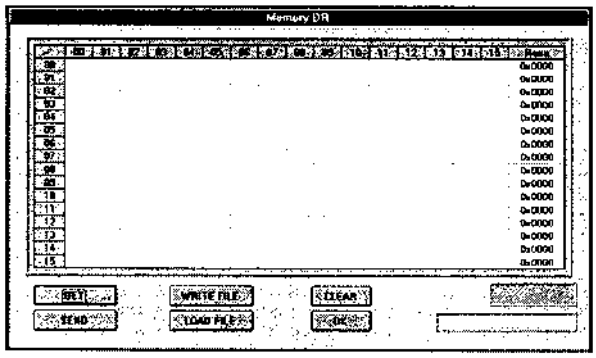


Fig. 10 - Monitorização/Edição da área de memória DR.

A edição é feita directamente sobre a grelha com um *double click* do rato, o que provoca o *toggle* do conteúdo da célula seleccionada. A coordenada do bit, correspondente a essa célula é mostrada junto ao canto inferior direito da grelha. Os botões existentes são:

Este botão acciona a rotina que lê e mostra a área de memória DR do autómato.

Este botão invoca o programa que transfere o conteúdo da grelha para a memória do autómato.

Permite gravar num ficheiro a informação apresentada na grelha.

Lê um ficheiro pré-gravado com um mapa de memória DR e apresenta-o na grelha.

Coloca a zero todas as células da grelha.

Abandona a janela.

D. Workbits

Nesta janela (Fig.11) são apresentadas grelhas que permitem, através do botão LOAD, visualizar o conteúdo dos diferentes *workbits*.

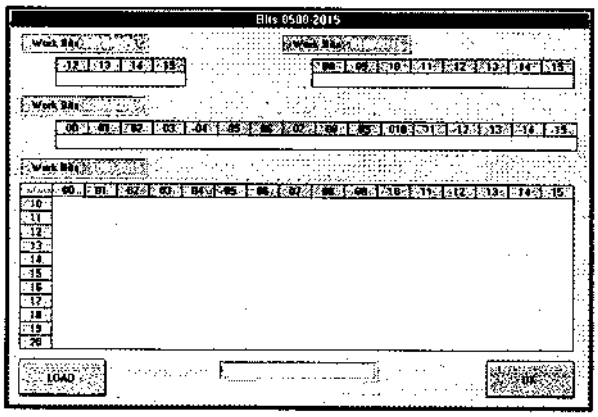


Fig. 11 - Monitorização dos *Workbits*.

E. I/O + Flags

As entradas, as saídas e as *flags* do autómato são monitorizadas nesta janela. Quando o utilizador acciona o botão LOAD, são lidos do autómato os estados referentes a cada uma delas.

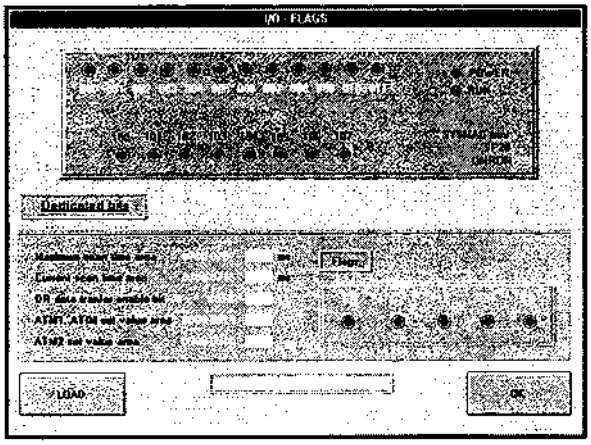


Fig. 12 - Monitorização de I/O e das *flags*.

Nesta janela é representado simbolicamente o autómato com as suas entradas, saídas e *flags*. Quando qualquer uma delas está ON é visualizada como se de um *led* vermelho (aceso) se tratasse.

Como se pode verificar, esta janela também apresenta algumas outras informações úteis.

REFERÊNCIAS

[1] Michel, Gilles, "Programmable Logic Controllers - Arquitecture and Applications", John Wiley & Sons, Chichester, 1990.

[2] Warnock, Ian G., "Programmable Controllers - Operation and Application", Prentice Hall, Hertfordshire, 1988.

[3] David, R. and H.Alla, "PetriNets and Grafcet: Tools for Modelling Discrete Event Systems", Prentice Hall, London, 1992.

[4] IEC 1132-3, International Standards, Programmable Controllers - part 3: programming languages, IEC, 1993.

[5] Omron, SP10/SP16/SP20 Operation Manual, Omron Corporation, Tokyo, 1992.

[6] Mason, Tony and Doug Brown, "Lex & Yacc", O'Reilly & Associates Inc., Sebastopol, 1990.

Projecto de Redes de Adaptação para Amplificadores de Banda Larga*

Raquel Castro Madureira, Nuno Borges de Carvalho, José Carlos Pedro

***Resumo*- Este trabalho visa a descrição de um método de projecto de malhas de adaptação para amplificadores de banda larga, que conduzam a um ganho de transdução constante numa banda previamente definida.**

I. INTRODUÇÃO

Considere-se o circuito da Fig.1:

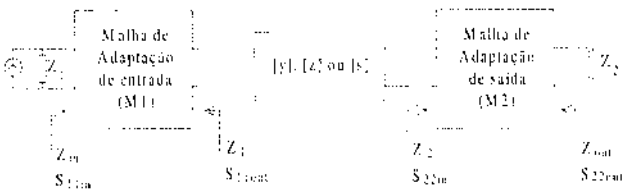


Fig. 1 - Circuito com malhas de adaptação.

Para se conseguir um ganho constante numa largura de banda pré-definida, as malhas de adaptação de entrada e saída deverão adaptar a fonte à entrada do transístor e a saída à sua carga. O ganho do circuito da fonte para a carga, Z_2, vale:

$$G = \frac{P\rho}{Pavs} = \frac{Pil}{Pavs} \cdot \frac{Pavo}{Pil} \cdot \frac{P\rho}{Pavo} = G_1 \cdot G_2 \cdot G_3;$$

Pρ - Potência entregue à carga;
Pavs - Potência disponível na fonte;
Pil - Potência de entrada do dispositivo activo;
Pavo - Potência disponível à saída do dispositivo activo;
G_1 - Ganho da malha M1;
G_2 - Ganho do dispositivo activo;
G_3 - Ganho da malha M2.

G_2 é definido como a razão entre a potência disponível à saída do dispositivo activo, e a potência fornecida na sua entrada, sendo dado por:

$$G2 = \frac{Pavo}{Pil} = \frac{Pavo}{Pavs} \cdot \frac{Pavs}{Pil} = G_A \cdot D_{IN}, \quad \text{com:}$$

G_A - Ganho disponível do transistor;
D_{IN} - Perdas por desadaptação à entrada.

Este ganho pode ser determinado utilizando os parâmetros Y do transístor que se encontram nos *data-sheets* deste.

Como a malha M1 será passiva, terá um ganho $G_1 < 1$ e as suas perdas por desadaptação serão contabilizadas por $IL_1 = \frac{1}{G_1}$; de igual modo $G_3 \leq 1 \Rightarrow IL_3 = \frac{1}{G_3}$.

O objectivo é portanto que IL_1 (*Insertion Loss*) e IL_3 tenham uma função transferência numa banda de frequências que levem a um ganho total constante ou seja que $G_1.G_2.G_3=K$, constante com a frequência.

Os passos necessários para fazer estas adaptações serão descritos a seguir: ***A***. Representação das impedâncias de entrada e saída do dispositivo activo por modelos de circuito equivalente; ***B***. Desenho e modelação da resposta em frequência das várias malhas de adaptação; ***C***. Absorção de elementos parasitas; ***D***. Síntese da malha; ***E***. Escalamento final e ***F***. Verificação de resultados.

II. DESCRIÇÃO DO MÉTODO DE PROJECTO

A. Modelação das impedâncias de entrada e saída do dispositivo activo.

Depois de escolhido o dispositivo activo, deve modelar-se substituindo o quadripolo por um modelo de circuito equivalente. Utilizam-se para o efeito os seus parâmetros Y, Z ou S, sendo preferível estes últimos pois relacionam-se melhor com o ganho em potência e são mais fáceis de medir a altas frequências.

Estando o dispositivo modelado desenha-se a sua curva de ganho na zona pretendida.

B. Desenho e modelação da resposta em frequência das várias malhas de adaptação:

B1. Dispositivos activos com ganho constante na frequência:

Conhecendo a curva de ganho do dispositivo, pretende-se desenhar as curvas de resposta das malhas que melhor adaptam o circuito. De entre as várias possibilidades, pode escolher-se entre uma aproximação do tipo Butterworth, Fig.2 ou Chebychev, Fig.3, de acordo com o que melhor se aproxima do desejado.

*Trabalho realizado no âmbito da componente prática da disciplina de opção de Electrónica de Rádio Frequência.

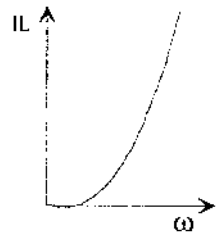


Fig.2 - Função de Butterworth

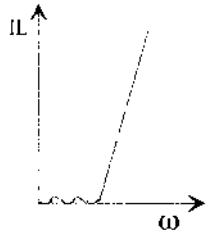


Fig.3 - Função de Chebyshev

As suas funções de transferência são respectivamente, Butterworth:

$$IL = Ko + Kco \cdot (\frac{\omega}{\omega_{co}})^{2N} ;$$

Ko - Perdas por inserção em DC;
Ko+Kco - Perdas por inserção em Wco;
ωco - Frequência de corte.

Chebyshev:

$$IL = Ko + Kco \cdot \left[C_N\left(\frac{\omega}{\omega_{co}}\right)\right]^2 ;$$

$C_N\left(\frac{\omega}{\omega_{co}}\right)$ - Polinómio de Chebyshev;

Ko - Limite do mínimo valor de *ripple*;
Ko+Kco - Limite do máximo valor de *ripple*;
ωco - Frequência de corte.

Os parâmetros destas funções calculam-se como se se tratasse de Filtros Passa Baixo (LP), tendo em atenção as especificações desejadas. De seguida, passa-se para Passa Banda (BP) como pretendido, ou alternativamente continua-se em Passa Baixo, efectuando a transformação no escalamento final. A transformação de Passa Baixo para Passa Banda segue as seguintes expressões:

$$\omega_{LP} \Leftrightarrow Ks \cdot \left(\frac{\omega_{BP}}{\omega_o} - \frac{\omega_o}{\omega_{BP}} \right);$$

$$\omega_o = \sqrt{\omega_L \cdot \omega_u} \; ; \; Ks = \frac{1}{\left(\frac{\omega_u}{\omega_o} - \frac{\omega_o}{\omega_u} \right)};$$

onde : ω_L e ω_U são as frequências inferior e superior da banda de passagem, e ω_{BP} e ω_{LP} são as variáveis de frequência Passa Banda e Passa Baixo, respectivamente.

B2. *Dispositivos activos com ganho variável na frequência:*

Visto o ganho em potência (G_2) dos dispositivos activos variar por vezes com a frequência na banda de interesse, torna-se necessário calcular malhas de adaptação de função de transferência complementar da do dispositivo activo.

Se o ganho do dispositivo activo for do tipo do da figura:

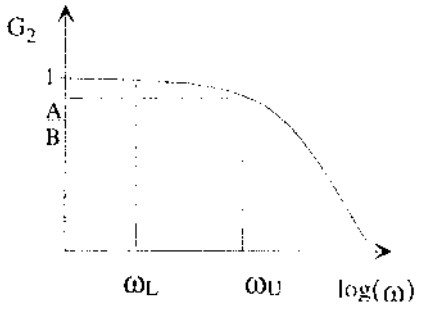


Fig.4 - Gráfico do ganho do dispositivo activo.

Então a malha de adaptação de entrada ou saída deverá ter uma função de transferência do tipo:

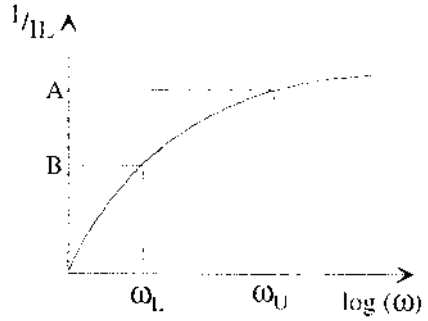


Fig5 - Gráfico da função de transferência requerida da malha de adaptação.

Para se conseguir esse efeito, basta dividir IL (calculado para ganho constante em ω_L) por ω^{2S}, onde S é a inclinação pretendida. Esta aproximação só é válida para S inteiro. Se S não for inteiro dever-se-á aproximar a inclinação por uma combinação linear de duas inclinações inteiras, uma com inclinação ligeiramente menor e outra com maior do que a pretendida, de modo a ter $IL_{pret} = A_1 \cdot IL_1 + A_2 \cdot IL_2$. Os pesos A_i podem ser determinados por:

$$A_1 = \frac{\omega^{2(S_2 - S)} - 1}{\omega^2 - 1}; \qquad A_2 = 1 - A_1 ,$$

em que: A_1 e A_2 são os pesos das componentes de inclinação inferior e superior; e S, S_2 as inclinações pretendida e superior respectivamente.

Utilizando este método existirá sempre uma pequena distorção, visto raramente se conseguir uma aproximação perfeita da inclinação pretendida. Pode provar-se que essa distorção vale:

$$Dist = 10\log_{10} \frac{A_1 \cdot \omega^{-2S_1} + A_2 \cdot \omega^{-2S_2}}{\omega^{-2S}} \text{, sendo } S_1 \text{ a}$$

inclinação inferior e a frequência de distorção máxima:

$\omega_{MD}^2 = \frac{A_2(S_2 - S)}{A_1(S - S_1)}$. Como estas alterações levam a perdas por desadaptação não nulas, convém entrar com essas perdas no dimensionamento da malha. Para isso determina-se a frequência para a qual o mínimo de perdas ocorre, fazendo $\frac{\partial IL}{\partial \omega} = 0$ obtendo $\omega=\omega_{MIL}$. Calculando $IL(\omega_{MIL})$ tem-se o mínimo de perdas, MILo. De seguida multiplica-se IL por uma constante $K = 10^{\frac{MIL_{des} - MIL_0}{10}}$; com MIL_{des} e MIL_0, as mínimas perdas desejadas, e introduzidas pelo circuito em dB;

Assim, IL vale finalmente:

$IL = \left(A_1 \cdot IL(S_1) + A_2 \cdot IL(S_2)\right) \cdot K$, sendo K o valor de perdas existentes, visto a curva não ter a inclinação pretendida. Deste modo consegue-se um ganho constante na banda desejada.

C. Absorção de elementos parasitas:

Elementos parasitas são condensadores ou bobinas, em série ou paralelo, necessários ao modelo do circuito equivalente do dispositivo a adaptar.

A malha de adaptação, originalmente sintetizada supondo terminações puramente resistivas, deve ter em conta a inclusão de possíveis elementos parasitas, o que se consegue com uma adequada escolha da topologia. Este procedimento denomina-se absorção dos elementos parasitas. Por exemplo, a absorção de um condensador em paralelo existente do modelo do porto a adaptar, faz-se impondo como condição à topologia da malha que esta inclua, à saída, uma capacidade paralelo como elemento terminal, e de valor igual ou superior ao do condensador parasita a absorver.

Normalmente as malhas não conseguem absorver completamente os elementos parasitas existindo sempre uma diminuição de ganho em relação ao ideal pretendido. Uma previsão teórica deste facto consegue-se usando as relações de FANO, que nos dão uma ideia do elemento parasita que é possível absorver:

$$\int_0^\infty \ln\left|\frac{1}{S_{11}}\right| d\omega \le \pi \cdot \left(\frac{1}{R_1 \cdot C} - \sum Z_{S11rhp}\right),$$

condensador parasita em paralelo ou bobina parasita série, (caso 1);

$$\int_0^\infty \omega^{-2} \cdot \ln\left|\frac{1}{S_{11}}\right| d\omega \le \pi \cdot \left(\frac{L}{R_1} - \sum \frac{1}{Z_{S11rhp}}\right),$$

bobina parasita em paralelo ou condensador parasita série, (caso 2).

S_{11} é o coeficiente de reflexão à entrada da malha de adaptação que, como se verá à frente, se relaciona com o ganho do dispositivo, e Z_{S11rhp} são os zeros de S_{11} no lado direito do plano S.

Como se pode ver da relação de FANO, a capacidade de uma malha absorver um elemento parasita, mantendo determinado ganho, é tanto menor quanto maior for o numero de zeros que S_{ii} tiver do lado direito. Logo, os zeros de S_{ii} deverão estar do lado esquerdo do plano S (a que estão associados S_{11} e S_{22}).

No entanto, como a malha tem dois portos, tem de absorver elementos parasitas de um lado e do outro, ou seja, do lado do dispositivo activo, e do da carga ou excitação. Assim, quando se pretender absorver elementos parasitas dos dois lados da malha vai ter de existir um compromisso entre os zeros de S_{11} e S_{22} do lado direito, que se colocam em cada S_{ii}. Esse compromisso tem a ver com o porto que se pretenda mais imune a elementos parasitas.

Considerando $\tau_{LP}=\omega_u$.C.R, (caso 1) ou $\tau_{LP}=\omega_u$L/R, (caso 2) as relações de FANO simplificam-se, resultando em :

$$\int_0^\infty \ln\left|\frac{1}{S_{11}}\right| d\Omega \le \frac{\pi}{\tau_{LP}}, \text{ com } \Omega = \frac{\omega}{\omega_u},$$

para condensador em paralelo e bobina em série e

$$\int_0^\infty \Omega^{-2} \cdot \ln\left|\frac{1}{S_{11}}\right| d\Omega \le \pi \cdot \tau_{HP}$$

para condensador em série e bobina em paralelo. Destas expressões extraíram-se vários gráficos[1] que dão a redução do ganho conforme τ e a largura de banda, de modo a determinar-se a redução de ganho necessária à extracção de certos elementos parasitas.

Obtido o valor da redução do ganho, basta multiplicar IL (calculado) por esse ganho, e teremos a expressão da função de transferência da malha pretendida. Todos estes cálculos dão apenas uma estimativa, pois são baseados numa resposta ideal, daí que a redução do ganho deva, na prática, ter sempre uma margem de segurança.

D. Síntese da malha:

Se a malha for passiva e não dissipativa, prova-se que[1]:

$1 - |S_{11}(\omega)|^2 = 1 - |S_{22}(\omega)|^2 = \frac{1}{IL}$.

Considerando a Fig.6:

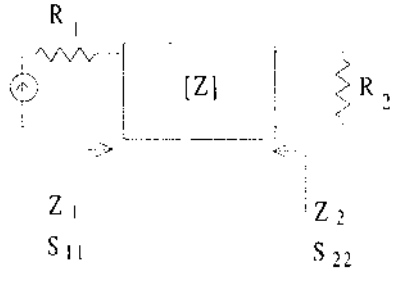


Fig.6 - Malha de adaptação.

com $S_{11}=\frac{Z_1-R_1}{Z_1+R_1}$; $S_{22}=\frac{Z_2-R_2}{Z_2+R_2}$, pode ainda provar-se que os zeros de $S_{11}(\omega)$são iguais aos zeros de $S_{22}(\omega)$, mas em oposição de fase[1].

Considerando as condições acima, a síntese de malhas de adaptação fica facilitada uma vez que:

- A função IL já foi previamente calculada.
- Através de $|S_{11}(\omega)|^2 = 1-\frac{1}{IL}$ calcula-se $|S_{11}(\omega)|^2$.
- Como IL está definido em ω, faz-se a transformação ω=s/j, para obter $|S_{11}(s)|^2$
- Factorizando $|S_{11}(s)|^2 = S_{11}(s)\cdot S_{11}^*(s) = S_{11}(s)\cdot S_{11}(-s)$ obtém-se o numerador e denominador e, consequentemente, os pólos e zeros.
- Seleccionam-se os pólos que estiverem do lado esquerdo do plano s para pólos de $S_{11}(s)$ visto conduzirem a uma solução estável. Dos zeros de $|S_{11}(s)|^2$ em pares complexos conjugados, escolhem-se só os do plano direito ou esquerdo, o que conduz a topologias diferentes.

Escolha de topologias:

Como $IL=\frac{a_0+a_2\cdot\omega^2+\ldots+a_n\cdot\omega^{2N}}{\omega^{2J}}$, com:

N-ordem do filtro;
J-número de zeros de transmissão em DC;
(N-J)-número de zeros de transmissão no infinito.
Então a topologia escolhida terá de satisfazer :

i) A malha deverá ter J elementos do tipo :

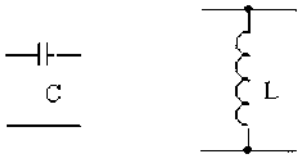


Fig. 7 - Elementos com zero de transmissão em DC.

visto ambos levarem a zeros em DC, e (N-J) elementos do tipo:

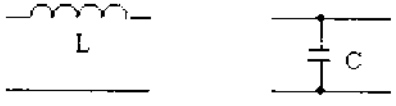


Fig. 8 - Elementos com zero de transmissão no infinito

que sintetizem zeros no infinito.

ii) A topologia deverá ter as impedâncias de entrada e saída que se comportem em DC e infinito de acordo com as especificações. Isto é o chamado comportamento assimptótico. Tomemos como exemplo o seguinte coeficiente de reflexão:

$$S_{ii}=\frac{k\cdot(s+a+jb)\cdot(s+a-jb)\cdot(s+c)}{(s+d)\cdot(s+e)\cdot(s+f)};\ k=\pm 1.$$

Como $Z_1=\frac{1+S_{ii}}{1-S_{ii}}$, então o sinal de S_{11} em DC e no infinito vai levar à definição do comportamento assimptótico da malha. Como os pares de zeros complexos conjugados não têm influência no sinal de S_{ii} e o denominador é polinómio de Hurwitz[1] (e portanto também não influencia o sinal), são os zeros no eixo real que determinam o sinal da função e, consequentemente, o seu comportamento assimptótico. Se S11(DC)=-1 => Z1(DC)=curto circuito, se S11(∞)=1 => Z1(∞)=circuito aberto. Cada opção conduz, portanto, a uma topologia distinta.

iii) Uma malha de adaptação deverá ter um mínimo número de componentes, pelo que são de evitar cascatas de bobinas ou condensadores.

- Construir S11 a partir das suas singularidades:

$$S_{11}=\frac{k\cdot(s-z_1)\cdot(s-z_2)\cdots}{(s-p_1)\cdot(s-p_2)\cdots},\ k=\pm 1$$ que, como já se demonstrou, determina a topologia a ser utilizada.

- Construir $Z_1(s)=\frac{1+S_{ii}(s)}{1-S_{ii}(s)}$.
- Determinar os elementos da malha por construção de uma expansão em fracções parciais à volta de zero e do infinito[3].

E. Escalamento final

O objectivo seguinte será fazer um 'escalamento' de modo a passar o circuito para a frequência e impedância pretendidas, de acordo com certas fórmulas de transformação[1].

Visto a resistência de carga obtida por este método ser sempre imprevisível, torna-se ainda necessário fazer transformações de impedância de modo a atingir o especificado.

Estas conversões do nível de impedância terminal realizam-se com certas redes denominadas transformadores de impedância[4].

[1]Polinómios de Hurwitz são aqueles que têm todos os seus zeros no lado esquerdo do plano s.

F. Verificação de resultados.

Utilizando programas de CAD verifica-se se as especificações iniciais são ou não respeitadas. Se sim então a malha está construída. Se não, convém utilizar outra topologia. Se mesmo assim não se conseguir, então refaz-se o problema desde a alínea A., de modo a tomar outro tipo de decisões nas várias partes da realização.

III.EXEMPLO

Por forma a clarificar melhor o uso do método acima exposto, apresenta-se de seguida um exemplo académico.

Pretende-se um amplificador com um ganho plano entre 6.5 e 13GHz. Escolheu-se como elemento activo um GaAs FET, cujos parâmetros S são conhecidos.

A. Modelação do transistor.

Usando os parâmetros S obteve-se o seguinte modelo simplificado do dispositivo:

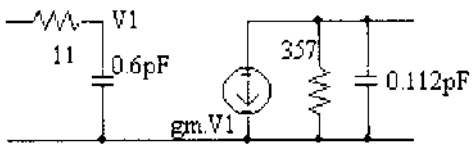


Fig.9 - Modelo do dispositivo a adaptar.

B. Desenho e modelação das malhas.

O objectivo será:

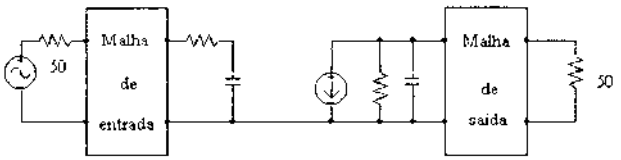


Fig.10 - Malhas de adaptação.

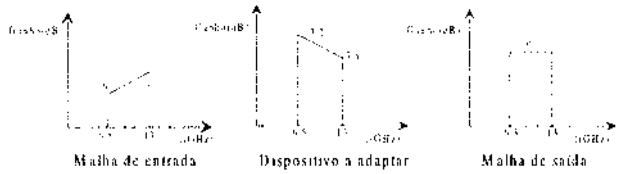


Fig.11 - Ganho dos três andares.

Por forma a reduzir ao máximo a consideração de detalhes não essenciais, resolveu-se atribuir toda a compensação de ganho à malha de entrada e apresentar somente a síntese da de saída. Esta está representada na figura 12.

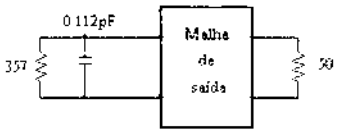


Fig.12 - Malha de saída.

C. Absorção de elementos parasitas.

A máxima transferência de potência na presença de elementos parasitas numa banda finita não é possível, tendo, por isso de se ajustar as especificações para menor ganho, ou permitir um certo ripple na banda.

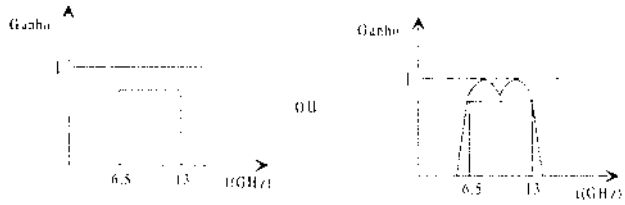


Fig13. - Absorção dos elementos parasitas.

Utilizando as relações de FANO têm-se que:

$\tau_{LP} = W_0 \cdot R_0 \cdot C_0 = 3.26$, e utilizando os gráficos na referência [1] tira-se que MIL=0.2dB. Este valor dá uma estimativa da redução do ganho, ou *ripple* necessária. A seguir verifica-se se as malhas conseguem incluir o elemento parasita. No caso afirmativo, ressintetiza-se a malha até este ponto, de modo a obter 0 dB de MIL e 0,7dB de *ripple*.

D. Síntese da malha de saída.

Na prática várias topologias podem ser utilizadas. Escolheu-se para este circuito a seguinte:

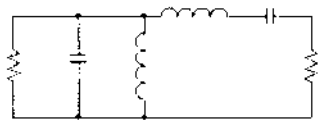


Fig.14 - Topologia escolhida.

Começa-se por desenhar uma topologia passa baixo:

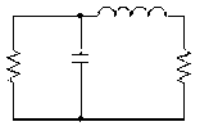


Fig.15 - Topologia passa baixo.

A especificação da resposta passa baixo será:

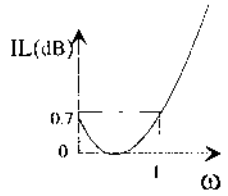


Fig.16 - Resposta do passa baixo.

A expressão matemática para a curva acima é:

$$IL = 1 + 10^{0.07}\left(2\omega^2 - 1\right)^2,$$

$$IL = 1.175 - 0.7\omega^2 + 0.7\omega^4.$$

Desta expressão retira-se o valor de S_{11}. Utilizando, por exemplo, o método de Cauher[3] extraem-se os componentes respectivos para a implementação do circuito.

Como esta expressão é a representação de um passa-baixo e o objectivo é um passa banda, duas hipóteses surgem para a sua realização. Uma é substituir ω na expressão pelas transformações na frequência e impedâncias respectivas, a outra, que será utilizada neste exemplo, consiste em desenhar uma topologia em passa baixo, e converter os condensadores e bobinas para passa banda.

A topologia passa baixo apresenta os valores:

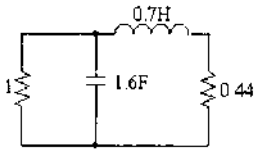


Fig.17 - Implementação do passa baixo.

Utilizando as transformações para passa banda têm-se:

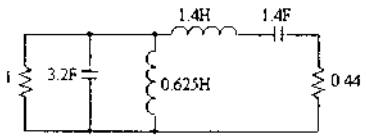


Fig.18 - Topologia passa banda normalizada.

Desnormalizando para a frequência pretendida, 13GHz, e impedância de saída do dispositivo (357Ω) têm-se:

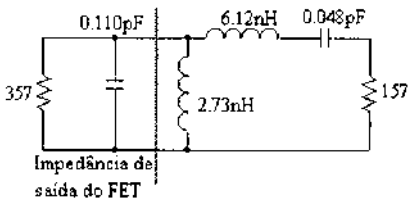


Fig.19 - Topologia desnormalizada.

Como o objectivo é adaptar uma carga de 50Ω, transforma-se a topologia de duas bobinas em "L" para três em "T"[4], por forma a conseguir uma transformação de impedâncias de razão:

$n^2 = \frac{50}{157} = 0.318 \Rightarrow n = 0.564$, o que resulta na malha final:

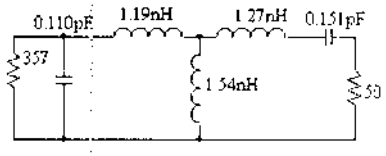


Fig.20 - Malha final.

IV. CONCLUSÕES

Como se pode observar, a realização de malhas de adaptação não é mais do que sintetizar filtros que obedecem a certas especificações de função transferência e impedâncias terminais numa determinada banda. Já existem livros[2], do tipo 'livro de receitas', que permitem construir a maior parte dos filtros por consulta de tabelas. É, no entanto, igualmente possível implementar todo este procedimento por computador, para o que a referência [1] apresenta a maioria dos dados.

Finalmente, espera-se ter transmitido a ideia que o projecto de um amplificador capaz de respeitar uma determinada especificação de banda deve seguir uma metodologia própria (bem distinta da, infelizmente, muitas vezes seguida que consiste em projectar para banda estreita e, de seguida, esperar um milagre do optimizador linear), mas que esse esforço inicial pode ser bem recompensado.

REFERÊNCIAS

[1] J.G.Linvill, D.J.Mellor, "Two-Port Network Theory", Lecture Notes.

[2] Anatol I. Zverev, "Handbook of filters synthesis", Jonh Wiley & Sons Inc., 1967.

[3] G. Temes, S. K. Mitra, "Modern Filter Theory and Design", Jonh Wiley & Sons Inc., 1973.

[4] H. Krauss, C. Bostian e F. Raab, "Solid State Radio Engeneering", Jonh Wiley & Sons Inc., 1980.

REFERÊNCIAS BIBLIOGRÁFICAS QUE AJUDAM À COMPREENSÃO DESTE ASSUNTO

P. Abrie, "The Design of Impedance - Matching Networks for Radio - Frequency and Microwave Amplfiers", Artech House Inc., 1985.

R. Carson, "High Frequency Amplifiers", Jonh Wiley & Sons Inc., 1975.

Comunicação de Dados em Sistemas Domóticos, usando as Redes de Potência (*Power Line*) e a Antena Colectiva (CATV)

José M. P. B. O. Antunes, Pedro M. M. Mostardinha,
Sidónio M. Brazete,A. Manuel de Oliveira Duarte

Resumo- **Neste artigo chama-se a atenção para os dois meios mais usados da Domótica, *Power Line* e CATV (Antena Colectiva). Apresentam-se as suas características mais relevantes, bem como as principais normas que regem as comunicações neles assentes.**

Descreve-se, ainda, o desenvolvimento de um *modem* para CATV. Este *modem* tem em vista possibilitar a ligação entre os apartamentos e o concentrador geral de informação.

Abstract- **This paper presents an overview of the most common networks used in Home Systems, Power Line and CATV (Collective Antenna Television). The main characteristics and rules for communicating in these networks are shown here.**

The development of a CATV modem is described. This modem will be used to perform the link between the flats and the manager of information of the building.

I. INTRODUÇÃO*

As comunicações de dados nos lares actuais revestem-se, hoje em dia, de uma crescente importância. O desenvolvimento de novos meios físicos para essas comunicações enfrentam alguns problemas que dificultam a sua implantação. Por exemplo, a instalação de um sistema destes num edifício já existente, implicaria a necessidade de se realizarem obras o que leva a encargos suplementares, que se iriam reflectir no custo final da instalação.

Fazendo face a este problema, apresenta-se uma solução bastante atractiva, que consiste em usar as infra-estruturas já existentes no edifício, nomeadamente, a Rede de Potência (*Power Line*), assim como a Rede da Antena Colectiva (CATV).

Assim, o grande esforço da Domótica, ramo recente de aplicação das tecnologias de Informação e das Telecomunicações, direcciona-se para o estudo e implementação de sistemas que proporcionem segurança, economia, lazer e serviços aos utilizadores, bem como explorar as máximas potencialidades que se podem extrair dos sistemas já instalados.

A qualidade das comunicações usando a rede de potência *versus* distâncias envolvidas, vocacionam esta rede para comunicações domésticas dentro de um mesmo apartamento. Apresentando, também, a seu favor o facto de estar acessível de uma maneira bastante fácil em todas as divisões do lar.

A rede CATV tem uma maior utilização para as comunicações para o exterior dos apartamentos, porque dentro duma casa este meio está disponível em apenas algumas divisões. Em contrapartida esta estrutura chega a todos os apartamentos de um edifício, podendo assim ser o meio físico a utilizar para comunicações entre os apartamentos e um concentrador geral de informação do edifício, funcionando como a «espinha dorsal» das comunicações. No âmbito deste projecto foi desenvolvido um *modem* CATV para comunicações entre os apartamentos e o concentrador geral de informação.

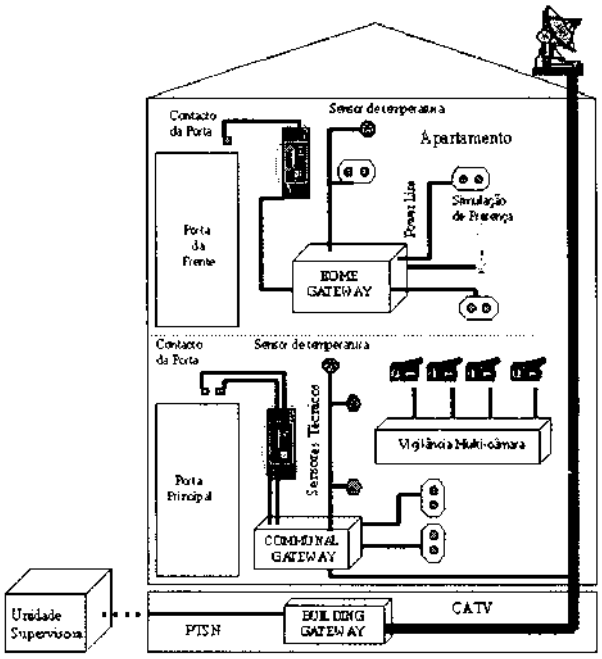


Fig. 1 - Edifício equipado com um Sistema Domótico

Este *modem* codifica os dados digitais usando uma modulação FSK que irão ser injectados num cabo coaxial. Pretende-se que não existam interferências nos canais televisivos que são distribuídos pela antena colectiva (ex.: TV por cabo, canais da antena parabólica comum), para isso foi desenvolvido também um filtro activo que limita a banda passante do Modem.

* Trabalho realizado no âmbito da disciplina de Projecto.

II. ARQUITECTURA DE UM SISTEMA DOMÓTICO

Um edifício equipado com um sistema domótico permite aos seus utilizadores um sem numero de facilidades, como por exemplo::

- Controlo da temperatura
- Simulação de presença
- Gestão de consumos (Electricidade, Gás e água)
- Monitorização de alarmes (Fogo, Água, Gás e intrusão

Todos estes serviços têm uma implementação transparente para o utilizador, podendo a sua administração e controlo ser efectuado por um elemento exterior ao edifício. Este tipo de serviço contribui para uma maior comodidade e segurança, para todos os que dele desfrutam.

III. REDE DE POTÊNCIA (*POWER LINE*)

A evolução registada, nos últimos anos, ao nível dos dispositivos domésticos, permite pensar que num futuro próximo estes venham a transferir informação via *Power Line,* de uma maneira generalizada. Tornam-se assim em sistemas económicos, pois não existe a necessidade de se instalar cablagem extra para a realização destas comunicações.

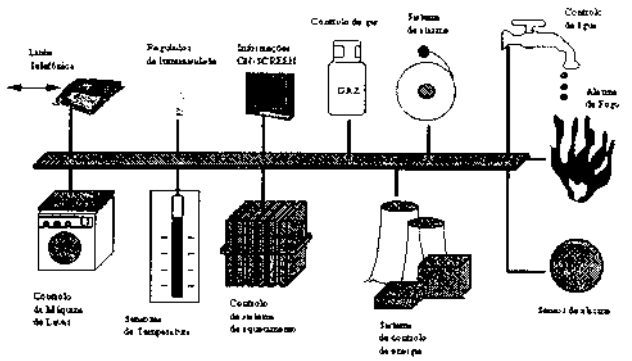


Fig. 2 - Cenário Típico da rede Power Line

A ligação dum qualquer dispositivo à rede é extremamente simples, basta ligá-lo a uma tomada existente numa das divisões da casa. No entanto temos que ter em atenção o seguinte, para que dois dispositivos comuniquem temos que garantir que estes estejam ligados na mesma fase da rede (R, S ou T). Isto é, se os aparelhos estiverem em fase diferentes (no caso de uma alimentação trifásica) não existe comunicação, a não ser que exista para além do *modem* um sistema capaz de executar o *routing* das mensagens entre as várias fases.

Um cenário típico desta rede é apresentado na figura 2. Pelas facilidades aí apresentadas pode-se avaliar a grande versatilidade deste bus. Bus este que é capaz de assegurar as ligações entre todos estes recursos que se encontram espalhados pela casa inteira.

No âmbito do programa europeu ESPRIT, *Home Systems*, foram desenvolvidos vários dispositivos que, para este meio de comunicação, respeitam as normas CENELEC 50065 - EUROPE.

Um *modem* para *Power Line* deve pois respeitar um certo numero de especificações.

As frequências de transmissão devem estar compreendidas entre:

- de 95KHz a 125khz e de 140khz a 148KHz , se a comunicação não tiver um protocolo definido;
- de 125KHz a 140KHz, se existir um protocolo definido para essa comunicação.

Verifica-se que a impedância da linha para as frequência de transmissão de 100 KHz, tem uma grande variação do seu valor. Apresenta valores entre 1,5Ω e os 80Ω, o que implica que o *modem* tenha que ser projectado de forma a que trabalhe independentemente do valor da impedância da linha.

Devemos realçar, ainda, que esta variação de impedância da linha, é devida não só às cargas existentes no nosso lar, mas também à carga da linha vinda do exterior

Além disso as especificações CENELEC impõem valores máximos para os sinais enviados. O nível máximo do sinal de saída é de 116dBμV, e os seus harmónicos não devem exceder os 46dBμV. Estes valores anteriores são respeitantes às condições imaginárias de rede (50Ω/50μH+5Ω) que simulam a *Power Line* o que equivale a uma rede de impedância 30.4Ω transmitindo a uma frequência de 132.45kHz. Estas limitações do nível do sinal destinam-se a evitar a ocorrência de possíveis interferências.

IV. *MODEM* PARA CATV (MOTIVAÇÃO)

O sistema por nós desenvolvido, tira partido das cablagens já instaladas no edifício, neste caso o cabo da antena colectiva (CATV).

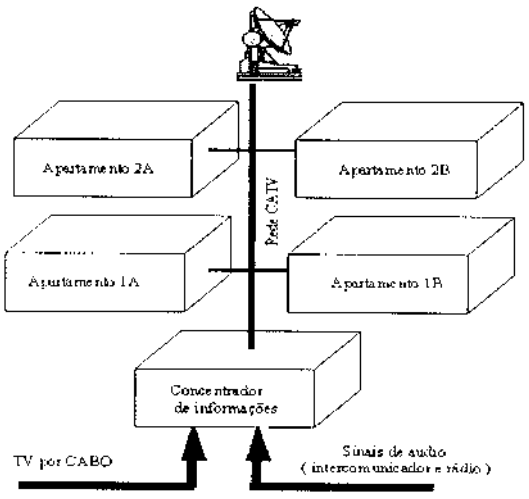


Fig. 3 - Cenário típico da rede CATV

As vantagens do uso deste suporte físico para a transferência de dados dentro do edifício são evidentes, nomeadamente:

- O CATV é uma cablagem que chega a todos os apartamentos do edifício;
- O tipo de cabo coaxial usado na antena colectiva apresenta uma boa largura de banda, pelo que podemos explorar esta qualidade, de forma a rentabilizar ao máximo este sistema;
- Podermos aproveitar o facto anterior para realizar a transferência de dados, não em banda base, mas fazendo a modulação de uma portadora.
- O uso de modulação, que neste caso específico é do tipo FSK, traz vantagens em termos de ruído, pois a banda utilizada para a comunicação pode ser escolhida de forma a registar uma maior SNR.
- Como o sinal modulado não tem componente DC e como este não transporta qualquer informação, existe uma economia de potência logo à partida.

A. Especificações

Este trabalho foi realizado no âmbito do projecto CHIMENE (*Collective Home Iterface Made on Existing Networks in Europe*), que apresenta trabalho desenvolvido na área da Domótica. Este grupo estabeleceu normas no que respeita às comunicações tanto no interior do apartamento assim como na área comum edifício.

O sistema realizado teve como directrizes as seguintes normas:

- Tipo de modulação: FSK
- Comunicação do tipo *Half-Duplex*
- Frequência de portadora: 25.3MHz
- *Thereshold* de detecção: 50dBμV

Este *modem* teve como objectivo fazer o *interface* entre as várias placas HDLC secundárias [4], existentes uma por cada apartamento, e a placa HDLC primária [4], que é a que se encarrega da concentração de todo eventos a nível do edifício, instalado no concentrador geral de informação.

B. Descrição do trabalho

O projecto por nós desenvolvido pode ser subdividido em dois blocos: um bloco que se encarrega da modulação, e um outro que faz a operação de desmodulação.

Como atrás referido, este MODEM vai ser utilizado na comunicação entre placas que implementam o protocolo de comunicação HDLC. Estas placas estabelecerão uma comunicação do tipo *Half-Duplex*. Surgiu, então, a necessidade de controlar o sentido da comunicação, pois o suporte físico é o mesmo em ambos os sentidos. Para esse efeito, usámos uma linha I/O do microprocessador P8044 da ***Intel***. Essa linha de controlo é usada na comunicação série para indicar qual o sentido da comunicação.

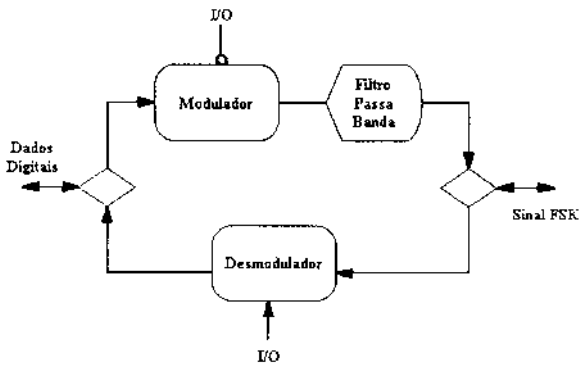


Fig. 4 - Diagrama de Blocos do Modem

1. Modulador

O módulo modulador é compatível com a *Standard* IEEE norma 802.4, dando no entanto a possibilidade de ser usado noutras condições de funcionamento.

Neste sistema é possível actuar sobre dispositivos de regulação, que permitem por exemplo, predefinir o tempo máximo que é permitido para uma emissão sem interrupção, ou ainda poder actuar sobre uma malha LC, de forma a modificarmos o valor da frequência da portadora para os valores desejados.

2. Desmodulador

Este módulo é também compatível com as normas IEEE, 802.4, sendo este o homólogo do bloco anterior.

O módulo desmodulador foi constituído, de forma a podermos ajustar certos parâmetros, para que a detecção se faça o melhor possível. Por exemplo, podemos fazer ajustes da frequência de portadora que o desmodulador está à espera de receber e podemos, ainda, impor qual o nível de sinal, a partir do qual, o que chega, à entrada do desmodulador, será considerado como informação e não como «lixo».

Associado ao desmodulador temos ainda um filtro passa-baixo que tem a funcionalidade de retirar os harmónicos da portadora, bem como o ruído provocado pelo meio em que o sistema está inserido.

3. Filtro activo

Durante os teste efectuados com o MODEM FSK para CATV, foi verificada a necessidade de adicionar um filtro passa-banda na saída do modulador para a linha, pelo facto de este provocar ligeiras interferências sobre os canais televisivos.

Para resolver este problema escolhemos um filtro activo, porque além de fazer a filtragem dos harmónicos produzidos, faz também o *drive* para a linha do cabo coaxial. Após a introdução do filtro passámos a respeitar as normas, quanto aos níveis de ruído causado sobre o espectro no CATV. A figura seguinte ilustra os resultados obtidos no analisador de espectros.

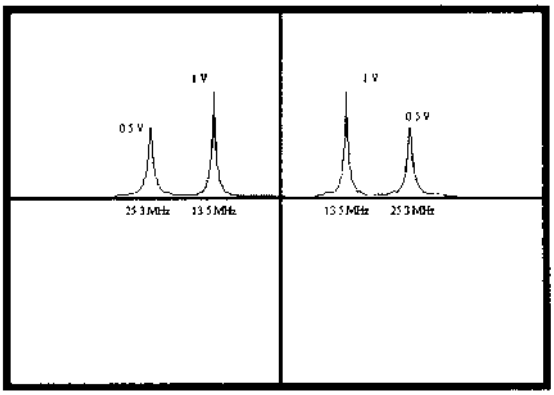


Fig. 5 - Resultados Obtidos no Analisador de Espectros

C. Testes e resultados

O sistema final começou por ser testado na comunicação entre as placas HDLC, para muito pequenas distâncias, partindo depois para testes de longas distâncias, entendam-se longas distâncias relativamente às envolvidas na rede de cabo coaxial existente dentro dum edifício.

Foram realizados testes, colocando uma placa num extremo da rede CATV e outra placa noutro extremo. Estima-se que a distância entre extremos seja de 500m.

Na posse destes resultados, elaborámos a tabela apresentando a estimativa do alcance deste sistema, garantindo, ainda, a fiabilidade da comunicação.

Esta estimativa tem em conta o tipo de cabo coaxial usado, bem como a frequência da portadora.

Frequência da portadora	Máxima taxa de transmissão	Tipo de Cabo			
		RG-59	RG-11	JT34125	JT3750J
1MHz	500Kbaud	1.8Km	6.4Km	10Km	15Km
3MHz	1Mbaud	1.5Km	3.6Km	6Km	9.7Km
5MHz	2Mbaud	1.2Km	2.8Km	4.5Km	7.6Km

Fig. 6 - Tabela da relação entre Distância e Frequência da Portadora

Como se pode verificar a distância depende não só da frequência, mas também do tipo de cabo coaxial utilizado. Assegurando uma distância máxima de 15 Km quando o cabo utilizado é o JT3750J, para uma portadora de 1 Mhz à sua máxima taxa de transmissão.

V. CONCLUSÕES

No final do projecto, obtivemos um sistema, que apresenta boa fiabilidade, associada a um bom comportamento no que respeita às distâncias envolvidas nas comunicações.

Este sistema permite futuras aplicações em edifícios, residenciais ou comerciais. Permitindo a instalação de novos serviços sem a necessidade de novos suportes físicos, tendo como consequência uma considerável redução de custos.

Este trabalho fez parte do projecto CHIMENE (*Colective Home Interface Made on Existings Networks in Europe*), que está englobado no programa ESPRIT (*European Strategic Programme for Research in Information Technologies*), da colaboração do Grupo de Sistemas de Banda Larga com o projecto CHIMENE, resultou na instalação de um destes sistemas numa residência de estudantes, permitindo o desenvolvimento e teste de novos sistemas em condições reais de funcionamento.

REFERÊNCIAS

[1] CHIMENE (Collective Home Interface Made on Existing Network in Europe), "Deliverables of the workpackage 2", Release V1.0, May 1993.

[2] ESPRIT (Home System Specification), ESPRIT-HS consortium, January 1991.

[3] "Power Line Modem ST7537", SGS Thompson Microelectronics, April 1994.

[4] H. Vale, J. Duarte, Relatório de Projecto, Julho de 1995.

Digital Virtual Neuroprocessor: Design experience using Verilog HDL

Jorge Velez, António de Brito Ferrari

Resumo— **No presente artigo são apresentados o procedimento adoptado e a experiência obtida no projecto e desenvolvimento de um processador neuronal, utilizando Verilog HDL. A descrição das várias etapas de projecto no sentido da descrição final do sistema e do correspondente ambiente de simulação, constitui uma ilustração concreta da aplicação das capacidades de uma metodologia baseada na descrição e simulação de hardware no projecto de sistemas.**
A configuração final do computador neuronal integra o ambiente de simulação de Verilog HDL como ferramenta fundamental para o teste e desenvolvimento de futuro software neuronal.

Abstract— **This paper presents the design flow and the resulted experience in the development of a neural processor, using Verilog HDL. Describing the design steps toward the final system's description and simulation environment, provides a concrete illustration of the application of Verilog HDL features in the system's design testing, characterization and improvement, at the early stages of the project.**
The end-use neural computer's configuration comprises the Verilog HDL simulation environment as a fundamental tool for future neural software test and development.

I. Introduction

The implementation of artificial neural networks (ANNs) by software simulators running on serial processing computers, is not suitable for most "on the ground" applications where the real time speed requirements are unreachable even if the fastest serial computers on the market were used. For this domain of applications the use of special parallel hardware implementations of ANNs is required.

This paper describes the design methodology used in the development of one such system. It presents the experience gained in building the Verilog HDL description of a digital neuroprocessor within the system where it is to be integrated. The system description and specification is presented in the following section.

As the complexity of current systems has been growing, the designers have increasingly adopted a high-level description of the system to be designed as the starting point in the development process. Current hardware description languages (HDLs) such as VHDL or Verilog, support the design hierarchy, allowing mixed-level descriptions to be submitted to simulation. They provide a means for designing different parts of the system, using distinct development strategies. The datapath of a processor can be simulated with a behavioral architectural description of its arithmetic and logic unit. Once all the synchronization and clocking schemes have been validated, the arithmetic and logic unit may be substituted by a more detailed structural model. The same may be applied for its constituting parts and so on.

Besides documenting and stating unambiguosly the system's specifications, the use of an HDL allows for their verification without resorting to iterations through the time consuming phase of implementation.

Designing a special purpose processor constitutes a good example of how useful an HDL description and subsequent simulation, can be. Often being distant from the traditional fairly tested architectural solutions, the exploration of the design space through the evaluation of the various alternatives coming up during system development, is of great importance.

The advantages of using an HDL do not end up with the completion of a project. In current integrated circuit (IC) design, the existence of a correct system description is very useful when testing the fabricated chip. The test vectors can be previously evaluated and the HDL simulation responses can then be compared to the real responses from the fabricated IC.

II. Basic Specifications

The objective is to design and build a computer whose architecture is oriented toward the efficient implementation of ANN models (neural computer or neurocomputer), in particular Multi-Layer Perceptrons (MLPs) with the backpropagation (BP) algorithm [1][2]. The architecture does not make any restrictions to the network configuration, allowing for the mapping of nets with different dimensions and any distribution of neurons among different layers.

Due to the heavy computing requirements of such models in current real-life applications, the neurocomputer to be built is based on a parallel configuration. Connected as an attached processor to a host computer, it is constituted by a number of processing units communicating via a common data bus (fig.1).

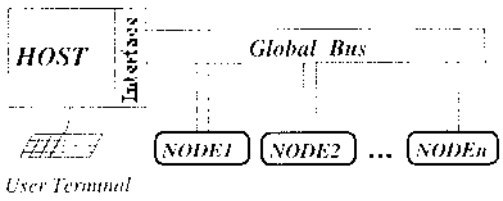


Fig.1

The proposed architecture envisages simplicity and low cost adopting solutions to the typical bottlenecks. The characteristics of neural nets (large number of neurons and interconnections, simple internal processing) make the algorithms communication-intensive. Hence the limitations in communication bandwidth tend to be the main bottleneck in multiprocessor neural architectures. The fundamental system characteristics (fig. 2) can be summarized as follows:

- *Neuroprocessor* - each node in fig.1 includes a special-purpose floating-point processor implemented as a VLSI chip, and its own local memory.
- *Parallel architecture easy to expand* - by implementing inter-node communications through a tagged inter-communication bus type, the architecture's typical bandwidth bottleneck, can be minimized.
  The system is scalable simply by adding new VLSI chips onto the global bus.
- *Broadcast communication type* - similar to the classical single bus architectures, its design principles come from the observation that nowadays memory is cheap in comparison to arithmetic resources [3][4]. Providing each node with sufficient local storage, traffic on the bus can be reduced to the new neuron activations [1]. The learning algorithm must be adapted to accomodate global communication.

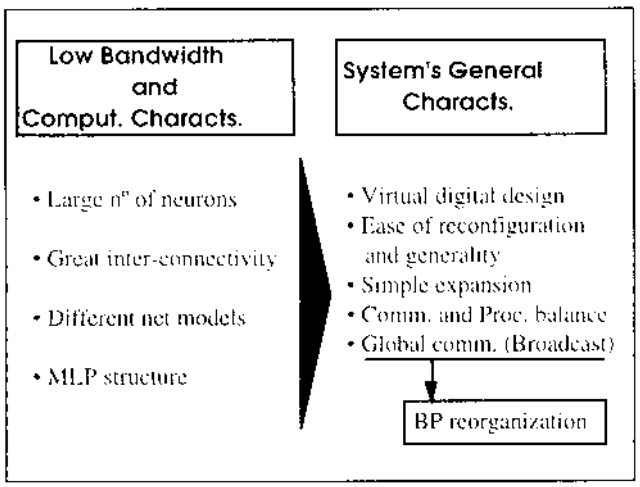


Fig. 2

- *Reconfigurability and Generality* - in programming mode, apart from loading the neuroprocessor microprogram, each node also receives the information about the network topology. Such topology is totally re-configurable through the host computer.
  Although the architecture was thought for improved performance with the BP algorithm, it may also be used for executing other algorithms on other ANN models.
- *Computation and Communication Balance* - a node is designed to fully support the overlap of communication with internal processing. Communication and processing are asynchronous, exchanging data through input/output first-in-first-out (FIFO) buffers.

The operation is controlled by the host and encompasses three main phases (modes):

1. **Programming** of the system by the host, where the network topology is defined, and the initial values for the weights are loaded into each node. After programming the operation may proceed to the training or recall modes.

2. In **training** mode the host presents the training patterns and searches for the output neuron activations (*forward* step). When one of such activations is read, the host calculates the error and sends it to the bus. Once the backpropagation of errors is complete (*backward* step), another input pattern is presented. At the end of the training set, all the nodes update their weights (*update*).

3. **Recall**, where the operation is just as in the forward phase of the learning process.

III. NODE ARCHITECTURE

Since the data word length has a major impact on operation speed and silicon area, a reduced floating-point representation was investigated [2]. The conclusion was that a 16-bit floating-point format (1-bit sign + 5-bit exponent + 10-bit mantissa) provided the required precision and dynamic range. Each processor consists of two separate units working asynchronously. A communication unit, the node's interface with the global bus, and a processing unit whose arithmetic resources are multiplexed for each neuron in a node (fig. 3).

A. Communication Unit

The chosen bus protocol is the 3-wire Daisy Chain (GRANT, REQUEST, BUS-BUSY), arbiter centralized at the

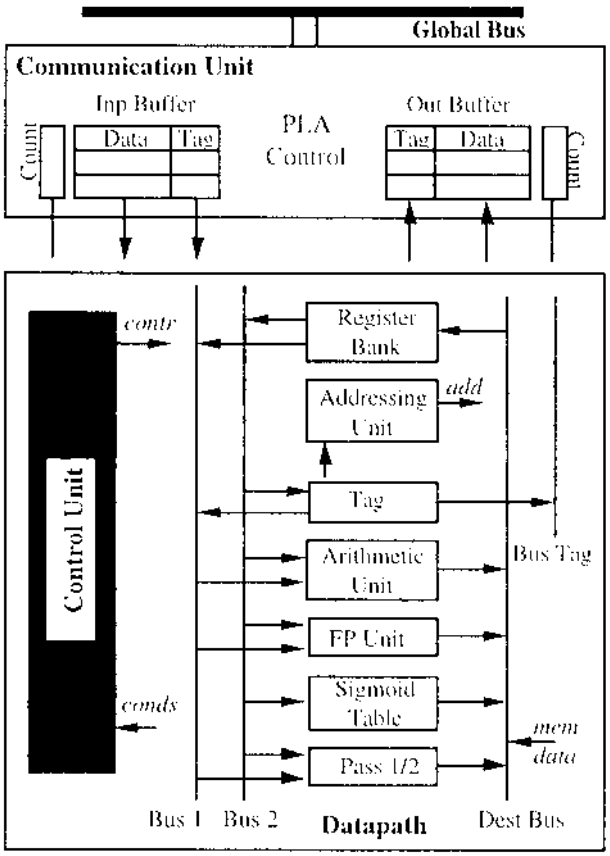


Fig. 3

host bus interface. How this choice affects performance was subject to experimentation through the final HDL simulation [2] and it will be discussed later.

The communication unit consists basically of input and output FIFOs (data and tag fields) and of a PLA-based controller which also implements the Daisy Chain protocol. The length of these buffers has been investigated [2].

B. Processing Unit

The processing unit, divided in control unit and datapath, is a (micro)programmable processor with a dedicated instruction set. A hardwired design would compromise the desired neuroprocessor flexibility.

This unit's fundamental design decisions derive from the system specifications and the BP processing characteristics:

- *Control Unit* - this unit is microprogrammable and based on SRAM where the machine code implementing the neural algorithm is to be loaded. The access time of the control memory is dependent on its dimension. Hence it is important, both for chip area and for processing speed, to keep the SRAM small.

- *Addressing Unit* - according to the tag coming through the bus, the neuron being processed and the type of data involved (weight, update or activation), a memory position has to be determined. This is a frequent operation, therefore the addressing scheme is of crucial importance to the processor's performance.
  Two addressing modes are supported. An absolute mode with no arithmetic required and a displacement mode involving the summation of the incoming tag, the neuron base address and their dimensions.

- *Floating-Point Unit* - the neurons intrinsic calculations (summation and multiplication) are executed by a 16-bit floating-point unit (FPU). The short mantissa greatly contributes to the feasibility of such FPU.

- *Sigmoid Table Look-up* - by using the exponent bits and the sign value, a 64-position SRAM implementing the non-linearity, is directly accessed. Another implementation choice would involve heavy calculations.

Other important relevant parts are:

- *Integer Arithmetic Unit* - the integer arithmetic unit is meant, basically, for pointer and index calculations needed to implement loops in the algorithms. Since the addressing unit also uses a 16-bit adder, a hardware multiplexing scheme was implemented so that some silicon area can be saved.

- *Register Bank* - there are 8 general purpose registers. Although this number represents a good tradeoff, a few more would allow for a slight decrease in memory traffic.

IV. VERILOG AND VERILOG-XL

Verilog (**Verify Logic**) is the HDL most used in industry, particularly in the US. Introduced in 1983 by Gateway Design Automation it was later acquired (1989) by Cadence Design Systems that offered it to standardization (1991). As a result a number of third-party simulators, most of them running on PCs, is now available.

The early availability of an efficient simulator supporting all the language constructs and well supported by the system environment (Verilog-XL), together with its similarity to the C programming language, were instrumental to its widespread acceptance. Its availability to universities under EUROCHIP, and its easy integration with Cadence Design Framework, together with the characteristics mentioned, led to the decision to choose Verilog as the HDL to be used.

The logic simulation process with Verilog-XL involves three separate steps: creation of a system model,

providing stimulus to exercise the model and indicate the format in which results are to be viewed. The simulation sessions may be interactive allowing the user to monitor the system variables, forcing new stimulus or changing the previous ones. The simulation may then proceed normally, step by step with or without trace [5].

Input/Output of data from and to files under the operating system, is possible. Such features allow the gathering of statistics, register final values, etc, and the reading of data values to memories, PLA contents, etc. The output display includes normal printing in the working window, fixed position textual printing, bars or waveforms. The last three are updated as time evolves. Past events can be viewed moving a cursor in a time bar. The output facilities and the great simplicity of the related commands, are undoubtedly a further strong motivation to use Verilog-XL.

The language basic concept is the **module**. It represents a piece of hardware connected with other modules through inputs and outputs, called **ports**. Such modules can be part of a hierarchy as a block of hardware that can be used one or more times in another module.

The modules functioning can be described either behaviorally or structurally. Behavioral descriptions can be done at the register transfer, algorithmic or architectural levels. The structural, netlist type, may be described using primitives such as gates or through transistor models (switch level).

All levels, behavioral or structural, can be mixed in a same description and submitted together to simulation. Thus high level behavioral descriptions of parts being designed, can be simulated together with the structural description of already implemented blocks.

The time concept constitutes the main particularity of HDLs. Since every piece of hardware works in parallel through time, Verilog code is executed in an event-driven way. Every action is triggered by specific user-defined events or through the value of a global variable representing the system simulation time.

The package language+simulator could be used as a standalone product or integrated in Cadence Design Framework. Before going to the implementation phase, for practical reasons, the standalone use is advised.

V. NODE DESCRIPTION

As referred, the first stage toward the neuroprocessor implementation was the C language neural network simulator. By defining the processor's data format, the basic specifications were finally achieved.

Design and description are indivisible. Naturally, ideas come first to light through scratches on paper. However, by creating models of the pieces of hardware, whatever the description level is, such ideas can easily be judged, for invalidation or for adoption. The advocated architecture is the basis for this kind of development. In the present design, by architecture one refers not only to the system but also to the node's architecture.

Level independent, the description structural detail is also a matter of concern. Using behavioral constructs one can build a module either modeling its function with behavioral procedural blocks and register transfers, or by detailing its constitution. By other words, one may use only the behavioral level to model the block function or, in a structural way, implement it with smaller and simple behavioral level modules. This latter type of description can be easily and automatically transformed into an ordinary schematic by mapping those simple modules, to the correspondent standard or user-developed cells, of a specific technology.

The system design flow will be exposed as the sequence of its main subparts description and their linking procedure (fig.4). All the description top hierarchical modules and the ones resulting from joining them, were simulated before being linked again. Some requirements and consequences of such procedure can also be viewed in fig.4.

A. Communication Unit

In the communication unit design, the first step was to implement the bus protocol. Thus, an arbiter and a controller for the 3-wire Daisy Chain, were built. The controller is part of the communication unit while the arbiter belongs to the host interface. An extra line (inhibition line) is included to prevent the arbiter to grant the bus if any of the input buffers gets full.

Having validated the bus protocol scheme description, the remaining bus communication unit functions were then described. Since its control will be a PLA-based state machine, it wasn't important to go deep in detail. At this stage, spending time describing the PLA contents and surrounding logic, was unnecessary. Instead, there were written some behavioral code lines to implement all the block functioning.

For an initial communication unit test, **statistical distributions** were used for representing the items processing time and the time interval between production of two consecutive activations. Besides the **waveforms** display, **graphical output bars** monitor the buffers occupation through time and the maximum number ever, in them. **Statistics** such as mean inter-arrival or longest wait times, are also generated.

Apart from validating the basic architectural concepts, several important conclusions could already be obtained. It was shown that the fixed priority established by the Daisy Chain protocol, had no effect unless the input buffers got full. However, such situation only affects performance if the output buffers are also filled up, which rarely occurs if the buffer lengths are well dimensioned. Some description related details and possible deadlock situations were identified. Communication aspects since they depend on algorithms, topologies and relative

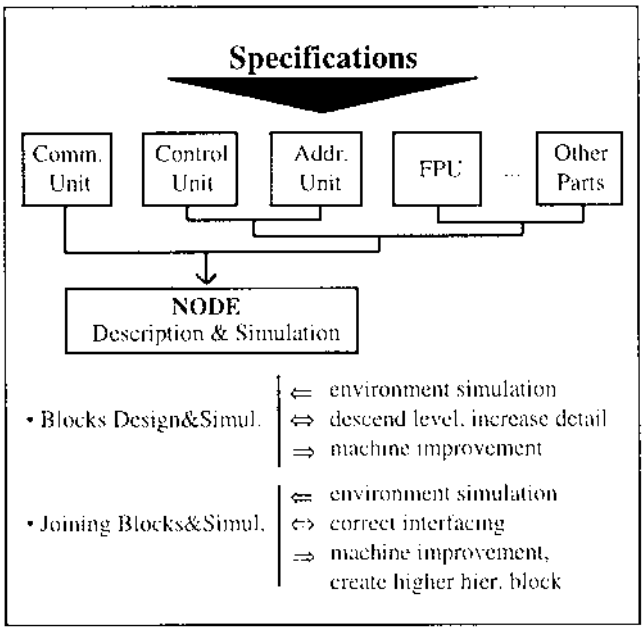


Fig.4

processing times between nodes, were not studied at this stage.

B. Control Unit

Independently from the bus communication module, the processing control state machine was specified. The main purpose of this first version was to validate its functioning principles and related clocking strategy. By supplying the host lines and emulating the conditions coming from the datapath, not yet built, all the unit synchronization was defined.

The description detail was such that behavioral models were only applied to pieces of hardware as simple as multiplexers, latches and registers. In some constituting blocks the detail evolved from less detailed descriptions. Behavioral code with no detail, was used first to determine or prove the effectiveness of the functional specifications of those blocks. This illustrates a development process starting from a high abstract level and ending with a schematic equivalent structural description. Thus, it will be quite fast to go from such a description to implementation.

At this stage it was important to question the feasibility of having SRAM for the control memory on-chip, though the details for the μword format were not known yet. In order to have an idea about the physical aspect of such memory, ES2's parameterized SRAM cells were used. The respective access protocol and timings, were followed.

C. Addressing Unit

A first module version was created so that the addressing scheme under the adopted clock strategy could immediately be validated. Separate parts descriptions constitutes not only a way to immediately verify the idealized architecture but also a safe way to develop the top hierarchical modules, as they grow up by integrating smaller ones.

A second version was finally built which combines the arithmetic unit and the addressing unit in the same module, and implements the 16-bit adder multiplexing scheme.

D. Datapath Parts

Having in mind the advocated generality, though it will be directed to the BP algorithm, a first version of the instruction set was defined through translation of pseudo-code algorithms into assembler-like code. The required datapath's transfers and operations determine the hardware to build and to optimize.

The basic datapath parts are: the FPU, the integer arithmetic unit, a bank of general purpose registers, the addressing unit, the function table and a special register — TAG register. The corresponding modules were independently developed though there are some common building blocks.

After specifying the modules synchronization and interfacing, the μword format could finally be defined. Since this processor is user-microprogrammed, the possible μword's bits combinations give rise to a huge instruction set dimension. The allowed instruction set, assuring machine integrity, will be a small subpart of all the combinations.

The simulation of all the parts working together was quite extensive since the μword lines had to be emulated. Incorrect synchronization of the applied stimulus allowed some defects to pass, being detected later.

The final description detail is similar to the control unit one. It requires a small step to implementation. For instance, the FPU was fully designed. Its basic building blocks are well known pieces of hardware (ex.):

```
module arith_unit(out,ovw,undw,op1,op2,sub/add,enable);
    ...
    exponent_comp       exp_comp(diff,exp1,exp2);
    sign_detector s_det(out,signA,carry_add,diff);
    normalizer          ...
    ...                 ...
endmodule

module sign_detector(out,signA,carry_add,diff_signs);
    ...
    assign #delay out=(diff_signs &&
                              ~carry_add)?~signA:signA;
    //Implemented with simple combinatorial logic.
endmodule
```

The FPU project is certainly an excellent example to illustrate the importance of using an HDL and simulation. By including high level constructs to perform the data format conversions, the result could be displayed in both machine and neuroprocessor representations. The conversion between floating-point formats was done building specific Verilog code functions used just as in C. Their existence facilitates enormously the full FPU test and, in case of error, ease for determining its location.

E. Joining Control and Addressing+Arithmetic Units

Joining these parts, was the next phase. Well defined the cycles and synchronization of the respective instructions, by filling the correspondent μword bits, the whole scheme was then tested. Imperfections mainly relative to the interfacing, could now be fully detected and corrected. A working processor able to perform addressing and arithmetic operations, and branch according to the resulting conditions, was obtained. The external SRAM interfacing was also included.

F. Joining the remaining Datapath Parts

The complete synchronization scheme was defined and verified, at this stage. As a new piece of hardware was included, the new datapath transfers involved were immediately simulated and tested. It wasn't as hard as testing the units separately because the stimulus were now filled as the respective μword fields.

The occurring errors were normally and easily detected, either by consulting the register values permanently displayed on screen or by messages originated in modules. Such fault detection messages were quite useful. When a hardware error was detected, such as a floating line on which the next μword address depends, then a warning message was sent or an interrupt was generated.

To simulate "all" the specified instruction set, it was only necessary to emulate the communication unit asynchronous signals and the respective buffers.

G. Joining Communication Unit

Finally, the communication module was joined to the processing unit description, creating a complete neural node. By making instances of several neural node modules connected to the bus and its arbiter, the system simulation would then be done.

The external SRAM requirements had to be evaluated. Through a few calculations involving the number of neurons and connections foreseen, and being aware about constraints such as the limit number for the nodes on the bus, the capacity and speed of current commercial SRAMs were found to satisfy the needs (64K words, <20ns T_{acc}).

Designing with the target technology permanently in mind, is fundamental. The impact of some of the crucial system subparts on the final design, was studied whenever it was possible. Such study was normally supported by the ES2 design tool kit, as in cases of the multiplier, function table and PLAs.

VI. System Simulation

The simulation of the complete system required more than the description of its parts. A **system module** integrating the various neural processor modules, the external SRAMs and the global bus arbiter, was created.

It is necessary to provide a practical way for adapting the Verilog system description according to the topology and configuration in use. The system module uses the Verilog *include* directive to include the configuration dependent parts. Those parts, such as node instances, output commands, node port variables, programming, etc, are automatically generated through **C code**. The C routines also fill the external SRAM with the global configuration data and random weights, if specified.

The function table is created by generating in C, the sampled function values, and then, converting them into the specified 16-bit floating-point format. If the function and the number of points to represent it remain unchanged, this procedure has to be executed only once.

Since programming directly in microcode would be extremely difficult and time consuming, an **assembler** was built which implements a subset of the allowed instructions. If a special instruction is required or an optimization through merging consecutive μwords, is desired (e.g. within cycles), it can always be done using one of the assembler directives which specifies explicitly in hexadecimal format, the contents of a control memory word. Further capabilities include the use of labels and comments.

The test module, representing the host computer actions, resulted quite complex. The host tasks had to be emulated in Verilog. Such module implements the mode transitions, drives the items into the global bus, snoops the bus for activations, controls the supervision line (e.g., to command the update operation), collects statistics and measurements of various system parameters, monitors the system variables for on screen visualization, etc. In addition to the described complexity, almost all the tasks above require re-writing the code whenever there is a configuration modification or a change in some other system parameters.

The devised functional description for the complete system, can be illustrated as in fig.5. Configuration and application data are situated at a different entry level since the algorithms shall be provided as part of a set of neural software and other tools.

Through distinct interface routines, the host front-end which comprises the assembler and all the configuration

routines, controls both the hardware implementation and the HDL description. The HDL simulation is useful not only while the system is being developed but also during the future practical applications. It will be used for new algorithms validation and characterization.

Accomplished the HDL description, several performance measurements can finally be made toward performance prediction and architecture evaluation. Communication and processing balance are dependent on algorithms, number of cycles to consume an item, number of cycles to produce an item, network configuration and network neurons correspondence to nodes, number of nodes, buffers length and bus protocol. A deep investigation on such matters demands extensive variation of system parameters and network configuration which may result in a huge simulation time. Additional data such as instruction use measurement for code optimization (size, efficiency) applied to the algorithm crucial parts, can be obtained.

Some predefined tracing analysis were provided: **bus utilization** vs time, **node processing** vs time and **buffer status** vs time. In such conditions, one can verify how different is the bus utilization during the various operating phases and its dependency with other parameter settings. The buffers length dependency on the network parameters, network partition and number of nodes, was appreciated.

The load distribution and its dependency on the simulation conditions, can be analyzed. For instance, different partitions of the same network can be tried toward a more even load balance. The number of output requests with the output buffer full and the waiting time for an item with the input buffer empty, may be counted. It measures the processing units forced latency.

Through the measurements in various simulation conditions, some architecture's parameters, such as its granularity and load balancing, can be evaluated. The HDL description and simulation can be used for full architecture performance characterization.

The following examples illustrate the importance and type of information possible through system simulation. Figure 6 is an example of bus utilization and nodes processing monitorization versus time, during the backward step of the BP algorithm (network 2x7x2 — 2 input neurons, 7 hidden and 2 outputs). The buffers size was 5 which allowed the output buffers to remain not completely full

The execution time constitutes a direct measure of system performance. It allows algorithm codifications appreciation, to qualify the network partitions, to optimize the system parameters (such as number of nodes and buffers size) for a particular application, etc. An example of performance versus number of nodes is presented in figure 7, for two distinct networks during the forward step or recall functioning mode.

The speed-up non-linearities are due to inevitable distinct number of neurons implemented by the different nodes in each simulation.

In relation to the buffers dimension analysis (figure 8 shows the direct graphical output in a specific simulation moment) several conclusions were obtained.

The maximum input buffer occupation is in fact imposed during the forward step for networks with hidden dimension higher than the output dimension. For the backward step the situation reverses.

The output buffer occupation is minimum as long as the input buffers are well dimensioned. Otherwise the maximum number of items possible is determined by the number of neurons implemented by the respective node. Therefore, a programmed solution for a possible deadlock when both buffers get full, is justified. Since such situation will be rare, a hardware solution would be too costly.

By locating the microprocessor control state machine eventual bottlenecks and by investigating their foreseen operation delays (e.g., consulting the ES2 tool kit),

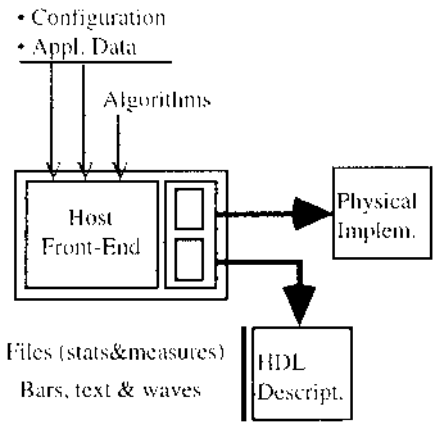


Fig.5

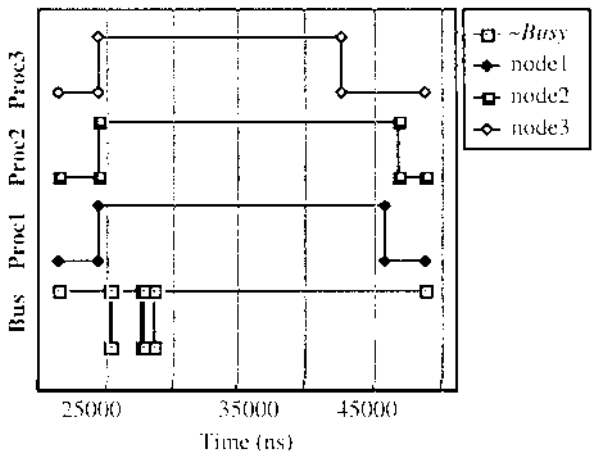


Fig.6

besides conducting the improvement efforts toward the most influent hardware aspects, it gives an idea about the future processing unit clock frequency. Among the cycle time constraining parts that may exist, it can be included the external SRAM access time, the ALU/ADDR unit operations or the FPU. Parts such as the function table, were eliminated as eventual bottlenecks.

To run the simulation only one terminal command is necessary. It indicates the system parts location (generally in separate code files), and what type of simulation delays is going to be used (**min**, **typ** or **max**).

VII. Simulation Difficulties

The main drawbacks of such system simulation, arise not from the testing and validating design phase but from building the whole system description so that its performance and architectural solutions, could be evaluated. Thus, the algorithms had to be machine programmable right away, the configuration (neural network and number of nodes) had to be automatic, programming issues and host control had to be emulated, etc. The creation of the surrounding environment (mainly host jobs) in conjunction with the various successive system parts test, constituted the less interesting and most time consuming phases.

The assembler had to be created to allow faster, lexical and syntactic error free code writing, floating-point format conversions were programmed, instances and variables in Verilog code had to be automatically created through specific C routines, the function table was also generated automatically, etc. A small modification, easy to accomplish on the HDL description, may have a much bigger impact on the configuration dependent routines. The code for collecting data, such as the statistics, had also to be automatically generated. However, Programming Language Interface (PLI) could certainly be used, in this case.

Verilog doesn't allow multidimensional constructs to be applied to variables and instances, otherwise, the configuration process could be practically automatic. Although such efforts should be unnecessary at this stage, one must notice that many of the configuration work is useful for the future host front-end package.

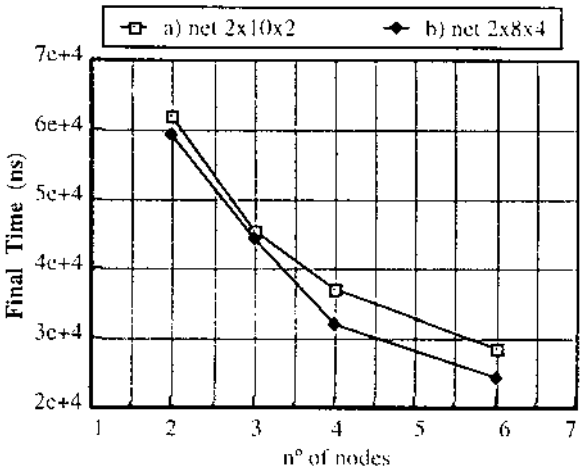


Fig.7

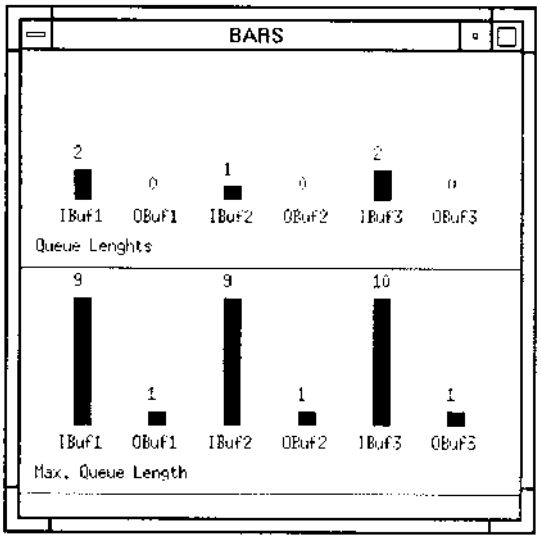


Fig. 8

The simulation time is too long for learning procedures limiting the simulations to few epochs (learning iterations over the entire training set). It certainly is not at all surprising, since even with the C language simulator, the algorithm could be, for large networks, very time consuming. For accelerating the simulation, the graphical output may be disabled.

VIII. Conclusions

Several HDL general advantages were proved and illustrated throughout this system's development. The description and simulation, easy to accomplish, are fundamental for a methodic and coherent design flow. They constitute a "helping rope" linking the former development stage, to state the specifications, to the final test design step.

Through the use of Verilog HDL, by clearly specifying the different modules and their interconnections, division for independent parts design, can easily be done. Therefore, it represents an adequate development tool for a design group job division, the way successful commercial products are currently developed.

The initially proposed architecture, could be validated in its principal aspects. The built HDL description constitutes a powerful framework for measuring, testing and, consequently, improving the system. In particular, several fundamental design decisions and optimization, mainly related to communication aspects, this system

typical bottleneck, were supported by the built simulation framework.

The possibility for creating the surrounding environment and emulating the real stimulus to the processors, gives more credibility to the simulated tests. Also, being aware of the target technology, there is no separation from realizability even when high level models are used.

The choice of level and description detail shall always consider, besides the modelling purposes, minimization of time to produce the code. For an immediate great detail description the schematics method would be preferable.
To pass to implementation will be an easier task. The designer can now concentrate his efforts onto the implementation aspects rather then worrying about the architectural solutions to be devised.

In parallel with the physical system, the HDL description will be useful for future use. Apart from end applications use, it is proper for investigation and teaching purposes. In addition, it will be fundamental for validating future user programmed software, before running it in the nodes.
Verilog HDL reveals serious handicaps to deal with varying configuration models. In the future, it should feature multi-dimensional constructs for variables and module instances. Mainly due to configuration dependent code, a great deal of C code is necessary for achieving the proposed objectives.
The user-friendly simulator graphical output, very attractive, constitutes an extra motivation for Verilog use.

REFERENCES

[1] António de Brito Ferrari, Y.H.NG, "A Parallel Architecture for Neural Networks", in *Proc. Parallel Computing*, 1991.

[2] J. Velez, "Processador Neuronal Virtual Digital para Implementação VLSI", Tese de Mestrado em Engª Electrónica e Telecomunicações, DETUA, Universidade de Aveiro, 1995.

[3] R.Kuczewski, M.Myers, W.Crawford, "Neurocomputers Workstations and Processors: Approaches and Applications", IEEE 1st ICNN, 1987, San Diego, CA.

[4] Robert Hecht-Nielsen, "Neurocomputing", Addison Wesley, 1990.

[5] "Verilog-XL Reference Manual", 2 Volumes, Version 1.6 March 1991, Version 1.6a November 1991 Release Notes, from Cadence Verilog Manuals.

Sistema de Apoio à Decisão para Problemas de Cobertura: definição e implementação de uma interface de utilizador (1)

Salviano Filipe Pinto[1], Beatriz Sousa Santos[2], Carlos Ferreira[3], José Alberto Rafael[2]

[1]Departamento de Engenharia, Universidade de Trás-os-Montes e Alto-Douro
[2]Departamento de Electrónica e Telecomunicações, Universidade de Aveiro
[3]Departamento de Matemática, Universidade de Aveiro

Resumo **- Apresenta-se uma interface para um sistema destinado a apoiar utilizadores na decisão baseada em Problemas de Cobertura (Simples e Múltipla) em ambiente multicritério, de uma forma interactiva que se pretende fácil de aprender e de utilizar. Este trabalho foi desenvolvido numa plataforma Windows, utilizando o Visual Basic™ 3.0.**

Abstract **- This paper presents a user interface for a Decision Support System on Covering Problems (Set and Multiple) using multicriteria models. Since this system must be interactive and easy to use and learn, its interface was developed on a Windows platform, using Visual Basic™ 3.0.**

I. INTRODUÇÃO

Os modelos de Cobertura, em Optimização Combinatória, têm uma vasta gama de aplicações práticas nas áreas de transportes, pesquisa de informação, parcelamento geográfico (político, hospitalar e outros), planeamento de produção e de redes, comunicação por satélite e em especial na localização de equipamentos. As referências mais importantes sobre estas aplicações podem encontrar-se em [1,2]. Estes modelos têm sido intensivamente estudados na formulação monocritério (tradicionalmente minimizando o custo) e existem algoritmos relativamente eficientes para a sua resolução. No entanto tem sido reconhecida a insuficiência desta aproximação, nalgumas das aplicações anteriormente referidas, dada a natureza multiobjectivo inerentemente à vida real. Como primeiro passo, para explicitação das compensações entre os objectivos conflituosos que surgem na aplicação real destes modelos, foi desenvolvida uma metodologia de resolução de modelos de Cobertura Bicritério, utilizando uma aproximação interactiva eficiente [3].

Neste artigo apresenta-se uma interface para um sistema baseado na referida metodologia e que permite, nesta fase, a utilização de dois modelos de Cobertura: Cobertura Simples (CS) e Cobertura Múltipla (CM).

O sucesso duma aplicação deste tipo está fortemente condicionado pelo tipo de interacção com o utilizador e apresentação dos resultados. Sendo assim, é fundamental para o desenho duma interface, ter uma ideia clara sobre as características do utilizador e do tipo de tarefas que este vai desempenhar[4].

O sistema foi desenvolvido numa plataforma Windows usando o Visual Basic™ 3.0 [5], mantendo uma arquitectura aberta que permite a fácil integração de novas funções ou procedimentos no sentido de melhorar ou reformular certos aspectos da interface ou introduzir novos modelos.

Nas secções seguintes será justificada a escolha dos estilos de diálogo mais (baseada no perfil do utilizador e características das tarefas) e feita uma descrição da interface desenvolvida.

II. ESTILOS DE DIÁLOGO

Como referido anteriormente, o perfil do utilizador típico para esta aplicação e o tipo de tarefas que pretende realizar com a sua ajuda são dois parâmetros fundamentais que condicionam a filosofia utilizada no desenho duma interface.

Em particular, esta aplicação foi desenvolvida para utilizadores com o seguinte perfil:

- pouco ou médio conhecimento sobre computadores
- baixa ou média motivação
- baixa ou média experiência em computadores
- pouco ou nenhum treino
- experiência média na tarefa

Tendo em conta estas características, os estilos de diálogo mais utilizados foram a manipulação directa e os menus.

Após algum tempo de aprendizagem da interface, o utilizador deverá ser capaz de introduzir um novo problema (ou utilizar um já existente) e interagir com o sistema no sentido de encontrar as soluções pretendidas para cada situação.

III. SISTEMAS DE APOIO À DECISÃO PARA PROBLEMAS DE COBERTURA (SADPC)

A aplicação deverá permitir, através da definição correcta de um determinado problema, fazer a gestão da

(1) Trabalho desenvolvido no âmbito da disciplina de Interface Homem-Máquina.

procura interativa das possíveis soluções, sendo para isto necessário que o sistema permita:

- introduzir um problema novo
- abrir um problema já existente
- alterar os dados relativos a um problema
- resolver dois tipos de problemas: CS e CM
- gerir ficheiros

Para que o algoritmo encontre as soluções é necessária como foi referido, a definição correcta dum problema. Todas as suas características são guardadas num sistema de ficheiros gerado num formato normalizado, permitindo a posterior utilização dos dados.

O sistema oferece as funções mais comuns de gestão de ficheiros permitindo a um utilizador 'carregar' um problema já existente para memória (opção 'Project-Open'), ou um novo (opção 'Project-New'). Ao criar um novo problema o sistema apresenta uma série sequencial de *dialog boxes* onde o utilizador faz a entrada dos dados relativos ao problema em questão. Os valores introduzidos em cada *dialog box* são testados, sendo dadas mensagens de erro e directivas no caso de não serem válidos.

Depois de um novo problema estar em memória, a aplicação permite a alteração de algumas das suas características ou o *display* gráfico de apoio à busca interactiva de soluções.

IV. INTERFACE COM O UTILIZADOR

Na fig.1 apresenta-se a estrutura hierárquica que foi utilizada na interface. O desenho desta estutura baseou-se num conjunto de princípios fundamentais no projecto de uma interface com o utilizador: compatibilidade com o utilizador e com a tarefa, familiaridade, simplicidade, consistência, minimização e fácil recuperação de erros [5,6].

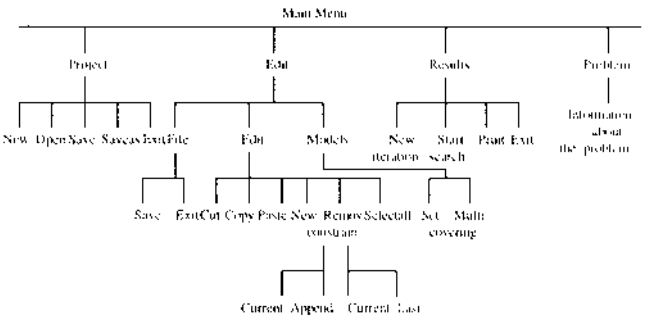


Fig.1.- Estrutura hierárquica da interface

Na fig.2 apresenta-se a janela principal, de onde o utilizador pode gerir o sistema. Nesta janela são oferecidas três opções ('Project', 'Edit', 'Results') e ainda um botão 'Problem', que ao ser pressionado faz com que se mostre no ecrã as características do problema a ser analisado.

A escolha das opções pode ser feita com o rato, estando associados a cada uma delas menus *pull-down* que surgem quando o evento *click* é detectado. Como o nome das opções sugere, 'Project' permite a criação, abertura ou a gravação em disco de problemas, 'Edit' permite aceder às características dum problema sendo possível a alteração de algumas e 'Results' é a opção na qual é feita a pesquisa de soluções, acompanhada da sua visualização gráfica.

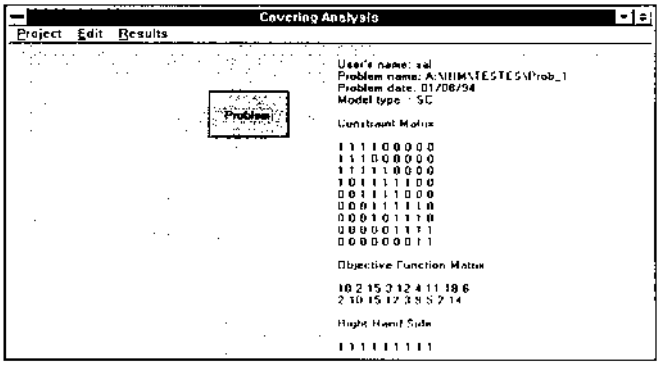


Fig.2.- Janela principal com características do problema

Faz-se, em seguida, a análise da funcionalidade mais importante oferecida pela interface.

1. *Project*

Nesta opção são oferecidas as funções mais comuns de gestão de ficheiros: 'Open', 'Save', 'Save As' e 'New'. O utilizador optará entre abrir um problema já existente, ou criar um novo; este é o primeiro passo para a utilização do sistema e corresponde à definição do problema.

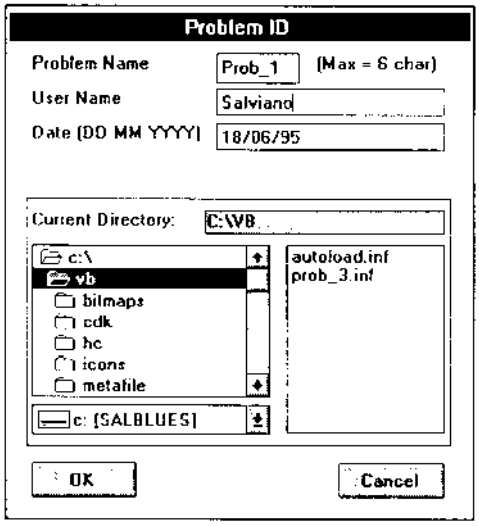


Fig.3.- Identificação do problema

Supondo que se está a utilizar pela primeira vez o sistema, como ainda não existe um problema, é necessário criá-lo. Seleccionando a opção 'Project-New', inicia-se uma sequência de *dialog boxes*, onde informação como o nome do utilizador, o nº de restrições, o nº de variáveis, as funções objectivo e os coeficientes das restrições do problema, deverá ser especificada. Muitos destes parâmetros têm gamas de validade, o que implica que os correspondentes valores de entrada devam ser testados. Se estes não forem válidos o utilizador é informado, através

de mensagens, desta ocorrência, apenas se retomando a entrada de dados quando os valores estiverem todos correctos.

Nas figuras 3 a 7, apresentam-se as janelas que permitem introduzir os dados referentes a um novo problema de Cobertura Simples ou Múltipla.

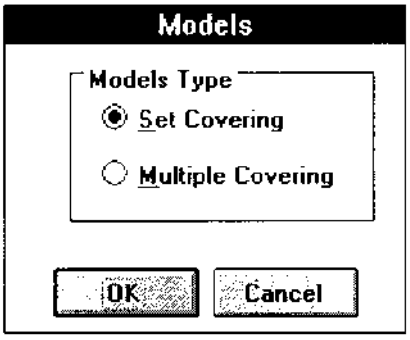


Fig.4.- Tipo de modelo

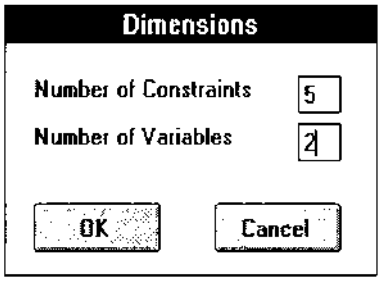


Fig.5.- Dimensões do problema

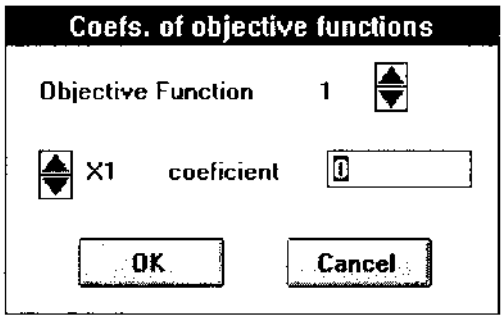


Fig.6.- Funções objectivo

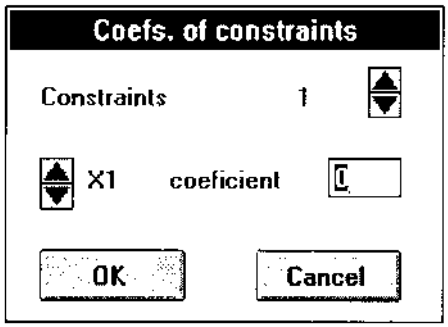


Fig.7.- Matriz das restrições

Embora as caixas de diálogo sejam apresentadas ao utilizador sequencialmente, este pode voltar à janela anterior pressionando o botão 'Cancel', se assim o desejar. Desta forma a navegação ao longo da introdução das características de um novo problema é mais simples, permitindo fazer altereções se necessário.

Depois de serem introduzidos todos os dados relativos ao problema a ser tratado, estes são guardados num conjunto de ficheiros, permitindo assim a sua reutilização.

2. *Edit*

Esta opção permite alterar os dados referentes a um problema e tem a estrutura hierárquica apresentada na fig.8, que foi usada com o objectivo de o fazer de forma eficiente (a fig. 9 mostra a correspondente janela).

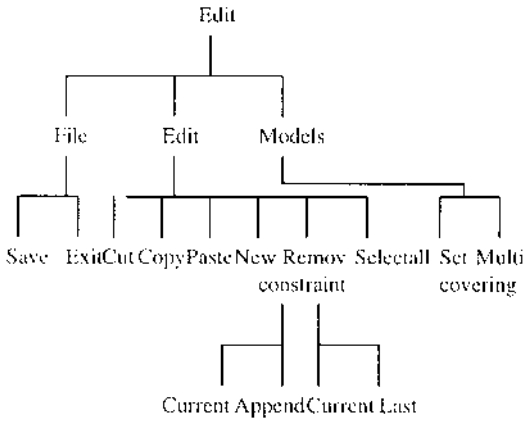


Fig.8.- Estrutura hierárquica do editor

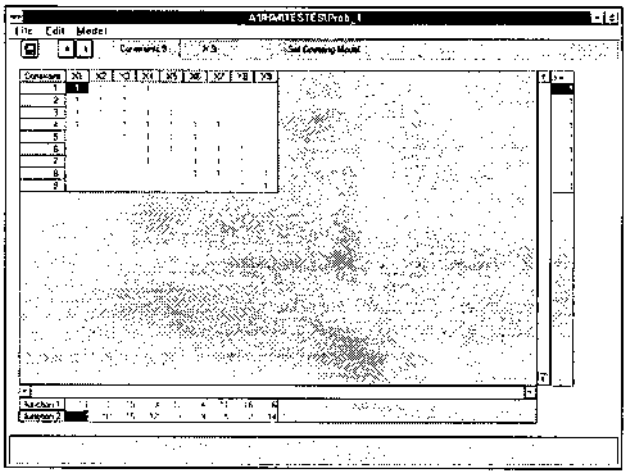


Fig.9.- Editor

3. *Results*

A fig.10 apresenta a janela correspondente a esta opção onde é mostrada ao utilizador alguma informação sobre o problema a analisar, o gráfico onde vai ser feita a pesquisa interactiva das soluções e três botões que permitem respectivamente, iniciar a pesquisa para uma determinada área, escolher uma nova área de procura e obter uma cópia

em papel. Na parte inferior aparecem os valores numéricos das soluções eficientes e o valor das correspondentes soluções não dominadas (*mapping* no espaço das funções objectivo). No exemplo em questão foi previamente seleccionada (botão 'New Iteration') uma área para procura de possíveis soluções não dominadas, aparecendo esta área a tracejado.

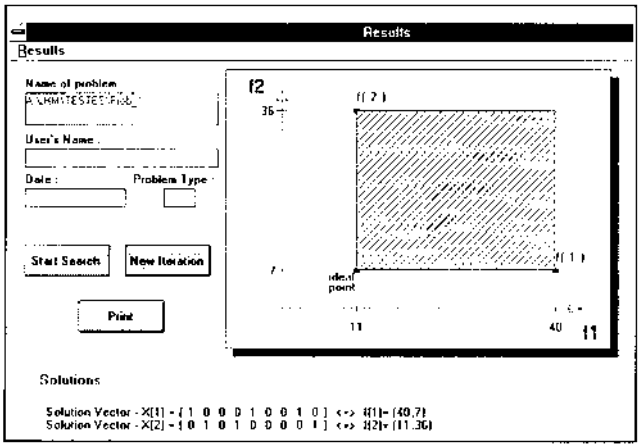


Fig.10.- Gráfico com área de pesquisa seleccionada

A procura de soluções é efectuada interactivamente do seguinte modo: à medida que vão sendo encontradas novas soluções não dominadas o utilizador vai sendo informado visualmente, através do gráfico, de áreas onde poderão existir soluções não dominadas: a branco (áreas ainda não pesquisadas) ou a tracejado (área seleccionada para pesquisa) e de áreas onde já não é possível encontrar soluções não dominadas (preenchida com pontos).

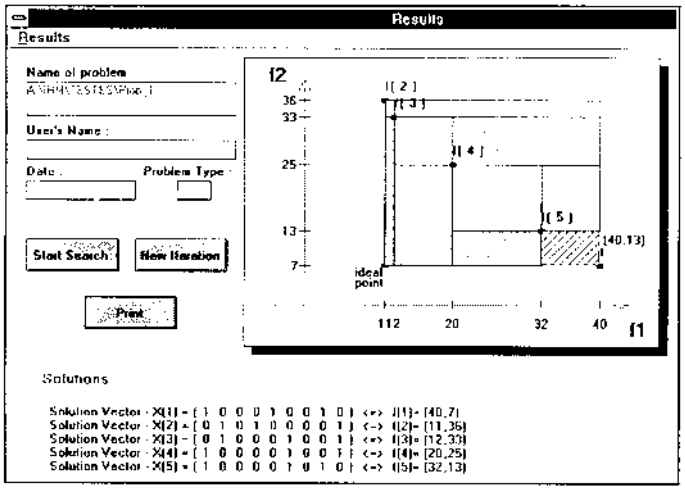


Fig.11.- Janela de resultados com várias soluções encontradas

Como se pode observar através da fig.11, depois várias iterações é possível visualizar os três tipos de áreas. Note-se que a apresentação das áreas onde já não é possível encontrar soluções não dominadas é fundamental para o utilizador, uma vez que reduz o processo exploratório, melhora a percepção em relação às alternativas do problema e permite-lhe focar a procura numa região particularmente interessante. Desta forma, em cada iteração o utilizador pode decidir, de uma forma interactiva, qual área de pesquisa de soluções, terminando o processo quando considerar que tem informação suficiente sobre o problema (não sendo obrigado à pesquisa exaustiva das soluções não dominadas).

V. CONCLUSÕES

A dificuldade do desenho de uma interface deve-se ao facto de não existir um procedimento normalizado para a sua criação. Existem apenas alguns princípios e orientações que podem fazer com que a interface se adapte com sucesso ao perfil do utilizador para o qual é criada. Assim, o conhecimento do perfil do utilizador e da tarefa, bem como a observação de princípios fundamentais como consistência, compatibilidade e protecção da interface foram, alguns dos critérios tidos em conta no desenho desta interface.

Com este trabalho foi possível por um lado, desenvolver uma interface para um sistema e por outro estudar e aplicar as técnicas de desenho e implementação de interfaces de utilizador.

O trabalho futuro deverá passar pela realização de testes à interface, com uma amostra representativa dos utilizadores, o que permitirá analisar factores importantes do bom desempenho duma interface, como por exemplo a facilidade de aprendizagem ou de navegação. Em função dos resultados obtidos e da sua análise, a interface poderá ser alterada no sentido de proporcionar uma melhor interacção com o utilizador.

REFERÊNCIAS

[1] Christofides N., Paixão J., "Algorithms for large scale set covering problems", Annals of Operations Research, 43, pp.261-277, 1993

[2] Schilling D., Vaidyanathan J., Barkhi R., "A review of covering problems in facility location", Location Science, 1, pp.25-55, 1993

[3] Carlos Ferreira, João Clímaco, José Paixão, Beatriz Sousa Santos, "On Solving Bicriteria Covering and Related Problems", XIth International Conference on Multiple Criteria Decision Making, Universidade de Coimbra, Agosto de 1994

[4] Deborah Meyhew, *Principles and Guidelines in Software User Interface Design*, Prentice Hall, 1992

[5] Microsoft Visual Basic - Language Reference - Programming System for Windows, 1991

[6] Foley J.D., van Dam A., Feiner S.K., Hughes J.F., Computer Graphics - Principles and Practice, Addison - Wesley Publishing Company, 2nd edition, 1991

Projecto Ester: Estudo de Viabilidade de um Sistema de Tipo Ground-based para Tele-Detecção Remota de Fogos Florestais

Fernando Ramos*, Carlos Borrego**, Sónia Baltazar*, Ana Miranda**

*Departamento de Electrónica e Telecomunicações, Universidade de Aveiro
**Departamento de Ambiente e Ordenamento, Universidade de Aveiro

***Resumo*- Este artigo descreve os objectivos e estratégias adoptados neste projecto, o qual é co-financiado pela JNICT/CNEFF (Projecto JNICT/CNEFF PEAM/FF/430/94). O objectivo essencial do projecto, iniciado em Janeiro de 1995 e que terminará em Dezembro de 1995, é o estudo da viabilidade de utilização na detecção de incêndios florestais de um sistema de televigilância desenvolvido para fins de segurança. O projecto inclui a construção de um protótipo laboratorial que permitirá demonstrar e validar a viabilidade prática do sistema.**

Neste artigo focam-se os vários aspectos do sistema, do ponto de vista operacional e tecnológico, e descrevem-se as opções mais significativas resultantes dos estudos já efectuados.

***Abstract*- This article describes the objectives and strategies of this project, that is co-funded by JNICT/CNEFF. The main purpose of this project is the feasibility study of the application in the remote detection of forest fires of a telesurveillance system previously developed for security applications. The project includes the setup of a laboratory demonstrator that will support the validation of the results of the studies.**

This paper describes the most significative aspects of the system and presents a set of results already achieved.

I. INTRODUÇÃO

Um dos aspectos base da política de gestão do fogo florestal consiste na rápida detecção e ataque ao fogo, o que é decisivo na minoração dos custos materiais e humanos devidos aos incêndios florestais. Em Portugal, o sistema de vigilância da floresta, durante a época normal de fogos, baseia-se numa rede de postos de vigia fixos, articulada com o patrulhamento móvel efectuado por brigadas terrestres de fiscalização, prevenção e vigilância e através de meios aéreos de vigilância. O sistema adoptado tem dado provas da sua eficácia, sendo considerável o número de focos de incêndio detectados logo na sua fase nascente.

No entanto, a rede de postos de vigia fixos implica a permanência de vigilantes, durante longos períodos de tempo, em locais cujas condições de trabalho e conforto não são as melhores, facilitando situações de monotonia e de fadiga; no entanto esta solução tem custos elevados de mão de obra. A introdução de sistemas de detecção remota nos postos de vigia poderá constituir uma alternativa aos vigilantes, ultrapassando-se, assim, os problemas identificados.

II. PROGRAMA DE TRABALHO

O programa de trabalhos deste projecto está dividido em quatro tarefas, cada uma delas com objectivos específicos e complementares:

- Tarefa 0 - Gestão do projecto. Com uma duração de doze meses, esta tarefa integra todos os aspectos relativos à gestão técnico-científica e administrativa do projecto.
- Tarefa 1 - Estudo dos aspectos operacionais. Com a duração de quatro meses, esta tarefa destina-se ao estudo da aplicação dos sistemas de detecção remota de fogos florestais de tipo ground-based em Portugal, bem como a sua articulação com sistemas de detecção de outros tipos. O resultado final integrará as especificações gerais do sistemas a desenvolver.
- Tarefa 2 - Estudo do aspectos tecnológicos. Esta tarefa têm a duração de doze meses e tem por objectivo o estudo dos aspectos tecnológicos do sistema, o que inclui a arquitectura, a configuração das ERs (estações remotas) e da EC (estação central), câmaras, sensores, comunicações e algoritmos de codificação e compressão.
- Tarefa 3 - Demonstrador. Com a duração de seis meses inclui a especificação da configuração e funcionalidades de um demonstrador que permitirá avaliar a viabilidade técnica de aplicação do sistema proposto, bem como a respectiva implementação.
- Tarefa 4 - Conclusões. Nesta tarefa serão reunidos os resultados do projecto, com o objectivo fundamental de avaliar a aplicação do sistema na detecção remota de incêndios florestais.

III. TAREFAS CONCLUÍDAS

Na data de redacção deste artigo encontra-se concluído o estudo dos aspectos operacionais. Seguidamente é apresentado um resumo dos estudos e conclusões desta tarefa.

O estudo dos aspectos operacionais realizado abordou os seguintes tópicos:

- caracterização dos aspectos do fogo florestal relacionados, directa ou indirectamente, com a sua detecção;
- análise comparativa das várias estratégias de detecção de incêndios florestais;
- identificação e descrição dos dispositivos de tele-detecção remota;
- breve abordagem do funcionamento interactivo das entidades responsáveis pela prevenção, detecção e combate ao fogo.

A.. Caracterização do fogo

Um fogo pode ser definido como a reacção química existente entre um combustível (vegetal) e um oxidante (normalmente, o ar ambiente) (fig. 1). A reacção de combustão de elementos vegetais requer a presença de uma fonte de calor para se iniciar (ignição). Uma vez estabelecida a combustão, esta liberta energia suficiente para manter a reacção mesmo que se retire a fonte de calor inicial: desde que haja combustível e oxigénio, a combustão mantém-se. O resultado dessa reacção é a libertação de energia (maioritariamente térmica) e a produção de espécies inertes e tóxicas, tais como dióxido de carbono (CO_2), monóxido de carbono (CO), vapor de água (H_2O), fuligem, fumo, etc. Frequentemente grande parte da energia libertada ocorre na forma de luz visível e radiação infravermelha.

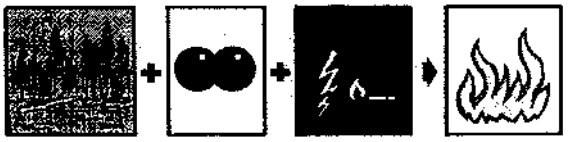


Fig. 1 - Representação da reacção química [4]

A ocorrência e o desenvolvimento de um incêndio florestal são condicionados por variados factores, nomeadamente:

- Combustíveis
- Topografia
- Meteorologia

A.1. Combustíveis

Os combustíveis são um dos parâmetros mais importantes na detecção de fogos florestais. Qualquer substância que sofra ignição e seja queimada é um combustível. No caso dos incêndios florestais, que ocorrem em regiões que abrangem zonas agrícolas, matos ou florestas, o combustível é sobretudo a vegetação. Esta é consumida rapidamente, pelo que a continuação da combustão implica a propagação a novos combustíveis adjacentes, alastrando o incêndio florestal ao longo de uma linha denominada frente de chamas, e cujo perímetro delimita a área ardida [15].

Para além dos factores meteorológicos e topográficos, a vegetação tem pois um papel decisivo na ocorrência e propagação dos incêndios. A inflamabilidade e a combustibilidade global do sistema dependem fortemente da sua estrutura e disposição espacial.

A flamabilidade é uma das características dos combustíveis que poderá ser mais significativa para a detecção, pois caracteriza a sua maior ou menor facilidade de entrar em combustão na presença de uma fonte de calor. No entanto, a inflamabilidade de um dado combustível depende consideravelmente do seu teor em humidade. As espécies serão tanto mais inflamáveis quanto menos tempo demorarem a produzir uma chama. As espécies associadas com vegetação do tipo resinoso, por exemplo, são altamente inflamáveis, propagando-se as chamas com grande intensidade.

As chamas são constituídas por gases resultantes da combustão, as quais se encontram a uma temperatura muito alta (da ordem dos 1000°C) [15]. Elas são a principal forma de libertação da energia de combustão e constituem a parte visível mais importante do incêndio.

Nos fogos florestais, os dois produtos da oxidação completa - CO_2 e H_2O - constituem cerca de 90% da massa total emitida. O vapor de água também pode resultar do processo de secagem das plantas aquecidas pela frente de chamas que se aproxima. Constitui um fumo esbranquiçado, que se observa nalgumas fases dos incêndios, sobretudo na presença de combustíveis verdes. No entanto, a combustão nos fogos florestais não é um processo quimicamente eficiente, sendo produzidos para além dos compostos referidos, os restantes 10% da massa total emitida, que incluirão os compostos do fumo potencialmente problemáticos: os hidrocarbonetos gasosos, as partículas, o CO, os ácidos orgânicos, aldeídos e óxidos de azoto [12].As partículas sólidas emitidas dão uma coloração negra aos fumos. Este facto é revelador de uma combustão rica, isto é com excesso de combustível. A quantidade de combustível disponível no terreno é pois um parâmetro importante. Varia com o local, com o tipo de combustível e com a época do ano.

A maioria dos fogos florestais, após a ignição, propaga-se em combustíveis rasteiros (herbáceas e arbustos) e do solo (folhas, raízes, ramos mortos, restos de cortes ou de limpezas). A progressão do fogo neste tipo de combustíveis, fogo rasteiro, constitui a principal forma de progressão dos incêndios e é a que se verifica quase sempre no seu início [15]. Caso as condições atmosféricas e a humidade dos combustíveis o permitam, o fogo poderá ascender até ao topo das árvores (combustíveis aéreos). Quando o fogo atinge o topo das árvores, denomina-se fogo de 'copas' (fig. 2). Este tipo de fogo poderá progredir

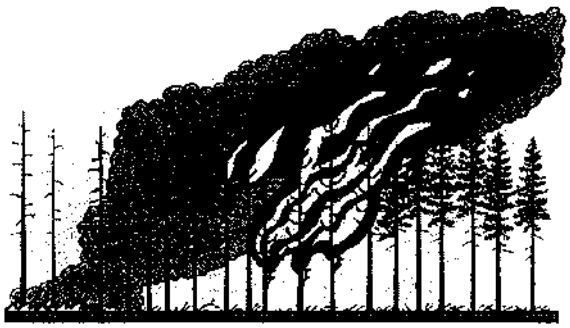


Fig. 2 - Representação de um fogo de 'copas' [4].

vários quilómetros num único dia, sendo muito perigoso e difícil de combater ou suprimir.

A.2. Topografia

A configuração do terreno tem grande importância nas condições de eclosão, propagação e combate aos incêndios florestais [15]. Em Portugal, onde a maioria dos fogos florestais ocorre em terreno de orografia complexa, este aspecto assume um carácter ainda mais relevante.

A topografia influencia consideravelmente as variações locais de temperatura superficial, devido à orientação do terreno relativamente ao Sol. As encostas viradas a Sul serão mais quentes e secas do que as encostas viradas a Norte, pelo que o risco de eclosão será maior. Para além disso, a associação usual da precipitação com ventos de uma determinada direcção, induz a que as encostas orientadas de acordo com essas direcções do vento sejam mais propícias ao desenvolvimento de vegetação durante o período das chuvas, possuindo, portanto, durante o Verão uma maior quantidade de combustível. O desenvolvimento de determinadas espécies está também condicionado pela altitude, já que esta se relaciona com a temperatura, diminuindo a temperatura à medida que aumenta a altitude.

O próprio sistema montanhoso influencia o escoamento atmosférico geral. Assim, por exemplo, as cadeias montanhosas poderão constituir barreiras efectivas ao escoamento atmosférico, enquanto que os vales, que lhes poderão estar associados, constituirão canais importantes de drenagem, estabelecendo a direcção do vento. Para além de alterarem o sistema de circulação geral, os sistemas montanhosos poderão criar o seu próprio regime de ventos. É o caso das brisas de montanha (diurnas-ascendentes e nocturnas-descendentes) que se formam devido ao aquecimento ou arrefecimento desigual do ar junto ao solo e das camadas superiores da atmosfera. As alterações provocadas pelos efeitos topográficos no escoamento atmosférico serão responsáveis por variações na direcção e intensidade do vento, podendo afectar a propagação do fogo.

A propagação do fogo depende directamente da inclinação do terreno. Comparativamente com a progressão em terreno plano, a progressão no sentido ascendente, subindo as encostas, é mais rápida, já que as chamas se aproximam dos combustíveis, facilitando assim o seu aquecimento e ignição (fig. 3). Passa-se o inverso no sentido descendente, ou seja, quando o fogo desce a montanha.

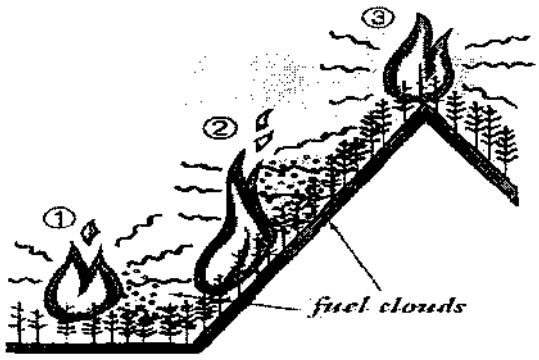


Fig. 3 - Progressão ascendente de uma frente de chamas [4].

A.3. Meteorologia

Os elementos locais da 'meteorologia do fogo' são: vento, temperatura, humidade e estabilidade atmosférica. Estas circulações de grande escala determinam os padrões regionais das alterações da 'meteorologia do fogo'.

Relativamente aos elementos locais:

- a temperatura dos combustíveis florestais e do ar ambiente é um dos factores determinantes do início e progressão de um fogo. A temperatura afecta também indirectamente o fogo, através da sua influência noutros factores que controlam a sua progressão, isto é, o vento, a humidade do combustível e a estabilidade atmosférica;
- a humidade é outro elemento importante da 'meteorologia do fogo'. Valores baixos de humidade relativa contribuem para uma maior secagem dos combustíveis florestais, constituindo um sinal de perigo;
- a estabilidade atmosférica está relacionada com a resistência da atmosfera ao movimento vertical, podendo encorajá-lo (instabilidade atmosférica) ou suprimi-lo (estabilidade atmosférica), e influenciando assim o fogo de variadas maneiras. Em condições de atmosfera instável, um pequeno impulso para cima pode produzir numa massa de ar uma corrente ascendente importante. Esta situação é muito perigosa, já que pode facilitar o desenvolvimento de grandes incêndios. O próprio calor gerado pelo fogo induz movimento vertical, pelo menos junto à superfície. Pela observação de uma coluna de fumo é possível avaliar se estamos em presença de condições instáveis, ou não. Um outro tipo de movimento vertical consiste na subsidência, que é a descida gradual de uma camada de ar sobre uma grande área. Quando se inicia em níveis elevados da troposfera, o ar, que inicialmente está pouco húmido, vai aquecendo e a sua humidade relativa vai diminuindo, à medida que se aproxima da superfície. Se esse ar quente e seco atingir a superfície poderá suceder uma séria situação de risco de fogo.

- o vento é o elemento meteorológico que, para além de ser o mais variável e menos previsível, mais afecta o comportamento do fogo. Contribui para a secagem dos combustíveis florestais, aumenta a reacção de combustão assegurando o fornecimento continuado de oxigénio, favorece o aquecimento dos combustíveis não queimados através da inclinação das chamas para diante e transporta brasas incandescentes, provocando a ignição de novos focos de incêndio a alguma distância da frente de chamas [13]. A direcção de progressão do fogo é determinada essencialmente pela direcção do vento. A figura 4 ilustra o efeito do vento na progressão de um fogo. Em Portugal as situações de maior risco de incêndio estão relacionadas com ventos de Leste, ventos geralmente muito secos e intensos.

B. Análise comparativa dos vários sistemas de detecção existentes

A detecção rápida e eficaz dos focos de incêndio tem sido, desde sempre, uma preocupação constante das equipas gestoras do combate ao fogo. [3] Fornece uma breve perspectiva histórica da detecção dos fogos florestais ao longo do tempo. Segundo este autor, uma das primeiras actividades no controlo dos fogos florestais consistiu na organização de patrulhas através da floresta, com o objectivo de detectar qualquer foco de incêndio. Os primeiros homens-patrulha atravessavam a floresta a pé e/ou a cavalo e geralmente estavam equipados para agir nos pequenos fogos que pudessem encontrar. Para além do seu papel de detecção e primeiro ataque ao fogo, estas patrulhas móveis tinham e têm, ainda hoje, um papel pedagógico e informativo, já que estão em contacto directo com a população.

Estas patrulhas deslocavam-se por caminhos estabelecidos e percorriam terras altas e picos onde a altitude lhes fornecia uma melhor visão da floresta, permitindo-lhes detectar qualquer coluna de fumo. O facto destas patrulhas descobrirem que certos locais lhes dariam uma boa visão da parte da floresta que patrulhavam levou-os à próxima etapa na evolução da detecção: a construção de postos de vigia e torres que permitiam, a um observador, vigilância da floresta. Foram construídas muitas torres, do mais variado grau de sofisticação, desde plataformas fortes e seguras no topo das árvores mais altas até torres metálicas de tecnologia avançada, com aparelhos de medida e comunicações rádio. O vigilante deverá possuir um bom conhecimento da área em que se encontra, no sentido de fornecer informações precisas e completas sobre o local de deflagração do incêndio. Actualmente, os postos de vigia fixos no terreno, instalados em locais criteriosamente escolhidos, constituem a base do sistema de vigilância da nossa floresta.

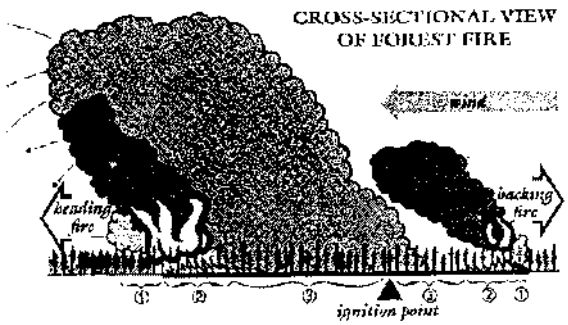


Fig. 4 - Efeito do vento na progressão de um fogo florestal [4].

Para além da detecção efectuada pelas brigadas terrestres, móveis e fixas, existem sistemas de detecção através de meios aéreos. Em Portugal, estes meios são já utilizados de uma forma sistemática nos dias de risco de incêndio não nulo. Tal como os vigilantes dos postos fixos, o pessoal encarregue da vigilância aérea deverá conhecer bem a região em que se desenrola o voo, de forma a identificar facilmente os pontos de referência observados. A eficácia da vigilância aérea não é total devido a só abranger uma área limitada em torno da zona sobrevoada, efectuando uma cobertura intermitente de um determinado local, e devido a ser difícil a utilização dos meios aéreos de vigilância durante os períodos nocturnos. Para além disso, a própria natureza dos meios empregues implica custos operacionais muito elevados.

Por outro lado, a detecção aérea permite uma maior flexibilidade, sendo possível alterar a área vigiada consoante a variação dos perigos e riscos de incêndio, e possibilitando a concentração da vigilância em zonas de risco particular como as de recreio e nos fins-de-semana. Adicionalmente, os meios aéreos permitem vigiar regiões remotas, de acesso difícil.

A detecção aérea não exclui a presença de torres fixas de vigilância e de patrulhas móveis de detecção, existindo regiões em que as acções conjugadas dos vários tipos de detecção permitem uma eficiência máxima na tarefa de detecção de fogos florestais. Em Portugal tem sido adoptada, nos dias de elevado risco de incêndio, uma solução integrando os vários tipos de sistema de detecção referidos, tentando-se assim atingir a eficácia máxima em termos de detecção de incêndios florestais.

Até muito recentemente, qualquer um dos sistemas de vigilância referidos, recorria essencialmente ao sentido da visão humana como 'instrumento de detecção', constituindo as colunas de fumo emitidas o principal indício da ocorrência de um fogo florestal. O desenvolvimento constante da electrónica permitiu a adopção de meios electrónicos como ferramenta de apoio à detecção via 'olho humano' e mesmo como substitutos dessa forma de detecção. Um dos primeiros sistemas a ser implementado baseia-se na implantação, em postos de vigia, de câmaras de televisão comandadas à distância. A Polónia é um dos Países onde a substituição dos vigilantes nas torres fixas por câmaras de televisão tem fornecido bons resultados [7]. Contudo, haverá que avaliar se a

eficácia quanto às indicações da localização de um fogo poderão ser tão precisas como as de um vigilante que conheça bem a zona que deve observar.

Ultimamente, tem-se recorrido também às imagens de satélite para detecção remota de fogos florestais [14,10,11]. Este tipo de sistema permite detectar e seguir incêndios a partir de sensores colocados em satélites. Existem diferentes sensores, tais como:

- Advanced Very High Resolution Radiometer - AVHRR. Encontra-se instalado em satélites da National Oceanic and Atmospheric Administration (NOAA). O AVHRR fornece imagens digitais em comprimento de onda do visível, infra vermelho próximo e infra vermelho. Algumas limitações deste sensor encontram-se no facto da resolução da imagem ser limitada pelo tamanho do pixel, o que torna difícil a obtenção de determinado tipo de informação de incêndios individuais (área, forma e temperatura), e na impossibilidade de monitorização de variações diurnas da queima devido á saturação do sensor.
- Visible Infrared Spin Scan Radiometer Atmospheric Sounder. O seu satélite é o Geostationary Operational Environmental Satellite. Tem uma resolução física inferior à do AVHRR.
- Detector de Alta Resolução Thematic Mapper, com vários canais espectrais, instalado nos Landsat 4 e 5, satélites da NASA. É utilizado na análise da radiação reflectida pelo solo e vegetação mas, também é sensível às emissões associadas à queima de biomassa. A limitação que se aponta a este sensor é o facto, de na presença de fumo contendo partículas ou na presença de vapor de água, a superfície poder ser *mascarada* e ocorrer a reflexão das radiações solares.

As imagens de satélite estão intimamente relacionadas com as condições meteorológicas, já que a existência de nuvens e nevoeiros afecta directamente a imagem adquirida. Assim, a identificação de um incêndio via imagem de satélite poderá não ser directa, exigindo alguma prática e estando sujeita a diversos tipos de interpretação.

As imagens de satélite constituem a maior fonte de informação disponível, num determinado instante, de um incêndio em áreas extensas. Consequentemente, a sua utilização na detecção de fogos em regiões tropicais e subtropicais, é plenamente justificada. No entanto, atendendo à resolução atingida pelos sensores remotos instalados em satélites, a sua utilização não será praticável na detecção e vigilância de incêndios no nosso País.

C. Dispositivos utilizados na detecção de fogos florestais

Qualquer sistema de prevenção, detecção e manutenção de fogos florestais deve satisfazer os seguintes requisitos operacionais [6]:

- avaliação do índice de risco e rápida localização dos fogos
- transmissão imediata de sinais de alarme para as entidades competentes
- previsão temporal e espacial da evolução do fogo
- fazer o 'display' para o operador do plano de intervenção
- monitorar a evolução do fogo.

Uma das tarefas de grande importância no sistema referido consiste na rápida detecção de focos de incêndio, e na sua localização precisa, de modo a apoiar ao máximo a definição de estratégia de combate ao fogo.

O objectivo final deste projecto consiste na avaliação da viabilidade de introdução de um sistema de detecção remota/automática nos postos de vigia fixos. Quando o sistema de detecção recorre a dispositivos de detecção automática, tal como se pretende no âmbito deste trabalho, as características de um incêndio que mais poderão contribuir para uma detecção rápida e eficaz são: (i) o aumento da temperatura, associado ao aparecimento de chama, no local onde se dá a eclosão; e (ii) o fumo e alguns gases que são libertados aquando da combustão da biomassa.

Para além das características do fogo que poderão permitir a sua detecção, o próprio sistema de vigilância dependerá do risco de incêndio associado a cada local, ao longo do tempo. A determinação do índice de risco meteorológico baseia-se em dados meteorológicos, sendo assim importante conhecer as condições meteorológicas do local que se vigia. A implantação de uma rede meteorológica automática associada aos postos fixos de vigia contribuirá para a definição de índices de risco, permitindo avaliar se a região em análise constituirá, ou não, uma zona de grande probabilidade de ocorrência de incêndio, e portanto se deverá ser sujeita a vigilância especial por parte das entidades responsáveis.

C.1. Fumo

As colunas de fumo emitidas no decorrer de um incêndio florestal têm sido desde sempre um dos principais sinais de eclosão e ocorrência de fogo. Tal como já foi referido na introdução, o primeiro dispositivo de detecção automática de fogos florestais baseou-se na utilização de câmaras de televisão, colocadas em postos de vigia fixos, que permitiam a visualização das colunas de fumo. Testes realizados com uma câmara de televisão industrial, com movimentos horizontais e verticais, controlada remotamente por um observador, revelaram que poderiam ser detectadas colunas de fumo emitidas a distâncias superiores a 20 km [3].

O facto destas câmaras detectarem essencialmente as colunas de fumo, torna-as dependentes, em termos de

eficácia, dos factores que afectam a visibilidade. No caso de existir nevoeiro, neblina ou fumo entre o ponto de observação e o fumo resultante do incêndio, a detecção será bastante dificultada, podendo mesmo vir a ser inviável.

Em muitos casos os fogos começam por baixo das árvores e arbustos, isto é, surgem escondidos da vista directa dos sensores montados num sistema *groundbased*. Medidas utilizando a gama de infravermelhos média-alta só detectam fogo quando este pode ser observado directamente através da chama, logo facilmente propagável.

Partindo desta perspectiva parece óbvio que se tente detectar o fogo quando ele ainda se apresenta rasteiro, isto é na sua fase inicial. Assim foi desenvolvido um projecto - Multi-Spectral Autonomous WildFire Dtection System - no âmbito de um programa de riscos naturais e climatologia da Direcção Geral para a Ciência, Investigação e Tecnologia da Comissão da União Europeia, onde foram utilizadas câmaras com tecnologia CCPD (*charge coupled photodiode device*), com um sistema adicional de filtros para equipar a câmara com capacidade de cor [16]. Este sistema é sensível na gama visível e baixa de infravermelho do espectro.

De acordo com [11] o único tipo de sensores remotos capazes de penetrar em nuvens consiste em radares activos. No entanto, as imagens que eles originam são muito difíceis de introduzir nos sistemas de transmissão, o que leva a que a sua aplicação prática na detecção de fogos não seja ainda uma realidade.

C.2. Temperatura

A eclosão de um fogo é acompanhada de um aumento de temperatura, já que a reacção de combustão produz energia (maioritariamente térmica), sob a forma de luz visível e de radiação infravermelha (I.V.).

Durante e após a II Guerra Mundial foram desenvolvidos, com objectivos militares, instrumentos electrónicos para detecção da radiação I.V. [3]. Na década de 50, alguns cientistas começaram a explorar o potencial da detecção por I.V. para outros campos que não o militar, por exemplo para aplicações médicas e para detecção de fogos.

A base de desenvolvimento de dispositivos de detecção de fogos através de radiação I.V. assenta na relação directa entre a temperatura de um objecto e o comprimento de onda ao qual este irradia ondas electromagnéticas. Assim, um dispositivo electrónico sensível às ondas electromagnéticas emitidas por um fogo nascente poderá ser utilizado na detecção dessa fonte de radiação, contra um fundo que seja floresta ou outro terreno mais frio.

A gama de I.V. engloba a porção do espectro electromagnético de aproximadamente 1 a 1000 microns, mas a maior parte das ondas I.V. detectadas a partir de um fogo variam entre 1 e 5 microns [3]. Contudo, da energia emitida pelo fogo na forma de ondas electromagnéticas que atravessam a atmosfera até atingir o instrumento detector, apenas estreitas bandas das ondas I.V. penetram realmente a atmosfera. Estas estreitas bandas são denominadas janelas.

Assim, um detector efectivo de fogos florestais, terá que reagir a comprimentos de onda no intervalo da energia máxima emitida pelo fogo, mas também a comprimentos de onda que coincidam com a janela. Estudos efectuados com esse objectivo permitiram determinar que a janela mais apropriada para um detector de I.V., corresponde a comprimentos de onda de cerca de 5 mícron [3].

Um sistema electrónico para detecção de fogo através de radiação I.V. deverá ser constituído por [3] :

- lentes ópticas ou sistema de reflectores e lentes através das quais a energia infravermelha emitida pelo terreno observado possa ser focada (opcional)
- célula de detecção infravermelha. Esta célula deverá converter a energia eléctrica em impulsos eléctricos.

A célula de detecção infravermelha pode ser uma câmara de infravermelhos, que fornece as variações de temperatura sobre a forma de imagens, possibilitando a detecção de estradas, rios, lagos, etc., contribuindo assim para a localização exacta do fogo. Os detectores infravermelhos, lineares ou bidimensionais, utilizados nas câmaras de infravermelhos são geralmente construídos em matérias como: silício, germânio índio antimónio, mercúrio cádmio telúrio, índio antimónio, platina, índio gálio.

O facto de as câmaras infravermelhas representarem a variação de temperatura de uma 'cena', permitiu a sua adaptação para visualização nocturna numa perspectiva "black-hot". É pois, perfeitamente possível observar uma 'cena' nocturna, já que as diferentes tonalidades da imagem obtida permitem a identificação dos objectos presentes. Uma câmara I.V., para funcionar em visão nocturna, necessita de pelo menos 256x256 pixeis, de uma sensibilidade de 0,3% e de operar a um 'frame rate' próximo da TV (30 Hz) [9.].

A célula de detecção infravermelha pode também ser um simples detector, que apresenta à saída um sinal relacionado com a radiação incidente. Estes sensores, simples, não são mais do que sensores remotos de temperatura.

A detecção de fogos através da radiação I.V. apresenta obviamente vantagens e desvantagens. A maior vantagem dos infravermelhos é a sua capacidade de penetrarem o fumo, o que permite a detecção de pequenos focos de incêndio entre o fumo de outros fogos que estejam a decorrer na zona.

Uma das maiores desvantagens da detecção por I.V. advém da impossibilidade das ondas I.V. penetrarem ar com vapor de água, ficando assim a detecção do fogo restringida pelo nevoeiro, neblinas e nuvens.

C.3. Gases libertados na combustão

Um outro sistema passível de utilização na vigilância de ocorrência de incêndios florestais, baseia-se na detecção electrónica de um determinado gás libertado durante um fogo. O interferómetro [3] permite a identificação de gases em pequenas quantidades na atmosfera. A sua aplicação à área da prevenção e detecção dos incêndios florestais, através da detecção de um determinado gás emitido pelo fogo, parece viável. No entanto, a dificuldade em encontrar um gás unicamente emitido pelos fogos, e portanto ausente da atmosfera na também ausência de fogo, e a dificuldade em discernir se o gás detectado provém ou não de fogos já existentes na região, constituem problemas, na utilização deste sistema.

O laser pode também ser utilizado para a detecção de um gás libertado na estado de precombustão do fogo, isto é, mesmo antes de ser visível qualquer chama. A utilização do LIDAR (Light Detection And Ranging) na detecção remota dos fogos florestais é uma hipótese em estudo [1,5].

D. Interacção com os sistemas de combate ao fogo

O presente projecto visa analisar a viabilidade da introdução de um sistema de detecção remota de incêndios florestais nos postos de vigia fixos. Como tal, a informação sobre a gestão do combate ao fogo em Portugal, é essencial para a correcta interacção desse sistema com o sistema de gestão do fogo existente.

O combate aos incêndios é efectuado pelos bombeiros, pelo pessoal dos serviços florestais, entidades privadas, populares, entidades militares e Centros de Coordenação de meios aéreos. A coordenação entre as acções aéreas e terrestres deverá ser a melhor possível, de forma a garantir um combate ao incêndio eficaz.

Ao nível Concelhio ou Distrital existem as Comissões Especializadas de Fogos Florestais (CEFF), que reúnem representantes das diversas entidades envolvidas. Estas comissões são presididas pelo Presidente da Câmara ou pelo governador Civil, consoante o nível a que são formadas.

A cooperação entre as diferentes entidades deverá ser feita frequentemente, em particular no Inverno, para conhecimento das chefias, instalações, equipamentos, métodos e âmbito de actuação de cada entidade, contribuindo assim para que na época de maior risco de incêndios o combate seja mais eficiente.

A fig. 5 apresenta um esquema do funcionamento integrado das várias entidades que, em Portugal, colaboram no combate ao fogo.

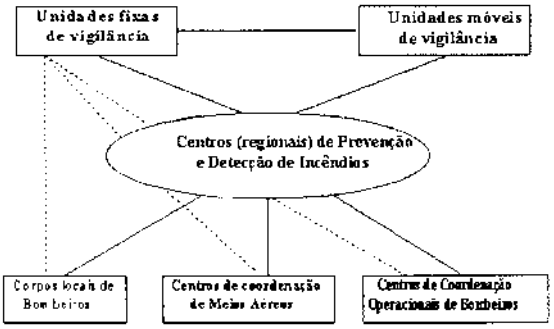


Fig. 5 - Esquema de funcionamento da rede de combate ao fogo florestal em Portugal.

No caso de detecção de um incêndio pelas unidades de vigilância, esse deverá ser comunicado prontamente, via rádio, aos Centros de Prevenção e Detecção de Incêndios (rede da Direcção Geral das Florestas), existentes em cada região. A estes centros afluem todas as comunicações recebidas tanto dos postos de vigia como das patrulhas terrestres. A informação será então transmitida aos Corpos de Bombeiros e aos Centros de Coordenação de Meios Aéreos. O contacto entre as diferentes entidades intervenientes deverá ser sempre permanente.

Em algumas regiões, os postos de vigia podem entrar em contacto directo com os Centros de Coordenação Operacionais de Bombeiros, com os Centros de Coordenação dos Meios Aéreos ou com o Corpo de Bombeiros local. Os postos de vigia fixos e as unidades móveis deverão ainda estar em comunicação constante, via rádio.

IV. DEMONSTRADOR

O objectivo deste projecto, como foi já referido, é o estudo de viabilidade técnica, económica e operacional da aplicação na tele-detecção remota de fogos florestais de um sistema de vigilância remota, desenvolvido para aplicações de segurança[8]. Este sistema para fins de segurança é baseado numa Estação Central de Vigilância (EC) e num conjunto de Estações Remotas de Vigilância (ERs), sendo ambos os tipos de estações suportadas em plataformas computacionais de tipo PC compatível (fig. 6).

Cada ER pode, na actual configuração, suportar a ligação de até seis câmaras vídeo preto e branco, ou duas câmaras RGB, ou uma câmara RGB e três câmaras a preto e branco, e dispõe ainda da possibilidade de permitir a ligação de um conjunto de sensores e actuadores em número somente limitado pelo número de cartas de I/O instaladas em cada ER. O sistema permite a transferência de imagem de uma câmara instalada numa ER para a EC, podendo essa transferência ser de imagens a preto e branco ou a cores consoante o tipo de câmara. No caso das imagens a preto e branco a transferência é efectuada em modo imagem móvel, com uma resolução de 180x144 pixels e uma cadência de 8 imagens/seg, sendo

utilizado um algoritmo de codificação proprietário. No caso das imagens a cores a transferência e efectuada em modo imagem fixa, com uma resolução de 360x288 pixels e uma taxa de transferência de 1 imagem por cada 2 seg. ,sendo as imagens codificadas de acordo com a norma JPEG.

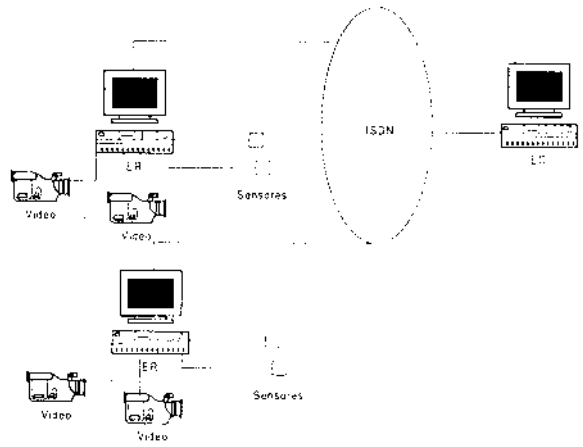


Fig.6 - Arquitectura do sistema desenvolvido para aplicações de segurança

O estabelecimento de uma comunicação entre a EC e uma ER pode ser da iniciativa quer da EC quer da ER. No caso de ser a EC a estabelecer a comunicação, isso resulta de uma acção explícita do respectivo operador, que deverá seleccionar de qual das câmaras existentes na ER pretende obter imagens. No caso da iniciativa ser de uma ER, isso resulta da activação de um dos sensores ligados à ER, sendo, portanto, consequência de se ter registado uma situação de alarme. Durante o processo de estabelecimento de comunicação por iniciativa de uma Er são transferidos para a EC dados que identificam a ER e o sensor activado, bem como a data e a hora de ocorrência do alarme. Logo que a comunicação esteja estabelecida inicia-se o processo de transferência e visualização da imagens, que serão recolhidas da câmara com a qual, por configuração prévia esteja relacionado o sensor. Neste sistema a comunicação entre EC e as ERs é efectuada através da RDIS, sendo suportada em canais de 64 kbit/s em modo circuito. Na unidade de demonstração a elaborar deverão ser mantidas, para já, todas as características do sistema de vigilância, excepto o meio de comunicação utilizado.

Após o estudo dos aspectos operacionais conclui-se que os dispositivos mais aconselháveis para detecção de fogos florestais são os por infravermelhos, câmaras ou simples detectores. A primeira solução é, no entanto, extremamente cara (uma câmara I.V. custa vários milhares de contos)

A flexibilidade proporcionada pelas diferentes soluções ao nível das comunicação influenciam a arquitectura geral do sistema que poderá ser constituída por uma estrutura multi-nível do tipo da apresentada na fig.7

Assim numa fase futura, o sistema, será composto ao mais baixo nível por um conjunto de ERs que poderão comunicar com um concentrador de zona através de ligações RF, cabo ou GSM. Estes concentradores de zona poderão dispor de ligações via satélite ou através de uma rede terrestre (por exemplo RDIS) que darão acesso, ou directa ou através de concentradores de região, a uma das

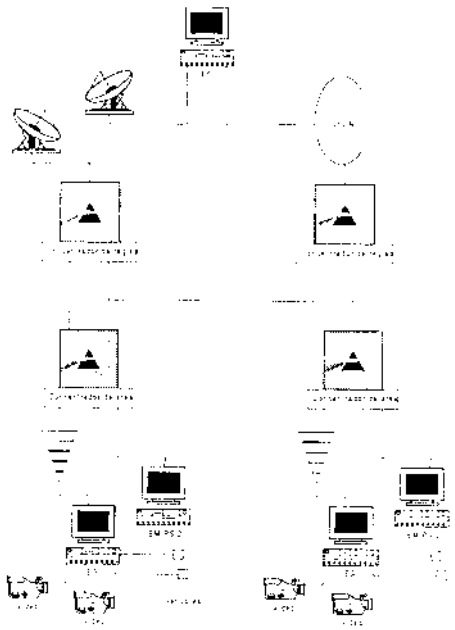


Fig.7 - Arquitectura genérica

ECs existentes. Nessa fase deverá também contar, na EC, com um sistema integrado SIG (Sistemas de Informação Geográfica) para complementar e melhorar a informação fornecida às entidades responsáveis pelo combate ao fogo.

O demonstrador (fig.8) será constituído por uma EC e por uma ER, com comunicações de tipo RDIS e RF, estando neste momento também em estudo a possibilidade de utilização de GSM. O orçamento disponível não permite, como era desejável, a inclusão de sensores de I.V., sendo a situação de alarme simulada na ER.

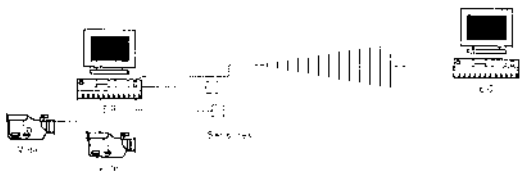


Fig. 8 - Arquitectura do demonstrador

V. CONCLUSÕES

As únicas conclusões possíveis neste momento dizem respeito essencialmente ao estudo efectuado e relacionado com os aspectos operacionais, e a alguns aspectos relacionados com as comunicações.

A primeira grande conclusão diz respeito aos dispositivos a utilizar na detecção: utilizar dispositivos infravermelhos para detecção de fogos florestais, câmaras ou detectores

simples. A segunda que se pode tirar diz respeito aos meios de comunicação: ao contrário do que inicialmente se pensava, utilizar comunicações por cabo não é uma ideia totalmente desprovida de sentido. Existem sistemas baseados neste tipo de solução, caso do SRI-10 Forest Fire Detection Systems produzido pelas empresas ALINEA e ARINC Research Corporation Este sistema está em experiência na Florida, USA, estado em que as cerca de 900 torres de detecção de fogos instaladas têm ligações telefónicas.

Por último, de facto não existe o sistema perfeito de detecção de fogos; no entanto um sistema deste tipo - ground-based - encarado como complementar dos outros sistemas existentes pode ser um instrumento poderoso na detecção de fogos florestais, nomeadamente em Portugal

REFERÊNCIAS

[1] Banta, R; Oliver, L.D. e Holloway, E.T. 1991 - Doppler Lidar Observations of a Rotational Convective Smoke Column. In Proceedings of the 11th Conference on Fire & Forest Meteorology. Missoula, Montana.

[2] Cahoon, D.; Levine, J.; Cofer, W.; Miller, J.; Munns, P.; Tennille, M.; Yip, T.; Stocks, B.; Heck, P. 1991. The Great Chinese Fire of 1987. A view from Space. In *Global Biomass Burning. Atmospheric, Climatic, and Biospheric Implications*. edited by Joel S. Levine. The MIT Press

[3] Chandler, C., Cheney, P., Thomas, P., Trabaud, L. & Williams, D. 1991 *Fire in Forestry, Volume II Forest Fire Management and Organization*. Krieger Publishing Company. Malabar, Florida.

[4] Cottrell, W. 1989 *The Book of Fire*. Mountain Press Publishing Company.

[5] Easthouse, K. 1994 - *Laser smoke detector*. In Santa Fe New Mexican, June.

[6] Fuschetti, F.- Automatic System for Forest Fire Detection in Campania - ItalyVol.II, CP.26, pp769-778, Coimbra, Nov. 1994

[7] Karlikowski, T. 1981. Systems for forest fire detection. Proceeding of the Forest Fire Prevention and Control. Warsaw

[8] J.Santos, F.Ramos, O.Santos, Sistemas de Televigilância suportado na RDIS, revista do Departamento de Electrónica e Telecomunicações, vol I, nº2, Setembro 94

[9] Laser Focus World: Detector Handbook - Imaging Systems. Active-pixel sensors, multiplexers, thermography, night vision, thermal imaging, . June 1993

[10] Malingreau, J. P. 1990. The Contribution of Remote Sensing to the Global Monitoring of Fires in Tropical and Subtropical Ecosystems. In *Fire in the Tropical Biota. Ecosystems Processes and Global Challenges*. edited by Goldammer. Springer-Verlag

[11] Robinson, J. 1991. Problems in Global Fire Evaluation: is Remote Sensing the Solution? In *Global Biomass Burning. Atmospheric, Climatic, and Biospheric Implications*. edited by Joel S. Levine. The MIT Press

[12] Ryan e McMahon. 1976. Some chemical and Physical characteristics od Emissions from Forest Fires. *Proceedings of the 69th Annual Meeting of the Air Pollution Control association*. Portland. Oregan. USA.

[13] Schroeder, M. e Buck, C. 1970. *Fire Weather. A Guide for Application of Meteorological Information to Forest Fire Control Operations*. U.S. Department of Agriculture. Forest Service. Agriculture Handbook 360.

[14] Stephens, G. & Matson, M. 1989. Fire Detection Using NOAA-N Satellites. In *Proceedings of the 10th Conference on Fire & Forest Meteorology*. edited by Maciver, Auld & Whitewood.

[15] Viegas, D. 1989. *Manual sobre Incêndios Florestais*. Secretaria-Geral do Ministério do Planeamento e da Administração do Território.

[16] Vries, J. S., Kemp, R. A. W. 1994- Results with a Multi-Spectral Autonomous Wildfire Detection System.. Proceedings of the 2nd International Conference on *Forest Fire Research*. Vol II. CP 779-791. Coimbra

Modelo de Integração de Aplicações Distribuídas e não Colaborantes

Rui Pedro Lopes, António Miguel Esteves, José Luís Oliveira, Joaquim Arnaldo Martins

Resumo- **Este artigo descreve um modelo, para o desenvolvimento de aplicações distribuídas, que permite a divisão de uma aplicação em diversos módulos de processamento. Serão igualmente apresentadas, as potencialidades deste modelo para integrar aplicações autónomas e não colaborantes. Apesar da generalidade do sistema proposto a sua aplicabilidade será discutida no âmbito de um projecto de um Sistema de Gestão de Redes.**

Abstract- **This paper describes a model that enables the development of distributed applications and provides an interface for a transparent integration of standalone packages. This model is presented in the context of a Network Management System scenario.**

I. INTRODUÇÃO*

O desenvolvimento de aplicações é, cada vez mais, condicionado por um conjunto crescente de requisitos, relacionados com o aumento de funcionalidade, com a necessidade de servir uma maior leque de sistemas e, ainda, com o aumento de desempenho. Estas solicitações, embora nem sempre sejam conciliáveis (desempenho *versus* funcionalidade), são, normalmente, prioritárias na estratégia de desenvolvimento, pelo que é importante encontrar soluções que permitam a sua optimização. A divisão da aplicação em módulos e a sua distribuição por diferentes processos concorrentes, é uma das técnicas que permite atingir este objectivo.

Esta sub-divisão pode resultar de uma política explícita no planeamento do sistema ou, pelo contrário, ser consequência da utilização de aplicações autónomas, desenvolvidas sem qualquer mecanismo que permita a partilha directa com outras aplicações. A proliferação, em domínio público, de aplicações que fornecem soluções parciais para determinados problemas, aumenta significativamente a quota deste segundo tipo de distribuição. A integração e a reutilização, numa aplicação, da informação fornecida por estes programas/módulos, assume-se como uma tarefa de especial importância.

O trabalho apresentado ao longo deste artigo foi polarizado segundo estes dois objectivos: a distribuição de processos e aplicações; e a integração de aplicações não colaborantes. Como primeira abordagem são apresentadas algumas técnicas de comunicação entre processos (locais ou remotos), especialmente o método de passagem de mensagens (*IPC message queues*) e o mecanismo de RPC (*Remote Procedure Call*). Será apresentado um modelo que permite utilizar, de uma forma transparente para o utilizador, os dois sistemas em simultâneo e que se baseia no conceito de comutador de aplicações. Será enquadrado, igualmente, um cenário que rentabiliza e justifica a utilização deste modelo (gestão de redes de dados).

II. TÉCNICAS GERAIS DE COMUNICAÇÃO ENTRE PROCESSOS

A partilha de dados entre diferentes procedimentos (*functions*), de uma dada aplicação, pode ser feita utilizando variáveis globais. Este método é, no entanto, pouco "elegante" pois conduz a códigos menos claros e mais sujeito a erros. Uma alternativa a esta solução é dada pelas chamadas a funções, ao permitirem a passagem de argumentos e o retorno de resultados.

Estas soluções, embora funcionais para ambientes monoprocesso, tornam-se ineficazes sempre que se pretende trocar informação entre diferentes processos (cenários multiprocesso).

Devido a esta limitação, foram sendo desenvolvidos, e integrados em alguns sistemas operativos, vários métodos de comunicação entre processos, ou IPC (*InterProcess Communication*). Alguns dos métodos são típicos do sistema UNIX, como por exemplo: *pipes*, *fifos* (ou *named pipes*), passagem de mensagens (*message queues*) e memória partilhada (*shared memory*), semáforos (*semaphores*) [1,2,3]. Todos estas técnicas utilizam o *Kernel* do sistema operativo como coordenador de informação (Fig. 1), apesar de não ser obrigatório para a comunicação entre processos.

Estes métodos apresentam algumas similaridades e são resultado, essencialmente, de diferentes soluções técnicas. Contudo, alguns estão limitados a relações tipo *parent-child* (*pipes*, *fifos*) enquanto que outros são aplicáveis

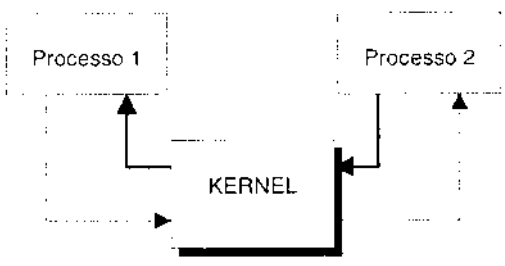


Fig. 1 - IPC entre dois processos num mesmo sistema.

* Trabalho realizado no âmbito da disciplina de Projecto.

sobre processos independentes (*message queues*, *shared memory*, *semaphores*).

Qualquer um destes métodos soluciona o problema de comunicação entre processos num mesmo sistema. A comunicação entre processos em diferentes sistemas coloca, no entanto, outro tipo de condicionalismos que não são resolvidos por estes. O mecanismo de RPC (*Remote Procedure Call*) é um dos métodos que permitem solucionar este tipo de problemas.

Na descrição seguinte serão apresentados os métodos de passagem de mensagens e de RPC.

A. IPC - Passagem de Mensagens

A passagem de mensagens constitui, juntamente com a memória partilhada e os semáforos, o *System V IPC*. O conceito principal deste sistema é a troca de mensagens (sob a forma de uma estrutura de dados) colocadas numa pilha, que é identificada por uma chave própria. A chave (tipo *key_t*, inteiro de 32 bits) pode ser obtida por três métodos:

- usar uma chave privada (IPC_PRIVATE) obrigando à troca do identificador, gerado pela chave, através de um ficheiro;
- pré-acordada entre o cliente e o servidor (estática e acessível no código);
- criada (*ftok*) a partir de um *pathname* e um valor entre 0 e 255.

Uma vez na posse da chave, que irá servir para gerar o espaço comum de passagem de mensagens, os processos podem inicializar as estruturas necessárias e desencadear o processo de comunicação.

O sistema de desenvolvimento disponibiliza um conjunto de primitivas para operação sobre mensagens (Tabela 1).

Uma pilha de mensagens é criada, ou uma já existente é acedida, usando a primitiva *msgget*:

```
int msgget(key_t key, int flag);
```

O valor *flag* indica a acessibilidade da pilha (semelhante ao sistema de proteção de ficheiros, no sistema UNIX).

Quando uma pilha de mensagens é criada, é seguida uma sequência de regras, tendo em conta o estado do sistema e as *flags* disponibilizadas à primitiva de criação. Estas regras podem ser resumidas (Fig. 2):

- Especificando o bit de IPC_CREAT da flag é criado um novo canal de passagem de mensagens para a chave especificada, caso ainda não exista. Se já se encontrar um com a chave especificada, este é referenciado.
- Especificando os bits de IPC_CREAT e de IPC_EXCL na flag, é criado um canal de passagem de mensagens baseado na chave especificada, apenas se ainda não existir. Se já existir uma entrada com a chave indicada há ocorrência de erro.
- Especificando IPC_EXCL sem IPC_CREAT não tem qualquer sentido.

Tabela 1 - Resumo de primitivas utilizadas na passagem de mensagens.

Primitivas	Descrição
msgget	primitiva de criação ou abertura
msgctl	primitiva de controlo de operações
msgsnd	primitiva de operação de escrita
msgrcv	primitiva de operação de leitura

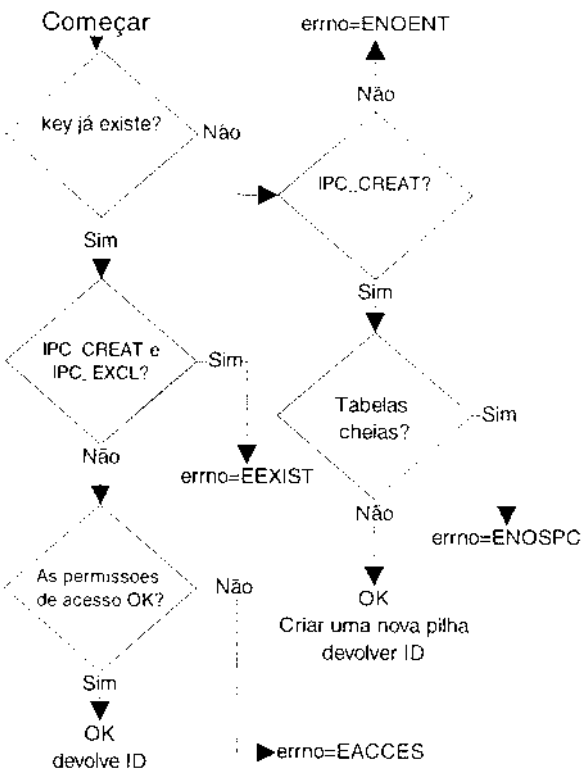


Fig. 2 - Fluxograma para o funcionamento de *msgget*.

Todas as mensagens são armazenadas pelo *Kernel* que utilizam um identificador, *msqid*, para referenciar uma determinada pilha. A forma de armazenamento assenta numa lista ligada, com uma estrutura de dados semelhante à seguinte, para cada pilha:

```
struct msqid_ds {
    struct ipc_perm msg_perm;
        /* Estrutura de permissões */
    struct msg *msg_first;
        /* Apontador para a 1. mensagem */
    struct msg *msg_last;
        /* Apontador para a última mensagem */
    ushort msg_cbytes;
        /* # de bytes existentes na pilha */
    ushort msg_qnum;
        /* # de mensagens na pilha */
    ushort msg_qbytes;
        /* # máximo de bytes da pilha */
    ushort msg_lspid;
        /* pid do último msgsnd */
    ushort msg_lrpid;
        /* pid do último msgrcv */
    time_t msg_stime;
        /* Hora do último msgsnd */
```

```
    time_t msg_rtime;
        /* Hora do último msgrcv */
    time_t msg_ctime;
        /* Hora do último msgctl */
};
```

Cada mensagem da pilha (Fig. 3) possui os atributos seguintes:

- *type* - inteiro tipo *long*;
- *length* - comprimento do conteúdo de dados da mensagem em bytes (pode ser zero);
- *data* - dados transmitidos (se *length* for maior que zero).

Uma vez aberto o canal de comunicação (criada a lista ligada com zero entradas), as mensagens podem ser colocadas na pilha utilizando a primitiva *msgsnd*:

```
int msgsnd(int msqid, struct msgbuf *ptr,
           int length, int flag);
```

O argumento *ptr* é um ponteiro para uma estrutura (definida em <sys/msg.h>) com a seguinte base:

```
struct msgbuf {
    long type;     /* Tipo da mensagem (>0) */
    char mtext[ ]; /* Dados */
};
```

O único campo com significado para o gestor de mensagens do *Kernel* e, portanto, de existência obrigatória, é o tipo da mensagem (*type*). O outro campo não tem qualquer tipo de restrição.

O argumento *length* (de *msgsnd*) especifica o comprimento da mensagem, excluindo o campo *type*.

O argumento *flag* pode ser IPC_NOWAIT ou zero. O valor IPC_NOWAIT permite que esta primitiva se torne não bloqueante, com devolução imediata se não houver espaço na pilha para a nova mensagem. Se não for indicado o valor IPC_NOWAIT a primitiva obriga o processo a bloquear até conseguir colocar a mensagem na lista ligada.

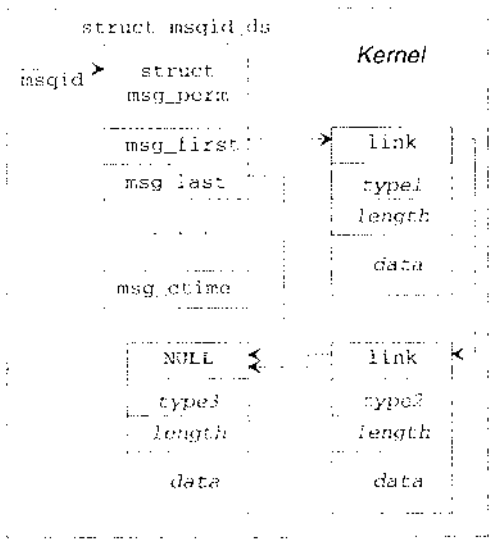


Fig. 3 Pilha de mensagens existentes no *Kernel*.

Uma mensagem é retirada da lista usando a primitiva *msgrcv*:

```
int msgrcv(int msqid, struct msgbuf *ptr,
           int length, long msgtype, int flag);
```

Para além de alguns elementos comuns à operação de escrita, o argumento *msgtype* permite especificar qual a mensagem da pilha que se pretende retirar. Se for igual a zero é adoptada a filosofia FIFO (*First In, First Out*).

O valor de retorno de *msgrcv* representa o número de *bytes* de dados na mensagem (ptr->mtext).

Para remover a pilha de mensagens (canal de comunicação) do *Kernel* é utilizada a primitiva *msgctl*:

```
int msgctl(int msqid, int cmd,
           struct msqid_ds *buf);
```

Para o efeito, *cmd* terá de ser indicado como IPC_RMID.

B. Remote Procedure Call (RPC)

O mecanismo RPC [4] é, basicamente, um processo que permite a uma aplicação em execução num determinado computador, a utilização de procedimentos que são executados noutro computador, interligado fisicamente com o primeiro. A execução desse procedimento remoto processa-se como se ele fosse local, através da passagem de parâmetros e da recolha dos valores de retorno.

O conceito de mecanismo RPC é a base para um outro conceito mais alargado, que designaremos de sistema RPC, que compreende o conjunto das ferramentas e dispositivos que constituem a base de trabalho para a construção de aplicações baseadas em mecanismos RPC. Um sistema RPC, inclui, vulgarmente, um conjunto de funções para definição do protocolo de comunicação, um compilador de protocolo e ainda funções que permitem a gestão da comunicação.

A principal vantagem deste tipo de sistema é o facto de proporcionar o desenvolvimento de aplicações distribuídas, num conjunto de computadores, sem que seja necessário considerar os aspectos respeitantes à utilização da rede e outros relacionados com a incompatibilidade entre sistemas.

Os RPC baseiam-se no modelo cliente-servidor, em que o programa que faz o pedido de execução do procedimento remoto é o cliente e o programa que proporciona a execução desse procedimento é o servidor. O cliente e o servidor não comunicam directamente entre si, mas através de processos intermediários que gerem a comunicação. A estes processos intermediários dá-se o nome de *stubs* e têm por função gerir a interacção dos sistemas ou seja, a construção, a formatação e a troca das mensagens, que circulam entre o cliente e o servidor através do protocolo RPC (Fig. 4). Os *stubs* utilizam filtros que convertem os tipos de dados usados em cada sistema, em tipos reconhecíveis por ambos, resolvendo eventuais incompatibilidades na sua representação.

Consegue-se, portanto, que, do ponto de vista do programador, a utilização de procedimentos locais ou remotos seja virtualmente semelhante.

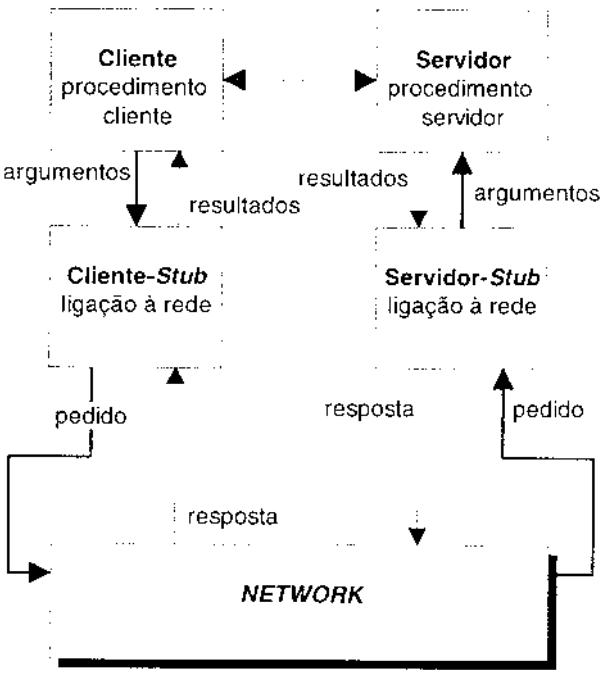


Fig. 4 - Chamada remota a um procedimento.

Presentemente existem dois sistemas predominantes de RPC, denominados, respectivamente, *Open Network Computing* (ONC) e *Network Computing System* (NCS). A breve apresentação de um sistema de RPC, que se segue, foca apenas o sistema ONC [4,5,6] (versão 4.0), visto que este é o mais divulgado e o único disponível em domínio público. No entanto, a filosofia e muitos dos métodos a seguir descritos, são também aplicáveis ao sistema NCS.

O sistema ONC permite a construção de aplicações distribuídas em dois níveis de desenvolvimento distintos, que denominaremos, genericamente, por alto e baixo nível. Do nível alto para o nível baixo, aumenta a complexidade da programação, aumentando simultaneamente, o grau de controlo e eficiência obtidos na aplicação. Convém realçar que estes dois níveis de programação, apesar de distintos, não são, de forma alguma, estanques. Pelo contrário, é muitas vezes desejável, que aplicações produzidas com primitivas de alto nível sejam posteriormente "afinadas" pelo programador, com primitivas de baixo nível, visando o aumento de eficiência da aplicação ou a introdução de funcionalidade que a alto nível não é possível obter.

O sistema ONC inclui um compilador de protocolo (RPCGEN) que, a partir da definição da interface entre o cliente e o servidor, gera automaticamente os *stubs* do cliente e do servidor, tornando assim mais expedito o desenvolvimento da aplicação. O programador pode refinar, posteriormente, o código gerado por este compilador.

Normalmente, o desenvolvimento de uma aplicação baseada num sistema RPC inicia-se pela definição da interface a usar na comunicação entre o cliente e o servidor. Esta definição consiste na identificação do procedimento remoto e na definição dos tipos de dados usados na comunicação.

Identificação do procedimento remoto

A identificação da função remota baseia-se na atribuição de identificações numéricas ao programa servidor a utilizar (identificações únicas em cada computador e limitadas a valores entre 0x20000000 e 0x3FFFFFFF), do procedimento propriamente dito e ainda da versão do procedimento, visto que o mesmo servidor pode manter várias versões distintas, da mesma função. Esta identificação será, posteriormente, utilizada tanto pelo cliente como pelo servidor para a troca de mensagens.

A definição seguinte é um exemplo típico:

```
#define MY_PROG     ((u_long) 0x20000000)
    /* Id do servidor */
#define MY_VERS     ((u_long) 1)
    /* Id da versão */
#define MY_PROC     ((u_long) 1)
    /* Id do procedimento */
```

Definição dos tipos de dados

Na definição da interface é essencial definir que tipos de dados devem circular entre o cliente e o servidor. Atendendo a que a representação interna dos diferentes tipos de dados varia de computador para computador, torna-se essencial definir os tipos dos dados de uma forma que seja compreensível por todos os computadores. Para atingir este fim o sistema RPC utiliza uma forma de representação dos tipos de dados, a que se dá o nome de *External Data Representation* (XDR) [6], que consiste num conjunto de rotinas que codificam e descodificam os dados usados localmente, segundo representações independentes.

Como exemplo apresenta-se um filtro que permite a utilização de *strings* como argumento ou valor de retorno de um procedimento:

```
#include <rpc/rpc.h>
bool_t xdr_filter(xdrs,objp)
XDR *xdrs;
char *objp;
{  return
   (xdr_string(xdrs,&objp,MAX_STRING_SIZE));
}
```

Neste exemplo, o argumento *xdrs* é utilizado para indicar se é uma codificação ou uma descodificação, o argumento *objp* é um ponteiro para a *string* de entrada ou de saída, conforme o valor de *xdrs*.

Para a transmissão de dados de tipos complexos como, por exemplo, as estruturas, é necessário construir filtros bidireccionais que procedem à sua decomposição e recomposição, em tipos simples, e à respectiva codificação e descodificação. A construção deste tipo de filtros pode ser efectuado pelo compilador de protocolo (RPCGEN).

O passo seguinte, no desenvolvimento da aplicação, é a construção do procedimento de serviço remoto. Este procedimento deve ser construído como se de um procedimento local se tratasse, apenas seguindo a restrição de usar como parâmetros de entrada e valores de retorno, dados dos tipos definidos anteriormente.

A ultima tarefa a executar é a construção do cliente, do servidor e dos respectivos *stubs*, através de primitivas de alto ou de baixo nível. De seguida apresentam-se, de uma forma simplificada, algumas destas primitivas.

Implementação de alto nível

As primitivas de alto nível, fornecidas pelo sistema, são as seguintes:

```
registerrpc(...);
svc_run(...);
int callrpc(...);
```

A primitiva *registerrpc* informa o computador servidor que o procedimento especificado, existente no programa servidor especificado, está pronto a ser utilizado por aplicações remotas (ou locais). Os seus argumentos incluem a identificação do programa servidor, do procedimento e da versão a utilizar, bem como os filtros XDR.

A função *svc_run* coloca o programa servidor num ciclo de espera por solicitações de outras aplicações.

A primitiva *callrpc* é utilizada pelo programa cliente para executar um procedimento existente num programa servidor. Como argumentos inclui, para além dos utilizados pela função *registerrpc*, a identificação do sistema servidor e os parâmetros de passagem à função remota. A utilização desta primitiva pressupõe que o programa servidor já procedeu ao seu registo (*registerrpc*) e se encontra á espera de pedidos de execução de um procedimento (*svc_run*).

A construção de um sistema RPC com primitivas de alto nível implica, como já foi referido, limitações relativamente ao controlo que é possível obter sobre o sistema. Uma das limitações mais importantes resulta da utilização do protocolo UDP para estabelecer a comunicação entre o cliente e o servidor. Existem restrições ao nível da quantidade de informação que é possível transmitir em cada ligação e não garante fiabilidade na comunicação [7].

Implementação de baixo nível

O compilador RPCGEN [4] é normalmente utilizado para produzir aplicações baseadas em primitivas de baixo nível.

O RPCGEN necessita que o programador especifique a interface a utilizar entre o cliente e o servidor. Esta definição é realizada produzindo um ficheiro em que se definem os nomes dos procedimentos remotos, que se pretende que o servidor reconheça, e os tipos de dados, utilizados nos seus parâmetros e nos valores de retorno. Baseado nesta especificação o compilador gera os *stubs* do cliente e do servidor (Fig. 5).

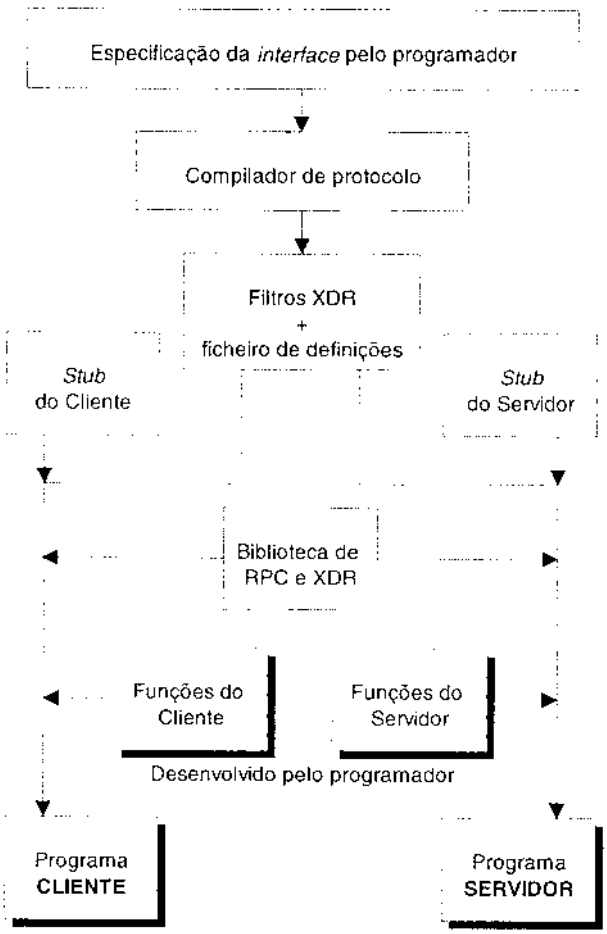


Fig. 5 - Desenvolvimento de aplicações utilizando o RPCGEN.

O compilador reconhece uma linguagem de descrição da interface (RPCL), organizada de modo similar à linguagem C e utilizando directivas de pré-processamento iguais às do *cpp*.

O compilador aceita seis tipos, diferentes, de definições:

- Constantes
- Enumerações
- Estruturas
- Uniões
- Tipos definidos pelo utilizador (*typedefs*)
- Programas

As enumerações, estruturas e *typedefs* são semelhantes às utilizadas em C, as constantes definem constantes, inteiras e simbólicas. As uniões são análogas aos registos variantes existentes em Pascal e finalmente as definições do tipo programa especificam os procedimentos que devem ser reconhecidos pelo servidor.

O RPCGEN reconhece ainda quatro tipos de declarações:

- Simples
- *Arrays* de tamanho fixo
- *Arrays* de tamanho variável
- Ponteiros

A sintaxe das declarações é, também, semelhante à utilizada em C.

Para terminar a apresentação do RPCGEN, é importante referir que a sua utilização implica que a interacção entre o cliente e o servidor não se faz por pedidos e respostas directas, mas sim através dos *stubs* produzidos. Ao interpretar a definição dos procedimentos disponíveis no servidor, descritos em RPCL, o RPCGEN atribui um nome para o procedimento (segundo uma regra bem definida) baseado no nome fornecido pelo programador na especificação da interface. Este identificador é utilizado simultaneamente para disponibilizar o procedimento (no servidor) e para invocar o respectivo serviço a partir do cliente.

As primitivas de baixo nível permitem um controlo mais eficaz dos diferentes aspectos do sistema, nomeadamente a escolha do protocolo de transporte (TCP ou UDP), a obtenção de informação respeitante ao cliente e ao servidor, a definição de temporizadores (*timeouts*) e dos processos de autenticação e encriptação [4].

Na construção de um sistema RPC, baseado neste tipo de primitivas, assume especial importância a existência de uma interface simples entre os *stubs* e os programas cliente e servidor. A importância desta simplicidade, tem por base o respeito pela filosofia deste tipo de sistema, no qual a chamada de um procedimento remoto deve ser virtualmente idêntica à chamada de um procedimento local.

Enumeram-se em seguida algumas das primitivas de utilização mais comum:

```
clnt_create(...);
clnt_destroy(...);
clnt_control(...);
clnt_call(...);
svctcp_create(...);
svcudp_create(...);
svc_register(...);
```

As primeiras três funções estão relacionadas com o desenvolvimento do cliente RPC.

A função *clnt_create* cria e inicializa as estruturas de dados respeitantes ao cliente e estabelece, ainda, a comunicação como o servidor. Os argumentos permitem especificar o sistema (*hostname*) e a aplicação servidores, o procedimento remoto, e a sua versão, e ainda o tipo de protocolo a utilizar na comunicação (UDP ou TCP). Esta primitiva deve ser chamada antes de qualquer outra acção que envolva o cliente.

A função *clnt_destroy* destroi a estrutura de dados criada para o cliente (*clnt_create*).

A primitiva *clnt_control* permite obter informações sobre as características do cliente ou actuar sobre estas de forma a modificá-las.

A função *clnt_call* faz a chamada efectiva do procedimento remoto. É possível explicitar o valor do *timeout* da solicitação. O uso desta primitiva implica, no mínimo, que já se tenha efectuado o *clnt_create* e que o servidor já esteja registado e pronto a aceitar pedidos.

Do lado do servidor existe um conjunto de serviços semelhantes.

A primitiva *svctcp_create* é utilizada para criar um serviço de transporte baseado no protocolo TCP e que será usado no servidor. Este tipo de serviço baseia-se *buffered I/O* pelo que permite definir o tamanho do *buffer* de recepção e de transmissão. Se estes valores não forem especificados serão usados valores por omissão que dependem do sistema.

A função *svcudp_create* é semelhante à primitiva *svctcp_create*, mas criando um serviço de transporte baseado no protocolo UDP.

A primitiva *svc_register* tem por função, registar a aplicação servidora no serviço *portmap*, do sistema servidor. Após o registo, o programa servidor passa a encontrar-se acessível a partir de outros computadores.

O sistema de RPC é, presentemente, um dos métodos de construção de sistemas distribuídos mais poderosos e flexíveis. Existem, actualmente, versões do sistema ONC que apresentam alguns melhoramentos relativamente á versão aqui apresentada, nomeadamente, um compilador de protocolo mais poderoso, melhores sistemas de segurança e ainda a possibilidade de construir aplicações independentes do protocolo de transporte.

III. MODELO GLOBAL DO SISTEMA DESENVOLVIDO

A apresentação anterior constitui um referencial das diferentes técnicas de programação de aplicações distribuídas, privilegiando os métodos RPC e passagem de parâmetros utilizados neste trabalho.

O modelo para partilha de informação entre processos distribuídos baseia-se em dois módulos cooperativos: um de comunicação local, assente em mensagens, e um outro para comunicações remotas, baseado em RPC.

A. Módulo de comunicação local

A arquitectura deste módulo de comunicação (Fig. 6), tem por base o método de passagem de mensagens apresentado anteriormente.

O elemento central deste modelo é um *router* que recebe e encaminha todas as mensagens. A transmissão e a recepção destas mensagens é da responsabilidade da unidade *MESG* que providencia os serviços de comunicação. Este objecto serve como uma *interface* entre qualquer processo comunicante e o sistema de passagem de mensagens. Os argumentos fornecidos a esta unidade (objecto) são o endereço do destinatário (sob a forma de um número inteiro) e os dados a enviar. Estes argumentos são codificados numa trama (Fig. 7) e colocados na pilha de mensagens.

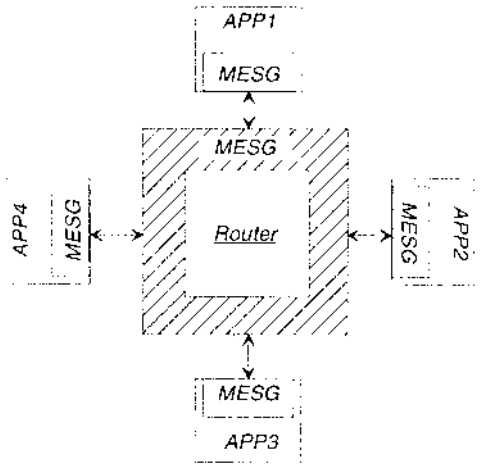


Fig. 6 - Esquema simplificado da arquitectura de comunicação.

A trama corresponde à estrutura de dados seguinte:

```
typedef struct MSGTYPE
{
    int len;    //#bytes em data (0 ou > 0)
    long type; // tipo da mensagem (> 0)
    struct
    {
        Address dest;
        char data[MAXMESGDATA];
    } frame;
} Mesg;
```

Cada processo que necessite de enviar dados a um outro processo, deve informar o *Router* da sua presença na rede virtual. Do mesmo modo, deve indicar o seu abandono no momento de fecho da sua actividade. O *Router* poderá, no entanto, confirmar a presença dos diversos intervenientes em intervalos de tempo regulares.

Um exemplo de funcionamento do sistema com dois processos, é apresentado na figura 8.

B. Módulo de Comunicação Remota

O módulo de comunicações baseado em RPC tem por objectivo permitir a uma aplicação, em execução num determinado computador, a utilização de serviços prestados por aplicações executadas noutros computadores. Pretendia-se que este módulo fornecesse uma interface independente (do ponto de vista das aplicações locais e remotas) para os pedidos e para as respostas obtidas, tornando-se o mais geral possível, de modo a permitir a sua utilização por aplicações dos mais diversos tipos. Para obter esta generalidade, a resposta a um pedido é efectuada na forma de uma lista ligada de

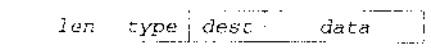


Fig. 7 - Formato da trama codificada por *MESG*.

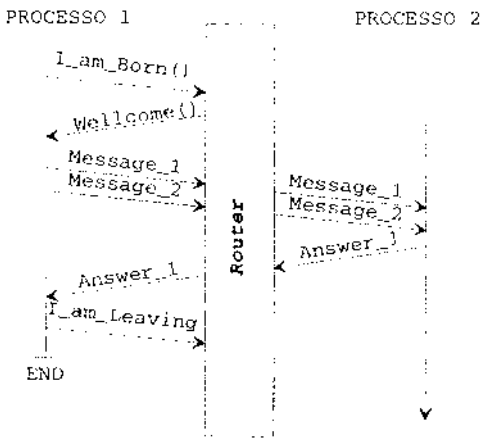


Fig. 8 - Exemplo de comunicação típico entre dois processos.

caracteres. Esta forma de representar a resposta, permite, quando necessário, o encapsulamento de qualquer tipo de dados, assim como permite, com facilidade, a recepção de resultados sob a forma de ficheiros. Esta última possibilidade revela-se extremamente útil devido ao facto de existirem muitas aplicações, que seriam, neste caso, as fornecedoras do serviço pedido, cuja execução resulta na criação de um ficheiro de dados.

No desenvolvimento deste sistema foi seguido um modelo que baseia a sua operação em dois canais distintos entre a máquina cliente e a máquina servidora. Um dos canais é utilizado na execução de pedidos e o outro para assinalar que os resultados já se encontram disponíveis (Fig. 9).

Quando os resultados das aplicações não colaborantes são disponibilizados sob a forma de ficheiros, o sistema funciona do seguinte modo:

1. A aplicação faz um pedido de execução de uma aplicação remota ao cliente local.
2. O cliente local transmite o nome dessa aplicação ao servidor existente na máquina remota. Antes de executar a transmissão do nome da aplicação, o cliente consulta uma base de dados local, para identificar a máquina a quem deve dirigir o pedido.
3. O servidor remoto inicia a execução do módulo de controlo de execução de aplicações, indicando-lhe qual a aplicação pretendida, colocando-se de seguida á espera de novos pedidos.
4. O módulo de controlo de execução de aplicações, procede á execução da aplicação pretendida, aguarda pelos resultados e coloca-os num ficheiro, se tal não for feito pela própria aplicação.
5. O módulo de controlo de execução de aplicações transmite ao cliente remoto o nome do ficheiro de resultados.
6. O cliente remoto transmite o nome do ficheiro de resultados ao servidor local.
7. O servidor local transmite ao cliente local o nome do ficheiro que contem os resultados.
8. O cliente local pede ao servidor remoto a transmissão do ficheiro de resultados.

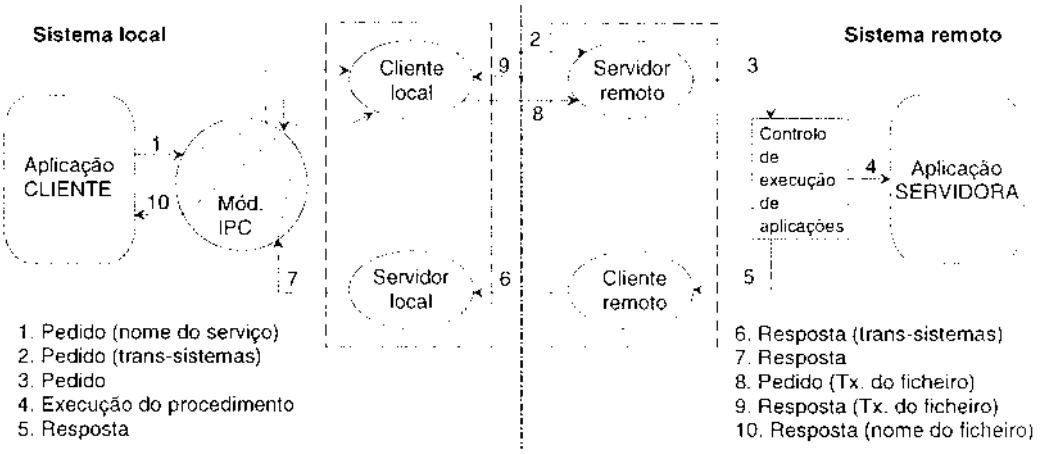


Fig. 9 - Implementação do sistema de comunicação remota.

9. O servidor remoto transmite o ficheiro de resultados ao cliente local.
10. O cliente local cria um ficheiro local de resultados, e transmite o nome desse ficheiro á aplicação que fez o pedido inicial.

A implementação apresentada não obedece ás regras normalmente seguidas num sistema deste tipo, visto que vulgarmente os pares cliente/servidor locais e remotos não se encontram sob a forma de programas distintos, mas sim integrados num único programa (Fig. 10).

A escolha deste sistema pouco usual, prende-se com a necessidade de fornecer um sistema que seja, simultaneamente, assíncrono e multi-tarefa. A primeira exigência está relacionada com o facto de a aplicação remota poder ter um tempo de execução longo, o que implica que o sistema remoto deveria receber o pedido, providenciar a sua satisfação e só após a sua conclusão enviar a resposta ao sistema que fez o pedido. A segunda exigência prende-se com a necessidade de garantir que um sistema possa facultar vários serviços simultaneamente, ou seja, após receber um pedido e antes de enviar o resultado, a máquina deve estar disponível para receber outros pedidos.

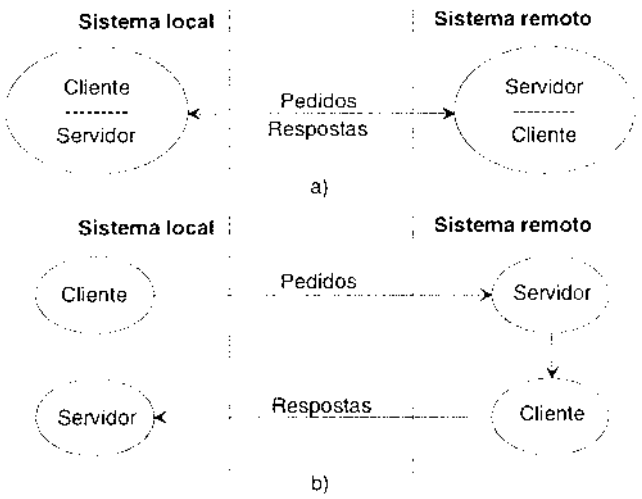


Fig. 10 - Ligação por RPC:
a) sistema clássico; b) sistema desenvolvido.

A principal vantagem da integração dos pares cliente/servidor num só módulo (Fig. 10-a), consiste no facto de a comunicação entre o sistema local e o remoto ser efectuado pela mesma entidade, quer para fazer os pedidos quer para receber as respostas o que, normalmente, é vantajoso, visto que tendo, pelo contrário, um servidor e um cliente seria necessário que o cliente estivesse permanentemente a verificar se a resposta já tinha chegado ao servidor. No entanto, neste caso específico essa vantagem deixa de existir, devido ao facto de ser utilizado, a nível local, um módulo de comunicação entre processos que permite ao próprio servidor informar o cliente da recepção da resposta esperada, evitando assim as verificações por parte do cliente. O sistema integrado tem também a vantagem de aproveitar melhor os recursos do sistema em que é desenvolvido, se bem que esta vantagem é relativamente reduzida.

As principais vantagens do modelo apresentado (Fig. 10-b), que levaram à sua escolha, baseiam-se na sua simplicidade, modularidade e facilidade de modificação, se tal for requerido no futuro. Em contrapartida o sistema integrado, implica uma complexidade no desenho dos módulos de comunicação varias vezes superior, o que implica que não possui nenhuma das características apontadas ao método de separação do pares cliente/servidor.

Pela avaliação feita das vantagens de cada um dos métodos concluiu-se que a poupança de recursos, no modelo integrado, era largamente excedida pelas vantagens do modelo de separação clientes e servidores.

C. Arquitectura global

A arquitectura proposta foi desenhada de modo a fornecer uma interface genérica para aplicações distribuídas, sejam elas locais ou remotas, interdependentes ou autónomas.

A comunicação entre processos residentes em sistemas diferentes implica a utilização conjunta dos dois métodos apresentados (Fig. 11). Esta interligação de processos não requer o conhecimento, por parte de nenhum deles, do

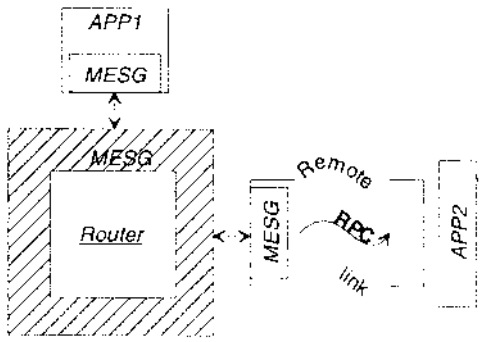


Fig. 11 - Arquitectura global de interligação entre processos.

sistema remoto, mas apenas das funções de comunicação disponibilizadas pelos módulos de IPC e de RPC.

O processo cliente será, em princípio, desenvolvido na aplicação principal que distribui tarefas e recolhe informação. O processo servidor poderá assumir diversas formas (Fig. 12):

- local, construído de raiz e adaptado ao sistema (Fig. 12-a);
- local e modificado (por intermédio de um processo adaptador) de modo a ser compatível com o sistema (Fig. 12-b);
- remoto e modificado (por intermédio de um processo adaptador) de modo a ser compatível com o sistema (Fig. 12-c).

Para qualquer destas transacções, a comunicação é feita de uma forma completamente transparente para o utilizador, como de uma *system call* se tratasse. Na figura não é representado o *Router*, pois, do ponto de vista dos processos comunicantes, ele não interfere na comunicação.

IV. DESENVOLVIMENTO DO SISTEMA

O sistema apresentado foi desenvolvido em sistemas

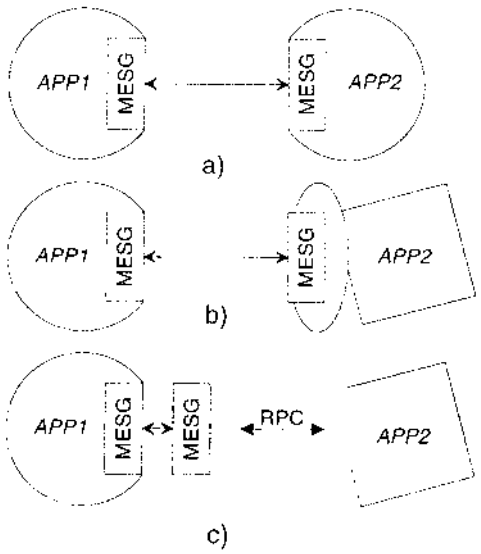


Fig. 12 - Comunicação entre diversos tipos de processos.

UNIX e SunOS, segundo o paradigma da programação orientada para objectos (C++). Descrevem-se em seguida alguns destes objectos relacionados com algumas das unidades anteriormente apresentadas.

A. Unidade "Router"

O processo *Router* é um processo autónomo que faz a comutação de mensagens entre todos os processos registados. O *Router* é baseado num objecto tipo `Core`, classe que é especificada segundo o protótipo:

```
class Core
{
private:
  Mesg mesg;
  int id;
  Address up[MAXPROCESS];
  int n_process;
  void ToMe(int);
  void Route(int);
  void mesg_send();
  int mesg_recv();
  void erro(char*);
  int execute(char*);
  void write_up(char*);
  void delete_up(char*);
  int gone(char*);
  int get_out(char*);
public:
  Core();
  ~Core();
  void Exec(char*);
  void ReadBus();
};
```

Em termos de operação, este objecto principia por ler uma mensagem da pilha, descodifica-a e efectua o seu processamento, que varia consoante a informação sobre o destinatário:

- mensagem para ele próprio - interpretação através do método *ToMe*;
- mensagem vazia - despreza-a e passa à próxima;
- mensagem de difusão - envia uma cópia para cada processo registado;
- mensagem dirigida a um processo registado - a mensagem é retransmitida (método *Route*).

B. Unidade "MESG"

A integração de uma aplicação no modelo de processamento distribuído proposto, deve ser efectuada à custa da unidade *MESG*, responsável pelo encapsulamento dos mecanismos de comunicação. O

objecto que representa esta unidade conceptual é especificado segundo a classe Bus.

```
class Bus
{
private:
  int id;
  char* buffer;
  Mesg mesg;
  Address my_add;
  int total_size;
  void mesg_send();
  int mesg_recv();
  void erro(char*);
  int read_data(char*, char*);
  int write_data(char*, char*);
public:
  Bus(Address);
  ~Bus();
  void Send(Address, char*);
  char* Receive();
  int Execute(char*,char*);
  void I_am_Leaving();
  void I_am_Born();
  Address getDestAdd();
};
```

A classe fornece métodos para o registo de cada aplicação (*I_am_Born*), para receber e enviar mensagens para os processos registados (*Receive* e *Send*), para executar localmente uma determinada aplicação (*Execute*) e para registar o abandono de um determinado cliente.

V. Cenários de Aplicação

O modelo de comunicação entre processos apresentado, foi desenvolvido no âmbito de um sistema de gestão de redes de dados. Este tipo de sistema tem uma necessidade particular de ferramentas de interligação, visto que a sua operação se baseia essencialmente num conjunto de aplicações autónomas distribuídas por diversos computadores ao longo da rede.

O sistema construído baseia-se numa aplicação central de processamento e gestão que coordena a operação de diversas aplicações, residentes em computadores distintos. Estas aplicações são autónomas e fornecem funcionalidades específicas, que no seu conjunto constituem o sistema de gestão. Estas funcionalidades abrangem áreas tão diversas como a monitorização de tráfego, detecção automática de topologia, contabilização e taxação de recursos e sistemas de controlo de segurança (Fig. 13).

A distribuição das aplicações autónomas tem por base, por um lado, a exigência de localização em pontos estratégicos da rede, e por outro lado, a falta de portabilidade de algumas aplicações desenhadas para arquitecturas computacionais particulares.

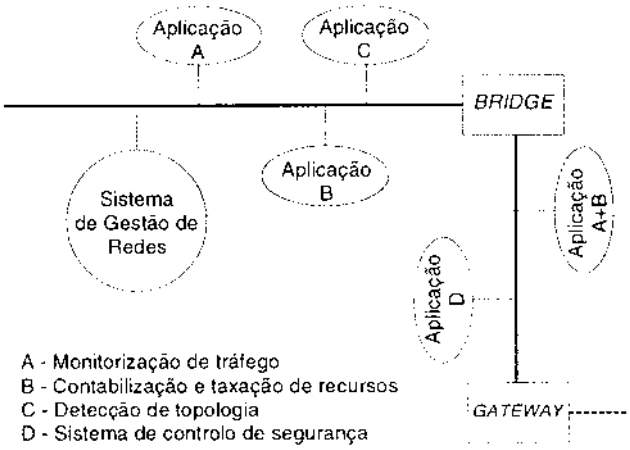


Fig. 13 - Comunicação entre diversos tipos de processos

VI. Conclusões

Foram apresentadas diversas técnicas para partilhar processos, distribuídos local ou remotamente em sistemas heterógeneos. Foi dado particular destaque ao método de passagem de mensagens (*IPC message queues*) como solução de distribuição local, e ao mecanismo de RPC (ONC RPC) pela suas potencialidades para distribuição de tarefas em diferentes sistemas.

Com base nestas duas técnicas foi proposto um modelo que permite o desenvolvimento de aplicações colaborantes e a integração de aplicações independentes e não colaborantes. Esta última facilidade explora a interface de entrada e saída de dados da aplicação para construir uma interface que adapta esta informação, segundo o modelo proposto.

Como cenário de utilização, foi apresentado um projecto de um Sistema de Gestão de Redes onde o modelo permite integrar aplicações específicas e autónomas, que enriquecem a informação disponibilizada, centralmente, pelo sistema de gestão

VII. Referências

[1] W. Richard Stevens, *Advanced programming in the UNIX Environment*, Addison Wesley, 1992.

[2] W. Richard Stevens, *UNIX Network Programing*, Prentice-Hall, Englewood Cliffs, N. J., 1990

[3] Rochkind, M. J., Advanced UNIX Programming, Prentice-Hall, Englewood Cliffs, N. J., 1985.

[4] J. Bloomer, *Power Programming with RPC*, O'Reilly & Associates, Inc., 1992.

[5] Sun Microsystems, Inc, "RPC: Remote Procedure Call, Protocol Specification, version 2", *RFC 1014*, 1987.

[6] Sun Microsystems, Inc. "XDR: External Data Representation standard", *RFC 1057*, 1988

[7] Douglas Comer, *Internetworking with TCP/IP - Principles, Protocol and Architecture*, Prentice-Hall, 1987.

Desenvolvimento de Aplicações para Sistemas Domóticos utilizando Programação Orientada por Objectos em C++

Henrique M. F. Vale, José A. C. Duarte, Sidónio M. Brazete
A. Manuel de Oliveira Duarte

Resumo- **Este artigo descreve o desenvolvimento de aplicações para sistemas domóticos utilizando programação orientada por objectos em C++, realizadas no âmbito do projecto final da licenciatura em Eng. Electrónica e Telecomunicações.**

Foi desenvolvido um sistema de comunicações para interligação dos apartamentos ao concentrador de informação do edifício, através da rede CATV, onde é feita toda a gestão dos eventos ocorridos nos apartamentos. O software de suporte às comunicações e de gestão de serviços foi desenvolvido em linguagem orientada por objectos em C++, sendo esta ultima parte desenvolvida em ambiente Windows, utilizando o compilador *Borland C++ 4.02*.

Abstract- **This paper describes the development of applications for domotic systems using object oriented programming in C++, made as a final project of Electronic Engineering graduation .**

During this work, was developed a communications system to link the apartments and make management of information on the building, using CATV network. The software that supports the communications and performs the management of information was done using object oriented programming for Windows with *Borland C++ 4.02*.

I. INTRODUÇÃO*

Nos últimos anos, com o desenvolvimento da industria de electrónica, tem-se assistido a um aumento considerável de dispositivos electrónicos destinados a executar tarefas domésticas e proporcionar lazer nos modernos lares dos nossos dias. É neste contexto que surge uma nova área na engenharia electrónica, a domótica, que tem por objectivo fazer uma gestão inteligente dos dispositivos eléctricos existentes na nossa casa e fornecer novos serviços, permitindo uma melhoria da qualidade de vida dos cidadãos.

Esta gestão, proporciona melhoramentos consideráveis na área da economia e segurança dos nossos lares. Na área da economia o objectivo é controlar dispositivos eléctricos como maquinas domésticas ou sistemas de iluminação optimizando a potência eléctrica disponível e utilizando sempre que possível horários de baixa taxação. Esta gestão não é apenas do interesse dos consumidores, mas também dos fornecedores, promovendo uma melhor exploração dos recursos energeticos que é um factor importantíssimo nos dias de hoje. Na área da segurança, os sistemas domóticos fornecem serviços indispensáveis num lar moderno, como são a detecção de intrusão, fogo, inundações ou fugas de gás, na existência de um destes alarmes são activados todos os dispositivos de segurança existentes no edifício, e são de imediato informados os serviços de socorro como por exemplo os bombeiros. Ainda na área de segurança o sistema oferece meios de vigilância e simulação de presença (na ausência dos habitantes permite a ligação de dispositivos eléctricos, como por exemplo iluminação, radio, etc. de modo a simular a presença humana na residência).

O trabalho de investigação e desenvolvimento no qual se inseriu este projecto, tinha como objectivo o desenvolvimento de um sistema de comunicações para interligação dos apartamentos com um concentrador de informação geral do edifício, e do respectivo *software* para gestão de serviços.

O trabalho aqui realizado ao longo do ano lectivo divide-se em duas partes, na primeira parte desenvolveu-se e testou-se o sistema de comunicações. Na segunda parte desenvolveu-se todo o *software* de suporte as comunicações e de gestão de serviços.

II. SISTEMA DE COMUNICAÇÕES

O sistema de comunicações a implementar deveria permitir, a transmissão no meio físico através da rede CATV, funcionar em modo *MASTER-SLAVE* e ter mecanismos de detecção e correcção de erros.

Após se ter feito um estudo dos protocolos *standard*, que poderiam ser implementados, optou-se pelo protocolo HDLC, que nos garante todos os pré-requesitos.

A. Protocolo HDLC

O protocolo de comunicação HDLC é um *standard* proposto pela ISO e IEC para a camada 2 (também conhecida como camada de ligação lógica ou "*data-link layer*") do modelo de referência OSI. Este protocolo define que as estações interlocutoras numa ligação podem ser de três tipos: primárias, secundárias e combinadas.

O protocolo assegura o controlo da comunicação através da troca de comandos e respostas entre as estações, as estações primárias enviam comandos e recebem respostas; as estações secundárias recebem comandos e enviam respostas e as estações combinadas recebem e enviam comandos e respostas.

* Trabalho realizado no âmbito da disciplina de Projecto.

As principais características do protocolo HDLC são:

- Utiliza transmissão bit a bit (*bit-oriented*).
- É um protocolo síncrono.
- Permite transmissão bidireccional (simultânea ou alterada).
- Estabelece ligações multiponto ou ponto-a-ponto.
- Utiliza a técnica de '*Sliding Window*'.
- Organiza as mensagens em tramas.
- É um protocolo aberto.

B. Estrutura da Trama HDLC

Designa-se por trama, ao conjunto de bits enviados por uma dada estação, a trama está organizada num conjunto de campos como mostra a figura 1.

Flag	Ender.	Ctrl	Informação	FCS	*Flag*
F	A	C	I	FCS	F
01111110	8 bits	8 bits	N bits	16-bits	01111110

Fig. 1 - Formato da trama HDLC.

Seguidamente descreve-se o significado de cada um dos campos:

- *Flag*: define o inicio e fim de uma trama. Esta sequência é constituída por um bit '0' seguido de 6 bits '1' contíguos, terminando com um bit '0', sendo este campo responsável pela sincronização da trama.
- Endereço: este campo identifica o endereço da estação secundaria que envia ou recebe as trama.
- Controlo: identifica o tipo de comandos ou respostas, pode conter ainda números de sequência das tramas.
- Informação/Dados: campo de dados, o seu comprimento é variável.
- FCS: este campo é composto por uma sequência de 16 bits, esta sequência permite a detecção de erros.

C. Modo de Funcionamento

O sistema desenvolvido é transparente ao utilizador, sendo todo o controlo e gestão da comunicação desempenhado pela estação primária (*MASTER*). A estação primária é que dá oportunidade às estações secundarias (*SLAVES*) de enviarem mensagens. Todas as mensagens têm que passar através da estação primária, podendo esta supervisionar todo o conteúdo das mensagens que circulam na rede.

Este sistema foi desenvolvido de modo a poder funcionar integrado num PC, através de um *bus* ISA, ou de um modo autónomo, característica que o torna bastante flexível.

D. Software do Building-Gateway[1]

Após se ter desenvolvido um sistema que permitisse comunicar com os diversos apartamentos de um edifício através da rede CATV, foi necessário desenvolver uma aplicação de software que através deste sistema fizesse toda gestão dos eventos que acontecem num edifício.

Este software permite ainda ao Buiding-Gateway estabelecer comunicações com o exterior via PSTN (Figura 2), nomeadamente com um centro de serviços especializado onde é feita uma gestão de serviços (supervisão de alarmes, telecontagem, televigilancia, etc.)

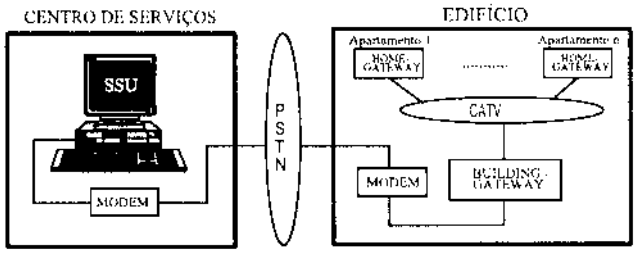


Fig. 2 - Mecanismos do sistema de comunicação

III. APLICAÇÕES PARA A UNIDADE SUPERVISORA DE SERVIÇOS (SSU)

A. Programação Orientada por Objectos em C++

A programação orientada por objectos faz-se através de uma abstracção do problema, organizando-o de uma forma mais "humana", permitindo lidar com a complexidade de uma forma mais eficiente. Esta organização faz-se através do estabelecimento de uma hierarquia para a aplicação a desenvolver, deste modo consegue-se desenvolver objectos que contém funções e variáveis para executar uma dada tarefa (encapsulamento de variáveis e dados), impossibilitando a interacção entre objectos, a não ser que isso esteja pré-definido.

[1] Concentrador de informação do edifício.

Para a implementação de uma aplicação orientada por objectos deve-se ter em conta as seguintes regras:

1. Para cada bloco funcional, i.e., conjunto de funções e variáveis necessárias para executar uma dada tarefa (acesso ao disco, impressão ou manipulação de dados de uma base de dados) deve-se definir uma classe.

2. Se duas classes diferentes possuem funções e variáveis em comum, deve-se definir uma classe base, esta classe base vai ter a definição das funções e variáveis que eram comuns, passando as duas antigas classes a serem descendentes desta nova classe.

3. Se uma classe é uma definição mais especifica de outra, então deve-se definir esta classe como descendente da classe mais geral.

4. Se a um dado objecto, for dada a possibilidade de criar novos objectos, então estes últimos serão membros do primeiro.

B. Desenvolvimento do software do SSU

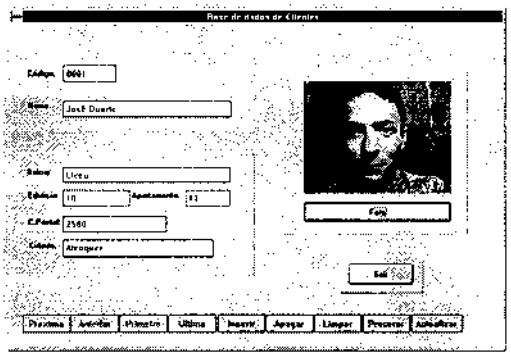


Fig. 3 - Base de dados de utentes.

O software para a Unidade de Sistemas e Serviços foi desenvolvido em *Windows*, utilizando o *Borland C++ 4.02*, esta aplicação é composta por:

- Base de dados de utentes.
- Comunicações.
- Base de dados de eventos.
- Configurações.

Todo o *software* desenvolvido, utiliza um interface gráfico 'amigável' e de fácil utilização por parte do utilizador. Todas as selecções e manipulações que o utilizador pode executar são feitas através de botões e menus de selecção.

Esta aplicação para o centro de serviços permite à entidade que gere o centro de serviços fazer a gestão remota dos eventos, gerados nos edifícios pertencentes a um centro habitacional, assim como fornecer serviços de televigilancia, telecontagem e supervisionamento de alarmes.

Esta *software* disponibiliza à entidade que gere o centro de serviços uma base de dados de clientes, com todas as informações relativas aos clientes aderentes ao serviço. Dispõe ainda de uma aplicação de comunicações que permite que o centro de serviços envie/receba mensagens de informação e/ou alarme dos clientes, e de uma base de dados de eventos (alarmes, acontecimentos, telecontagem, etc.) onde é feito a gestão destes mesmos eventos, a colecta destes eventos pode ser desencadeada manualmente pelo utilizador ou automaticamente pela própria aplicação.

Todas as comunicações entre o centro de serviços e os *Building-Gateway* dos edifícios, são feitas através da rede pública de telefone (PSTN).

C. Base de Dados de Utentes

A base de dados de utentes (Figura 3), foi desenvolvida, utilizando uma *pack* de *software* da *Borland* que permite a criação de bases de dados, programando apenas de uma forma visual, através do *Resource Workshop* da *Borland*. Esta base de dados, além de incluir todas as funcionalidades de qualquer base de dados, permite ainda a visualização da foto de cada elemento da base de dados.

D. Comunicações

Esta aplicação (Figura 4), permite o envio de mensagens do centro de serviços para cada um dos apartamentos. As funções que executam as comunicações foram criadas, recorrendo ao *Windows API*.

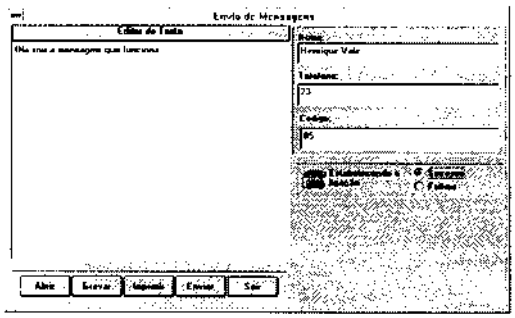


Fig. 4 - Comunicações.

E. Base de Dados de Eventos

Esta base de dados (Figura 5), permite ao centro de serviços fazer a visualização e processamento de todos os eventos de um centro habitacional, a colecta de

informações pode ser feita manualmente ou automaticamente, para a colecta ser feita automaticamente é necessário programar o número de colectas diárias que devem ser feitas.

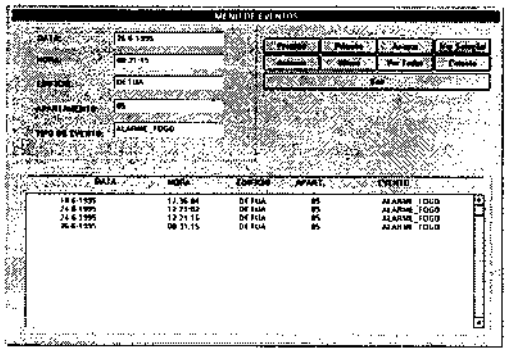


Fig. 5 - Base de dados de eventos.

F. Configurações

Esta aplicação permite ao utilizador fazer a configuração das portas de comunicação a serem utilizadas, bem como definir os comandos utilizados para estabelecer e cessar a chamada.

IV. CONCLUSÕES

O trabalho descrito, permite a implementação de futuros Centros de Serviços para edifícios inteligentes, através de uma *package* flexível de software orientada por objectos.

Este trabalho permite ainda que se crie uma rede de comunicação eficiente entre os apartamentos e o *Building-Gateway*, utilizando as infra-estruturas existentes, como é o caso da rede CATV, para disponibilizar serviços como os descritos anteriormente.

Este trabalho fez parte do projecto CHIMENE (*Colective Home Interface Made on Existings Networks in Europe*), que está englobado no programa ESPRIT (*European Strategic Programme for Research in Information Technologies*), da colaboração do Grupo de Sistemas de Banda Larga com o projecto CHIMENE, resultou na instalação de um destes sistemas numa residência de estudantes, permitindo o desenvolvimento e teste de novos sistemas em condições reais de funcionamento.

REFERÊNCIAS

[1] Borland International, "Object Windows 2.0 Programmer's Guide", 1994

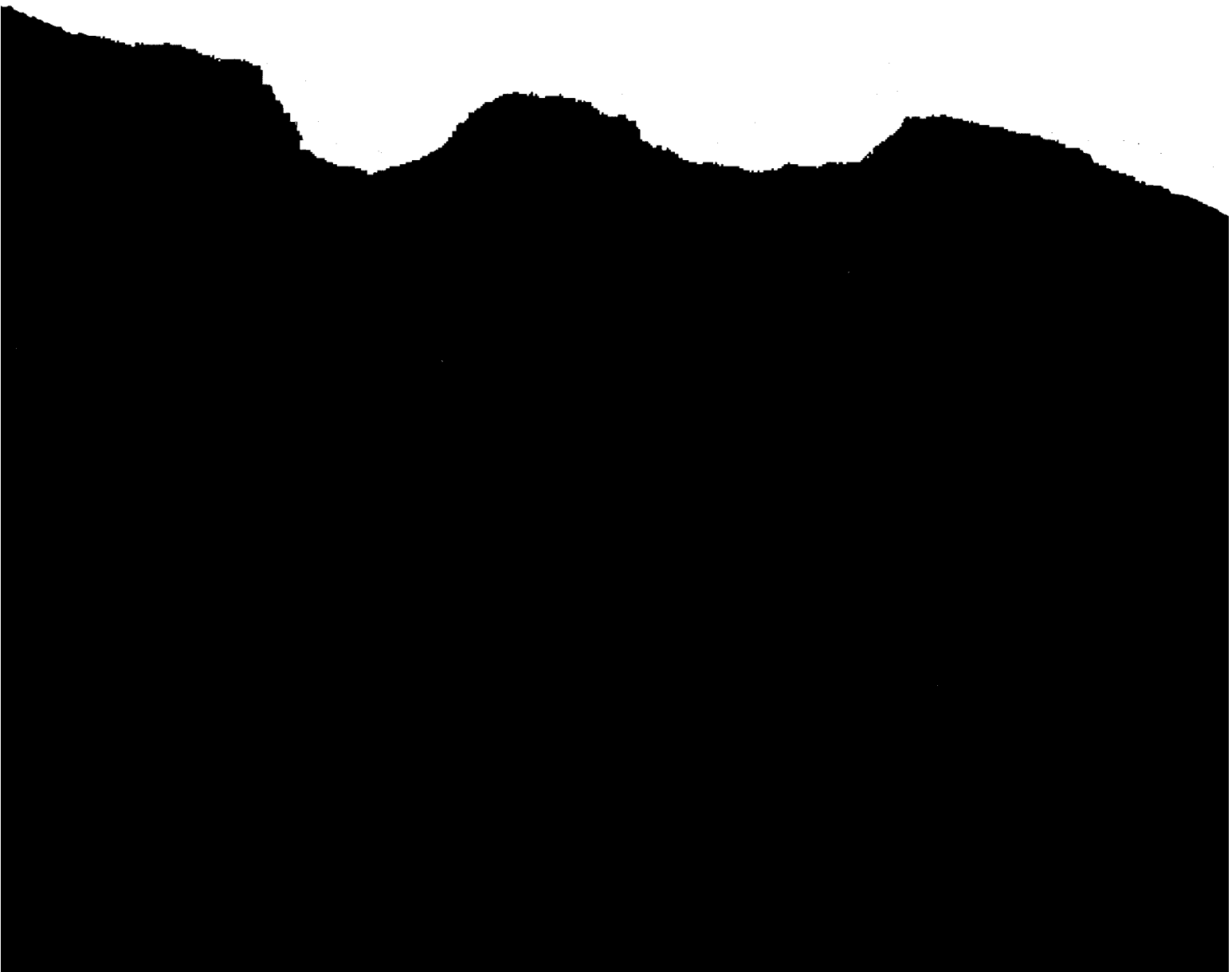
[2] Borland International, "Resource Workshop User Guide", 1994.

[3] Borland International, "Windows API", 1994.

[4] William Stallings, "Data and Computer Communications", Macmillan, 1990.

[5] B. Stroustrup, "The C++ Programming Language", Addison-Wesley, 1985.

[6] Valery Sklyarov, "The Revolutionary Guide to Turbo C++", Wrox Press Limited, 1992.



ARTES GRÁFICAS
Sérgio Cabaço