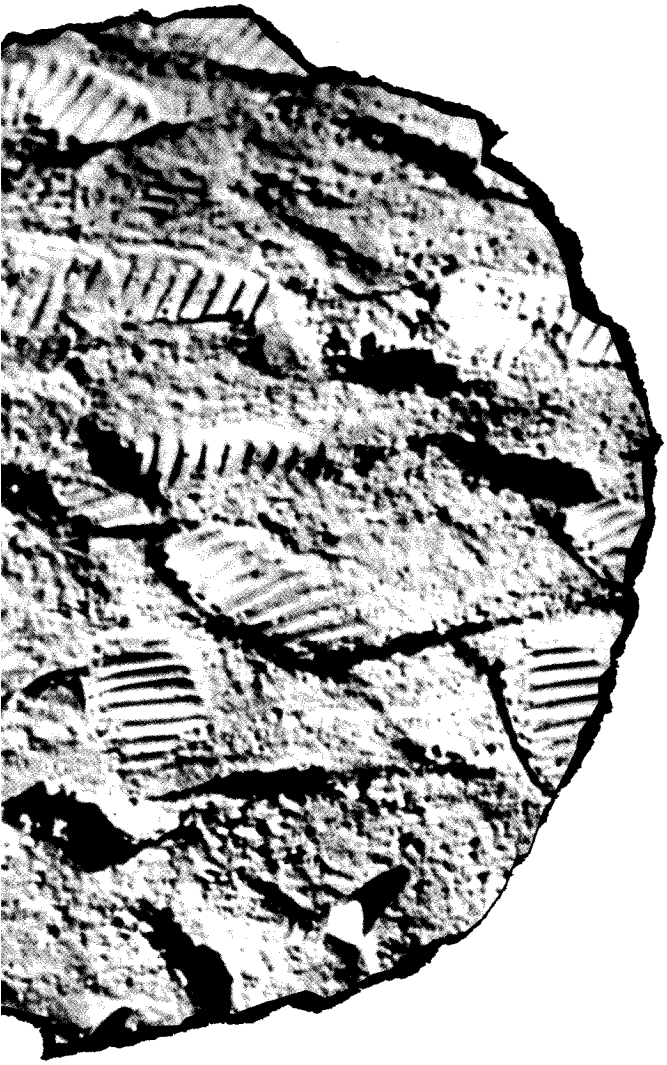
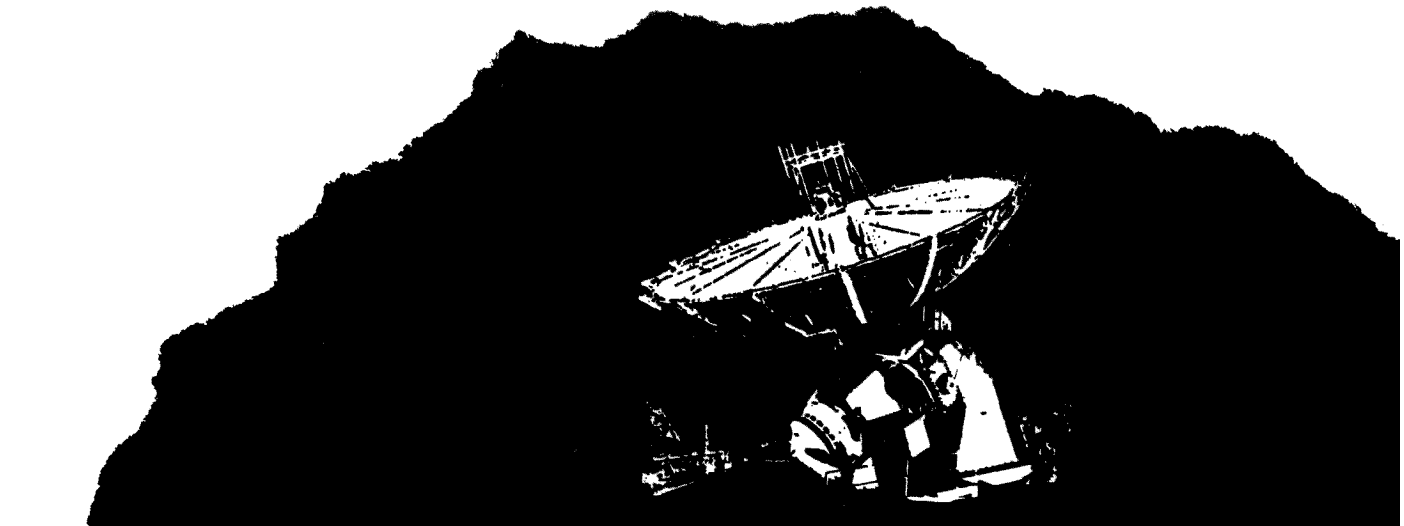
ISSN: 1645-0493

Electrónica e Telecomunicações

universidade de aveiro

theoria poiesis praxis





AVEIRO • JAN • 2001 • VOL. 3 • Nº3

Revista do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro

Electrónica e Telecomunicações

Revista do Departamento de
Electrónica e Telecomunicações
da Universidade de Aveiro

Editores:

Francisco Vaz
Augusto Silva

Comissão Editorial:

Alexandre Manuel Moutela Nunes da Mota
Ana Maria Perfeito Tomé
Aníbal Manuel Oliveira Duarte
António Joaquim da Silva Teixeira
António José Nunes Navarro Rodrigues
António Luis Jesus Teixeira
António Manuel Adrego da Rocha
António Manuel de Brito Ferrari Almeida
António Manuel Melo de Sousa Pereira
António Manuel Nunes da Cruz
António Rui Oliveira e Silva Borges
Armando Carlos Domingues da Rocha
Armando Humberto Moreira Nolasco Pinto
Armando José Formoso de Pinho
Atílio Manuel da Silva Gameiro
Augusto Marques Ferreira da Silva
Carlos Alberto da Costa Bastos
Dinis Gomes de Magalhães dos Santos
Ernesto Fernando Ventura Martins
Fernando Manuel dos Santos Ramos
Francisco António Cardoso Vaz
João Nuno Pimentel da Silva Matos
João Paulo Trigueiros da Silva Cunha
João Pedro Estima de Oliveira
Joaquim Arnaldo Carvalho Martins
Joaquim Manuel Henriques de Sousa Pinto
José Alberto dos Santos Rafael
José Alberto Gouveia Fonseca
José Artur Ferreira da Silva e Vale Serrano
José Carlos da Silva Neves
José Carlos Esteves Duarte Pedro
José Fernando Rocha Pereira
José Joaquim Gomes Fernandes
José Luis Costa Pinto Azevedo
José Luis Guimarães Oliveira
José Luis Vieira Cura
José Manuel Neto Vieira
José Rodrigues Ferreira da Rocha
Luis Filipe de Seabra Lopes
Luis Miguel Pinho de Almeida
Manuel Alberto Reis Oliveira Violas
Manuel Bernardo Salvador Cunha
Maria Beatriz Alves Sousa Santos
Nelson Fernando Pacheco da Rocha
Nuno Miguel Gonçalves Borges de Carvalho
Osvaldo Manuel da Rocha Pacheco
Paulo Jorge dos Santos Gonçalves Ferreira
Paulo Miguel Nepomuceno Pereira Monteiro
Pedro Nicolau Faria da Fonseca
Rui Fernando Gomes de Sousa Ribeiro
Rui Jorge Morais Tomás Valadas
Rui Manuel Escadas Martins
Tomás António Mendes de Oliveira e Silva

Morada e Secretariado:

Departamento de Electrónica
e Telecomunicações
Universidade de Aveiro
Campo Universitário
3810 AVEIRO
Portugal

Artes Gráficas :

Sérgio Cabaço

Tiragem : 400 exemplares

Depósito Legal Nº 115607

ISSN: 1645-0493

Editorial

A Revista do DET continua a merecer a atenção das pessoas que ensinam e investigam no nosso Departamento. Na realidade, esta publicação que agora conta com o ISSN 1645-0493, surge novamente como um veículo priviligiado para a divulgação das nossas actividades investigação e ensino com realce significado a pós-graduação.

Os editores aproveitam para manifestar abertura para continuar a acolher, ainda que de forma limitada, iniciativas editoriais associadas à publicação de actas de eventos científicos cuja organização envolva de forma significativa membros do DET.

Aguardando desde já contribuições para uma próxima edição, os editores manifestam o seu agradecimento a todas as entidades que duma ou doutra forma viabilizaram a concretização de mais uma edição desta Revista.

Francisco Vaz, Augusto Silva

A edição desta revista é subsidiada pela
Fundação para a Ciência e Tecnologia

Índice

Pitch Detection in Vocal Music Monophonic Signals
António Sá Pinto, Ana Maria Tomé 197

Modelos matemáticos e problemas de optimização combinatória
Iouliia Skliarova, António B. Ferrari 202

Diário da Assembleia da República Electrónico
Joaquim Sousa Pinto, Pedro Almeida 209

HiParaGraphs, uma Linguagem de Especificação de Algoritmos de Controlo Paralelos e Hierárquicos
Andreia Melo, Valery Sklyarov, António Ferrari 214

A Coprocessor for Traffic Scheduling and Schedulability Analysis in FTT-CAN
Ernesto Martins, José Alberto Fonseca 222

An Introductory Course on Human Computer Interaction
Beatriz Sousa Santos, Joaquim Sousa Pinto 228

Interacção com um Robô através de Linguagem Natural Falada
Nuno M. Ferreira, Nuno F. Ferreira, Mário Rodrigues[1], *António Teixeira, Francisco Vaz, Luís Seabra Lopes* 233

Goodears – Robot orientado por um farol acústico
João Reis, Paulo Santos, José Neto Vieira e Luís Almeida 242

Extracção de dados sobre o formato dos lábios nos sons da Língua Portuguesa
Alexandra Grilo, Pedro Duarte, António Teixeira, Armando Pinho 248

Sistema de Controlo em Tempo Real da Qualidade da Água para Consumo Humano
Osvaldo Rocha Pacheco, Luís Pedro Antunes, Rui Ferreira[2] 254

Resumos PhD e MsC

Síntese articulatória das vogais nasais do Português Europeu
António Joaquim da Silva Teixeira 260

Reconstrução de Sinal e Codificação
José Manuel Neto Vieira 261

Modelos, Métodos e Ferramentas para Implementação de Unidades de Controlo Virtuais
Arnaldo Silva Rodrigues de Oliveira 262

Especificação, Optimização e Teste de Algoritmos de Controlo Hierárquicos
Andreia Barbosa de Melo 264

[1] Faculdade de Ciências da Universidade do Porto

[2] DECQ - Serviços Municipalizados de Aveiro

Pitch Detection in Vocal Music Monophonic Signals

António Sá Pinto, Ana Maria Tomé

Resumo – **Este trabalho é acerca da determinação de pitch em sinais musicais monofónicos produzidos por uma fonte vocal. O Algoritmo de Detecção de Pitch (PDA) baseia-se na função de autocorrelação, um dos mais explorados métodos de detecção da frequência fundamental. Contudo, é desenvolvida uma nova aproximação ao processo de estimação de pitch. Esta nova estratégia consiste na introdução de uma unidade interactiva de processamento lógico que aumenta a cooperação entre o extractor central e o bloco de pós-processamento (dois dos três blocos que caracterizam a maior parte dos PDAs), de forma a evitar estimativas erradas de pitch.**

Abstract - **This work is concerned with pitch* determination in vocal music monophonic signals. The proposed Pitch Detection Algorithm (PDA) is based on the autocorrelation function, one of the most explored fundamental frequency detection methods. However, a new approach to the estimation process is developed. This new strategy consists in the introduction of a new logic processing interaction unit that enhances the co-operation between the central extractor and postprocessor blocks (two of the three blocks that characterise most PDAs), in order to avoid erroneous pitch estimates.**

I. INTRODUCTION

The problem of estimating the fundamental frequency of speech signals occupies a key position in the signal processing research area. It has many potential applications in different areas such as transmission, synthesis and recognition of speech, and plays the leading role in systems for helping to correct speech impediments of the handicapped [1]. Pitch determination of speech signals is not a simple task. The arduousness of this operation arises from the non-stationary nature of the speech waveform.

This paper will however focus on the application of pitch estimation to vocal music processing, an area that has been relatively little explored in comparison to the massive investigation carried out by the speech community. However, an automatic pitch detector, capable of extracting the fundamental frequency from singing voices would have many interesting applications, such as: systems for computer-assisted singing teaching or ear-training, automatic score transcription, analysis of microtonal non-Western music, real-time control of MIDI devices, etc. [2].

The main differences of vocal music signals, in comparison to speech signals, are related to the wider range of fundamental frequency (from ≈82.4Hz to ≈987.7Hz)[6], and the enormous variations in timbre (and therefore in spectral content) that a singer can produce in a single piece of music. These aspects should be carefully regarded in order to develop a reliable method of extracting the fundamental frequency of vocal music signals.

In this work we present a Pitch Detection Algorithm based on the autocorrelation function[3]. Some modifications are introduced in the PDA structure in order to increase its performance, reliability and accuracy. The software implementation of this PDA was developed in MATLAB.

We begin with a brief description of the structure of general PDAs, and the problems associated with this kind of approach. Then, the basic characteristics of the pitch estimation method of this PDA, the autocorrelation function, are described. Afterwards, we approach the implementation of the proposed PDA and finally, we present and analyse the results obtained with the analysis of synthesized, samples and real signals.

II. PITCH DETECTION ALGORITHMS

Most of the PDAs are characterised by the following blocks: the pre-processor, the central extractor and the postprocessor [1]. The central extractor performs the main task: it converts the input signal into a series of pitch estimates. The task of the pre-processor is data reduction and enhancement in order to facilitate the operation of the central extractor. The postprocessor operates in a more application-oriented way. Some of its typical tasks are error correction, smoothing the pitch contour and refining the pitch estimation.

The main problems with this structure occur when in presence of more complex situations, such as voicing transitions. In our point of view, this is due to a lack of an effective interaction between the central extractor and the postprocessor, since in this model, the pitch estimation and its correction are made separately.

* Although there is a psychoacoustical distinction between "pitch" as a perceived quantity and "fundamental frequency" as a physical quantity, in this paper, these terms are used indistinctly in reference to the fundamental frequency of voice and the measurement unity used is Hz.

III. THE AUTOCORRELATION FUNCTION

There are several reasons why autocorrelation methods have generally met with good success. The autocorrelation computation is made directly on the waveform, is a fairly straightforward (albeit time consuming) computation, and, above all, is phase insensitive [3]. The use of a zero phase method is particularly promising for the study of musical signals, since this means that contributions from all of the harmonics occur at the period of the fundamental, and any problem of a non-existent or weak fundamental is thus circumvented [4]. However, there are several problems associated with the use of this method.

Although the autocorrelation function of a voiced section of a vocal piece generally displays a prominent and isolated peak at the pitch period, there are also often present peaks due to the detailed formant structure of the waveform [1]. Another problem is the required use of a window for computing the short-time autocorrelation function. This exigency comprises three difficulties. First there is the problem of choosing an appropriate window. Second, no matter which window is selected, it will taper the autocorrelation function smoothly to 0, an effect known as linear tapering [5]. This effect tends to compound the difficulty mentioned above in which formant peaks in the autocorrelation function (which occur at lower indices than the period peak) tend to be of greater amplitude than those due to the fundamental. A final difficulty is the problem of choosing an appropriate analysis window size. The ideal analysis frame should contain from two (necessary) to three (preferred) complete pitch periods. Thus, for male voices (low pitch), the analysis frame should be long, whereas for female voices (high pitch) it should be kept short.

IV. THE PROPOSED ALGORITHM

The proposed method includes a logic processing interaction unit, an intermediary processing step between the central extractor and the postprocessor. The goal of this interaction unit is to prevent the erroneous estimation of pitch, in opposition to typical PDA analysis, for which the burden of correcting estimation errors produced by the central extractor is exclusively imputed to the postprocessor. This new interaction processing step implements a logic based on a four-state model, capable of dealing with the characteristics of different-nature segments of this type of signals.

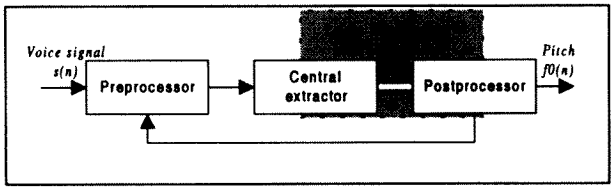


Figure 1. Block diagram of the proposed PDA

A. Pre-processor

The goal of the pre-processor is to eliminate, or at least reduce the problems of the autocorrelation method as a voice pitch estimator.

Our pre-processor is composed of four blocks. The first one implements the adaptive segmentation of the voice signal, which allows the use of appropriate values for the analysis frame size and reduces the computational cost of the autocorrelation computation. The default analysis frame size is 30.8 ms, the maximum size required to cope with pitch values corresponding to the low end of the fundamental frequency voice range (≈65.4Hz). After five consecutive voiced segments, the window size is altered to the triple of the average of the pitch periods of these five segments. The factor of 3 allows up to a 50 percent variation in pitch period from the estimated average pitch period, and still ensures that at least two complete pitch periods are contained within each analysis frame. The second block distinguishes between silent or final transient segments and other type of segments. The silence level threshold is set to 1/15 of the magnitude of the maximum peak in the whole voice signal. A final transient segment is defined as one for which its maximum peak is below ½ of the peak magnitude of the previous segment. The third block function is to whiten or spectrally flatten the signal, with a time domain non-linear distortion method, based on previous works [2]. The analysed frame is centre and peak clipped, resulting in a signal which can assume one of three possible values: -1, 0 or 1. The last block deals with the computation of the autocorrelation function. The autocorrelation function (which is equivalent to the inverse Fourier Transform of the power spectrum [5]) is calculated with the use of FFT techniques, in order to decrease its processing time. Then, it is normalised to unity at origin (lag 0). Finally, the effect of linear tapering is corrected with its inverse transformation.

B. Central Extractor and Postprocessor

As mentioned above, these two blocks collaborate within the logic processing interaction unit, which is based in a four-state model, depicted in figure 2.

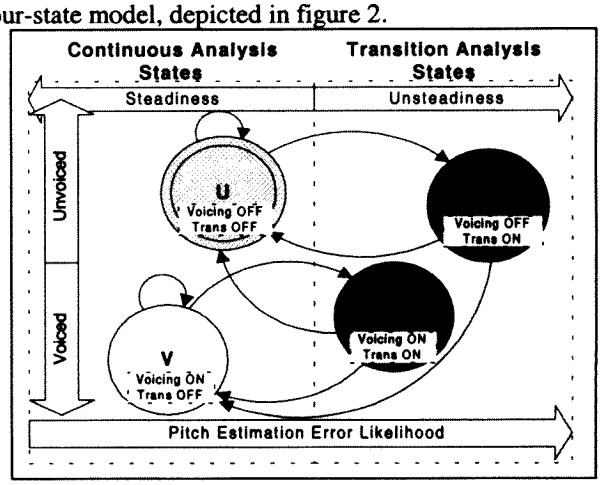


Figure 2. Four-state analysis model

There are four separate logic paths (or states), each of which are selected, based on two control variables (*voicing* and *trans*). These two variables can assume two values: on or off. The first one regards the voiced or unvoiced classification of the last segment. The second one indicates if there were detected evidences of a possible voicing onset or offset transition.

The goal of the transition analysis states (**V**oiced-**U**nvoiced or **U**nvoiced-**V**oiced) is to ascertain the veracity of the voicing transition hypothesis, raised by their corresponding continuous analysis state (**V**oiced or **U**nvoiced). These logic paths implement a more cautious processing, given the ambiguous nature of the segments they analyse. This is achieved, for example, with the use of higher constraints for threshold parameters.

The pitch estimation is made basically with an inspection of the maximum of the autocorrelation function (detected by the central extractor), and comparison with predefined thresholds, empirically obtained from the analysis of several musical phrases from different singers. Whenever the last segment (*n-1*) analysis exhibited a prominent peak, its position is used in order to restrict the range of acceptable autocorrelation peaks for the *n*th. segment, since there are obvious physical limitations to voice fundamental frequency variability in adjacent frames. This pitch tracking strategy reduces both computational effort and pitch error estimation likelihood.

One of the major drawbacks of the autocorrelation function, its proneness to harmonic or subharmonic detection, is due to the fact that the function is itself periodic in the true pitch period [1]. The harmonic detection likeliness is reduced by the linear tapering correction routine in the pre-processor. The subharmonic detection impairs drastically the prospect of success for the overall pitch detection, since both the adaptive segmentation and the logic processing unit are very susceptible to gross pitch errors. This problem is circumvented with the introduction of an algorithm that checks the autocorrelation samples corresponding to the sub-multiple positions of its maximum peak, and chooses the lowest order peak whose magnitude exceeds 80% of the original peak magnitude. Since the subharmonic detection probability attains its maximum at the onset of voicing (for high-pitched singers), the correction algorithm mentioned above is activated for the states U and UV.

Finally, a frequency-domain method of interpolation was implemented in order to refine the pitch estimation for high-pitch signals, since the autocorrelation resolution progressively falls for increasing fundamental frequency values.

V. RESULTS

One main aspect for the development of a PDA is its evaluation through standard databases. Such databases exist only for speech signals[2], which constitutes one of the main difficulties in developing a musical PDA. In order to surpass this problem, we developed a database composed of synthesized[13], samples[14] and real signals. The three types of signals allow us to analyse the performance of our PDA on a growing difficulty logic. The use of synthesized voice signals allows the objective measurement of PDA accuracy, but does not faithfully represent the characteristics and peculiarities of the human voice. Since the samples may have vibrato, we don't have exact information on their fundamental frequency. Nevertheless, this kind of signals allow us to evaluate the PDA in a global way, since we can reproduce rather fast and complex melodies(with a great degree of certainty concerning the pitch), representing all kinds of human voices(from the bass to the soprano). Although the human nature of voice impedes an absolute control over the produced fundamental frequency, thus turning the evaluation process in some kind of a subjective measurement, the real signals allow the analysis of the overall performance and robustness of the PDA, for the situations for which it was created.

A. Synthesized Signal

Synthesized Signal				**Results**		
Begin Time (ms)	**Note**	**Frequency (Hz)**	**Obtained* Frequency (Hz)**	**Unvoiced Frames**	**Voiced Frames**	**Voiced Estimates Average**
112,1	E1	82,410	82,645	1	20	82,645
354,1	F1	87,310	86,957	0	23	86,957
607,1	F#1	92,500	92,593	0	23	92,593
855,5	G1	98,000	98,039	0	22	98,039
1100,3	G#1	103,830	104,167	0	23	104,167
1359,5	A1	110,000	109,890	0	23	109,890
1605,2	A#1	116,540	116,279	0	22	116,279
1854,6	B1	123,470	123,457	0	23	123,457
2105,7	C2	130,810	131,579	0	23	131,579
2356,5	C#2	138,590	138,889	0	22	138,889
2601,3	D2	146,830	147,059	0	23	147,059
2852,9	D#2	155,560	156,250	0	23	156,250
3102,5	E2	164,810	163,934	0	23	163,934
3352,6	F2	174,610	175,439	0	23	175,439
3603,4	F#2	185,000	185,185	0	23	185,185
3851,8	G2	196,000	196,078	0	23	196,078
4101,7	G#2	207,650	208,333	0	23	208,333
4351,3	A2	220,000	222,222	0	23	222,222
4603,3	A#2	233,080	232,558	0	23	232,558
4852,7	B2	246,940	250,000	0	23	250,000
5100,7	C3	261,630	263,158	0	23	263,158
5351,5	C#3	277,180	277,778	0	24	277,778
5603,5	D3	293,660	294,118	0	24	294,118
5851,7	D#3	311,130	312,500	0	25	312,500

Table 1. Synthesized Signal Analysis

The results obtained with the analysis of the synthesized signal were very satisfactory, as we can see by Table 1. The only error reported was a voiced-unvoiced decision, made by our PDA at the beginning of the first note. All other results allowed us to verify the accuracy of our PDA,

* The Obtained Frequency of the signal refers to the real frequency of the synthesized signal, due to the discrete nature of the voice synthesizer.

since the pitch estimates coincided (with a precision of 0.0005Hz) with the synthesized signal (obtained) frequency.

For the results shown in Table 1, it were not included the pitch estimates corresponding to different note overlapping frames.

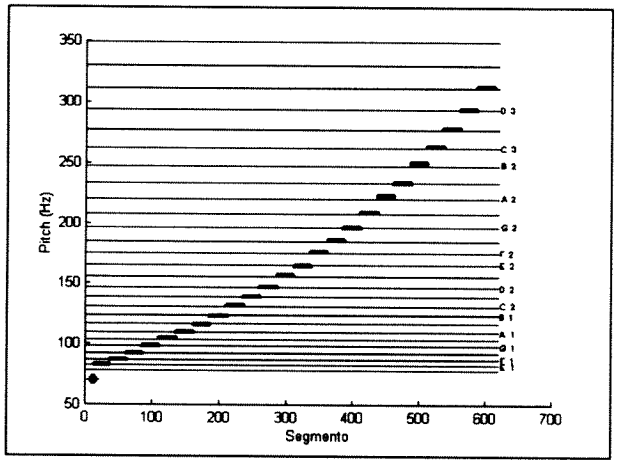


Figure 3. Chromatic Scale (2 octaves)

B. *Samples*

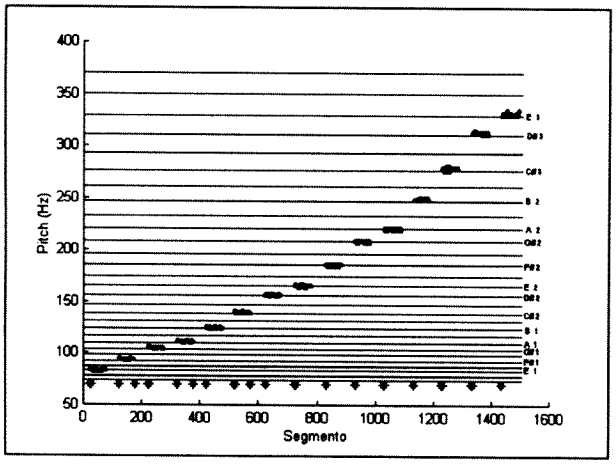


Figure 4. Bass sample, E Maj scale(2 octaves)

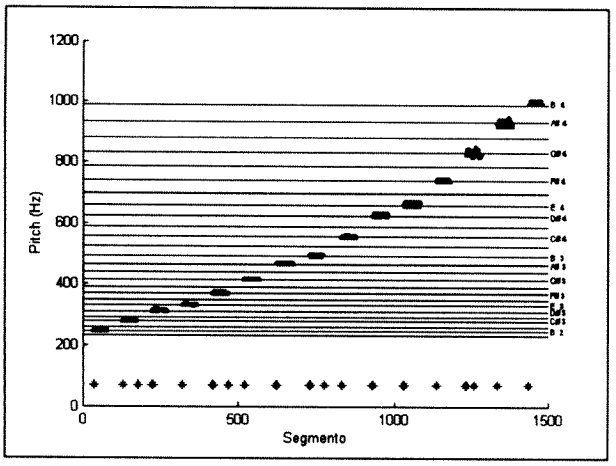


Figure 5. Soprano sample, B Maj scale(2 octaves)

The major contribution of the samples analysis was the ability to evaluate the PDA performance with different kinds of voices. As we can see by figure 4 and figure 5, the PDA proved its ability to analyse the full human voice fundamental frequency range. Once more, it were detected erroneous voiced-unvoiced decisions at the voicing onset.

C. *Real Signals*

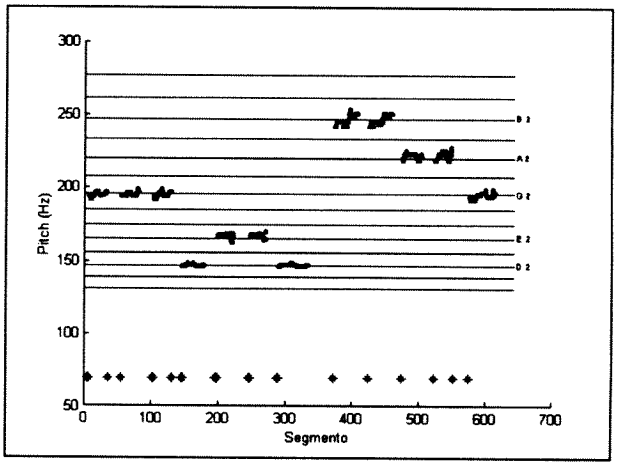


Figure 6. "Old MacDonald had a farm" - Male voice

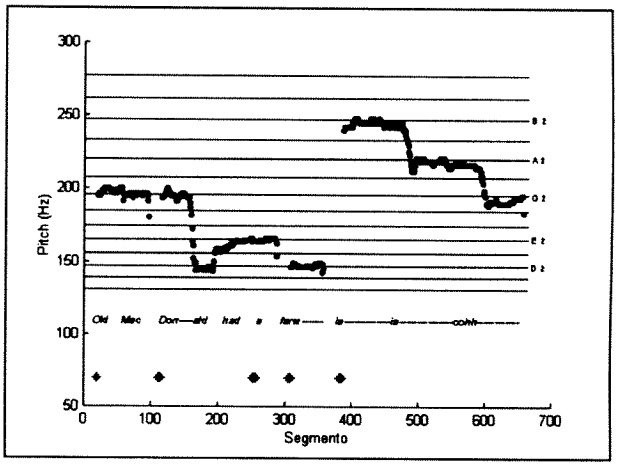


Figure 7. "Old MacDonald had a farm" - Male voice (with lyrics)

The analysis of the real signals demonstrated the robustness of the PDA, namely with the analysis of a vocal excerpt with (figure 6) and without lyrics (figure 7), that introduce uncharacteristic aperiodicities on the signal, thus raising difficulties to the good performance of the PDA.

The fine detail of our PDA was also demonstrated with the detection of some musical occurrences characterised by minor frequency variations: vibrato, portamento and minor untunings.

VI. CONCLUSIONS

We think that the good performance of this PDA is mainly due to its innovative architecture, that reflects the understanding of the human voice production system and

the performance of the autocorrelation function as a pitch detector.

The overall performance of our PDA was highly satisfactory, since it could detect the pitch of different kinds of voice signals, covering the full human voice fundamental frequency range. The only inaccuracy detected was its aptness to classify segments of low periodicity (typically the voicing onset) as unvoiced segments. This innacuracy is due to the severe constraints imposed when processing onset voicing segments, which in most of the PDAs originate gross pitch determination errors.

VII. REFERENCES

[1] Hess,W.J. (1983). " Pitch Determination of Speech Signals-Algorithms and Devices ", Springer-Verlag, Berlin, Germany

[2] F.J.Casajús-Quirós; P.Fernandez-Cid (1994). "Real-time, Loose-Harmonic Matching Fundamental Frequency Estimation for Musical Signals", ICASSP-IEEE International Conference on Acoustics, Speech, and Signal Processing, 221-224.

[3] Rabiner,L.R. (1977). "On the Use of Autocorrelation Analysis for Pitch Detection", IEEE Transactions on Acoustics,Speech and Signal Processing 25:24-33

[4] Brown,J.C.; Puckette, M.S. (1991) "Calculation of a Narrowed Autocorrelation Function" J.Acoust.Soc.Am., vol.85, nº4: 1595-1601

[5] Rabiner,L.R.; Schafer, R.W. (1978) "Digital Processing of Speech Signals" Prentice-Hall

[6] Doscher, B.M. (1994) "The Functional Unity of the Singing Voice"-2nd ed., The Scarecrow Press, Inc.

[7] W.B.Kuhn. (1990) "A real-time pitch recognition algorithm for music applications", Comp. Music Journal : 60-71

[8] Carey, Michael J.; Parris, Eluned S.; Tattersall, Graham D. (1997) "Pitch Estimation of singing for re-synthesis and musical transcription", ESCA, EuroSpeech97 :887-890

[9] Doval, Boris.; Rodet, Xavier (1991) "Estimation of Fundamental Frequency on Musical Sound Signals", ICASSP, vol.5 :3657-3660

[10] Brown,J.C. (1992) "Musical Fundamental Frequency Tracking using a Pattern Recognition Method", JASA, vol.92, nº3: 1394-1402

[11] Yavelow, C. (1987) "Personal Computers and Music - The State of the Art ", JAES (Journal of the Audio Engineering Society) 35(3):160-193

[12] Sondhi,M.M. (1968) "New methods of pitch extraction ", IEEE Transactions on Acoustics, Speech and Signal Processing 16:262-266

[13] Rabiner,L.R. (1976) "A comparative performance study of several pitch detection algorithms", IEEE Transactions on Acoustics,Speech and Signal Processing 24:399-417

[14] Teixeira, A.; Vaz, F.; Príncipe, J. (1997) " A Software Tool to Study Portuguese Vowels", ESCA, Eurospeech97, Rhodes, Greece, vol.5, pp.2543-2546

[15] Obtained with Yamaha SY77 AWM Samples

Modelos matemáticos e problemas de optimização combinatória*

Iouliia Skliarova, António B. Ferrari

Resumo – **Este artigo apresenta alguns resultados da análise de modelos matemáticos, tais como conjuntos, grafos, matrizes discretas e funções booleanas, utilizados para a especificação e a resolução de problemas de optimização combinatória. É demonstrado que estes modelos são mutuamente convertíveis uns nos outros. São apresentados também exemplos de problemas combinatórios típicos que surgem na área de projecto de dispositivos digitais. A maioria destes problemas podem ser resolvidos com a ajuda dos modelos referidos, através da aplicação de métodos combinatórios. Por fim, analisam-se diferentes possibilidades da implementação de algoritmos combinatórios.**

Abstract – **The paper presents the results of the analysis of different models that are used in problems of combinatorial optimization, such as graphs, sets, discrete matrices, and Boolean functions. It is shown that these models can be mutually converted one into another. Many examples of typical combinatorial tasks, which appear at different steps of the design of digital devices, are considered. The majority of these tasks can be solved with the aid of the presented models and the respective combinatorial methods. Finally, different ways of implementation of combinatorial algorithms are analyzed.**

I. INTRODUCÃO

Os problemas de optimização combinatória surgem em várias áreas práticas tais como desenvolvimento de circuitos digitais, diagnóstico técnico, cartografia, reconhecimento de padrões, planeamento de circulação de transporte, etc. Uma das características distintivas destes problemas é que a sua resolução está ligada com o exame de elementos de um conjunto finito especificado pelos dados iniciais. Os problemas de optimização possuem um conjunto de soluções $S = \{s_1, ..., s_n\}$ no qual é definida uma função de custo $c(s_i)$. Na resolução dum problema de optimização é preciso encontrar uma solução com a função de custo mínima. Os algoritmos desenvolvidos e utilizados para a resolução destes problemas são avaliados com os 2 critérios seguintes: a qualidade da solução e a sua complexidade. De acordo com o critério da complexidade os algoritmos são divididos normalmente em duas classes: polinomiais e exponenciais. Os algoritmos exactos, que garantem a obtenção da solução exacta do problema, têm frequentemente complexidade muito elevada, o que impede a sua aplicação na prática. Por outro lado os algoritmos aproximados nem sempre levam à solução exacta mas asseguram uma aproximação bastante boa a esta.

A resolução de problemas combinatórios é bastante difícil especialmente quando é necessário encontrar uma solução óptima. Contudo a complexidade dos dados iniciais em muitos problemas práticos é limitada, o que poderia levar a concluir que não é preciso desenvolver algoritmos eficientes teoricamente, i.e. basta que os algoritmos sejam eficientes na prática. Na resolução de muitos problemas combinatórios os requisitos de qualidade da solução podem ser diminuidos, i.e. frequentemente a obtenção da solução óptima não é necessária, portanto os métodos exactos podem ser substituidos pelos aproximados que são normalmente bastante mais rápidos. Em muitas situações práticas é possível aplicar métodos de redução que, diminuindo a dimensão do problema em questão, reduzem essencialmente o número de variantes a rever [1]. O desenvolvimento de métodos deste tipo com base na investigação de características de problemas concretos é a área essencial da aplicação da análise combinatória.

Entre vários modelos matemáticos utilizados no desenvolvimento de algoritmos combinatórios destacamos conjuntos, grafos, matrizes e funções lógicas. Notamos que um modelo pode ser transformado noutro.

II. EXEMPLOS DE PROBLEMAS COMBINATÓRIOS

Analisemos alguns exemplos de problemas combinatórios típicos que surgem frequentemente em projecto de dispositivos digitais. Classificamos os problemas de acordo com os modelos matemáticos usados normalmente para a sua especificação. Vários algoritmos de resolução destes problemas podem ser encontrados nas referências indicadas.

* Trabalho financiado com a bolsa da FCT-PRAXIS XXI/BD/21353/99

A. Conjuntos

A.1. Obtenção de partição mínima

O problema é formulado de maneira seguinte. Num conjunto U é necessário encontrar uma partição π que é composta por um número mínimo de blocos. Cada bloco U^i desta partição deve corresponder a determinados requisitos e

$$\pi = \{U^1, ..., U^n\}, \quad U^i \subseteq U, \quad U^i \neq 0, \quad U^i \cap U^j = 0 \ \forall i, j, i \neq j, \quad \bigcup_{i=1}^{n} U^i = U$$

B. Grafos

B.1. Coloração de grafos

Este problema é um dos mais conhecidos problemas combinatórios. Normalmente exige-se colorir um grafo usando um número mínimo de cores atribuindo a todos os vértices adjacentes cores diferentes [2, 3]. A fig. 1 ilustra a coloração mínima de um grafo com 3 cores.

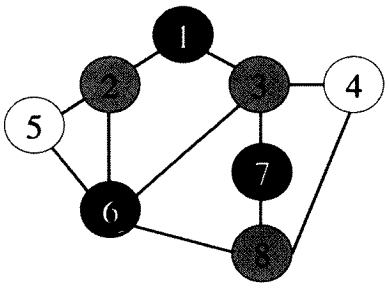


Fig. 1 – Coloração mínima do grafo

B.2. Encontro de caminhos

Problema de encontro de caminhos num grafo também é muito conhecido [3, 4]. Este é formulado de modo seguinte. Há um grafo orientado e interpretado G(E,V,W), um vertice inicial v_b e um vertice final v_e. È preciso encontrar o caminho mais curto ou mais comprido (em termos de pesos dos arcos) do vertice v_b para o v_e. Na fig. 2 é apresentado um grafo interpretado, se $v_b = v_2$ e $v_e = v_5$ então o caminho mais curto é o v_2-v_3-v_5, e o mais comprido é o v_2-v_5.

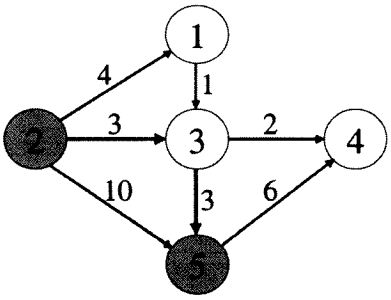


Fig. 2 – O caminho mais curto de v_2 a v_5 é v_2-v_3-v_5

B.3. Partição em cliques

Uma clique dum grafo é um subgrafo completo. Um grafo diz-se decomposto em cliques quando o conjunto dos seus vertices fica dividido em subconjuntos que não se intersectam, cada um dos quais representa uma clique [5]. O problema mais frequente é a procura da partição mínima. O grafo G(V, E) na fig. 3 pode ser dividido em duas cliques $\{v_1, v_2, v_3\}$ $\{v_4\}$.

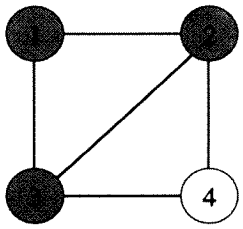


Fig. 3 – Partição em cliques

B.4. Corte

Este problema consiste em cortar um grafo em vários subgrafos de tal maneira que o número de vértices em cada um dos subgrafos não exceda k e o peso total de arestas eliminadas fique minimizado [6]. Na fig. 4 é apresentado um exemplo de corte de um grafo com $k = 2$.

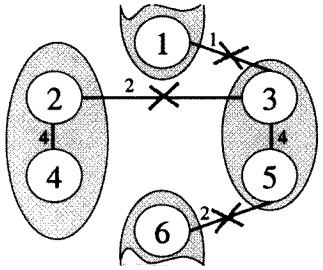


Fig. 4 – Corte do grafo com $k = 2$

C. Matrizes

Aqui consideramos apenas matrizes booleanas e ternárias, utilizadas para a especificação de relações binárias e ternárias arbitrárias.

C.1. Encontro de menores

Muitos problemas estão relacionados com o encontro de um menor (subconjunto de linhas ou colunas) com determinadas características numa matriz. Um dos mais conhecidos é o problema da obtenção da cobertura mínima, onde se procura descobrir um menor mínimo composto pelas colunas (linhas) que contenham em cada linha (coluna) pelo menos um elemento igual a 1 [7]. O problema de determinar o teste diagnóstico mínimo é um outro representante desta classe. Aqui pretende-se descobrir um menor mínimo, cujas linhas sejam todas diferentes (supõe-se que na matriz inicial estas também

são diferentes) [8]. Consideremos a matriz booleana **B**. Neste caso o vector booleano [01100] representa a cobertura mínima (composta pela segunda e terceira colunas da matriz **B**) e o vector [01011] representa o teste mínimo.

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

C.2. Relação de ortogonalidade

Um dos exemplos de problemas deste tipo é o seguinte: é necessário encontrar um vector que seja ortogonal a todas as linhas de uma dada matriz ou concluir que este não existe [7]. Relembramos que dois vectores estão em relação de ortogonalidade se pelo menos numa das suas componentes um vector tem o valor 0 e outro o valor 1.

C.3. Construção de uma nova matriz

Muitos problemas exigem a construção de uma nova matriz **C** que deve estar numa determinada relação com uma dada matriz **B**. Um exemplo desta classe de problemas é o da determinação da base disjuntiva mínima, onde é preciso construir a matriz **C** com o número mínimo de linhas de tal maneira que cada linha da matriz **B** seja igual à disjunção de várias linhas de **C** [1, 7]. Ilustramos este problema com as seguintes duas matrizes **B** e **C**.

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

C.4. Minimização

Quando uma matriz representa a DNF (*Disjunctive Normal Form*) de uma função (ver secção III) surgem frequentemente problemas de minimização da mesma. Por exemplo, para uma dada matriz ternária é preciso encontrar uma outra matriz equivalente com o número de linhas minimizado. Assim, a seguinte matriz **T** representa uma DNF inicial e a matriz **U** representa a DNF mais curta equivalente à inicial [7].

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & - \\ - & 0 & 1 & 1 \\ - & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} 1 & 0 & - & - \\ 0 & 0 & 1 & - \end{bmatrix}$$

D. Funções booleanas

D.1. Minimização

Relembramos que qualquer função booleana pode ser representada em DNF, i.e. em forma de disjunção de um conjunto finito de conjunções elementares diferentes. O problema de obtenção de DNFs mais simples para uma dada função booleana é bastante complicado. Normalmente por critério de simplicidade toma-se o número total de argumentos (quanto à procura da DNF mínima) ou o número de conjunções elementares (quanto à procura da DNF mais curta) [7, 9]. Por exemplo, para a função $f = x_1x_2 \vee \bar{x}_1x_2 \vee \bar{x}_1\bar{x}_2$ obtemos a seguinte DNF mais curta $f = \bar{x}_1 \vee x_2$. Note-se que o problema de encontro da DNF mais curta coincide com o da determinação da matriz ternária mínima (com o número de linhas mínimo) equivalente à dada.

D.2. Decomposição

Este problema consiste na representação de uma função booleana em forma de composição de várias funções booleanas cada uma com um menor número de argumentos [7].

D.3. Satisfabilidade

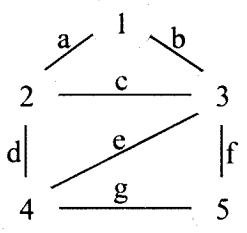
Este problema consiste na verificação da existência de valores de argumentos tais que tornem a função igual a 1 [9]. Tautologia é um problema complementar que verifica se a função é equivalente a 1, i.e. adquire o valor 1 independemente dos valores dos seus argumentos. Notemos que este problema coincide com o da determinação de um vector que é ortogonal a cada linha da matriz que representa a DNF da função (caso seja impossível encontrá-lo, pode-se concluir que a função é equivalente a 1).

III. TRANSFORMAÇÕES MÚTUAS DE MODELOS

Com a ajuda de matrizes lógicas pode-se formular muitos problemas combinatórios e construir algoritmos para a sua resolução. As matrizes possuem uma estrutura que as torna bastante convenientes para o armazenamento e processamento em computador. Portanto mostramos como especificar grafos e funções booleanas através das matrizes respectivas.

Um grafo G(V, E) pode ser representado em forma de uma matriz de adjacência ou de incidência. Uma matriz de adjacência **A** é uma matriz quadrada de dimensão |V|*|V|, onde cada elemento a_i^j é igual a 1 caso o vértice *i* seja adjacente ao vértice *j*, e é igual a 0 caso contrário. Uma matriz de incidência **I** é uma matriz com as dimensões |V|*|E|. Para um grafo não orientado cada elemento (i, j)

da matriz é igual a 1 caso a aresta *j* seja incidente ao vértice *i*, e é igual a 0 caso contrário. Para um grafo orientado um elemento em posição (i, j) é igual a 1 se o vértice *i* é a origem do arco *j*, é igual a -1 se o vértice *i* é o destino do arco *j*, e é igual a 0 caso contrário. Na fig. 5 é representado um grafo não orientado e as suas matrizes de incidência e de adjacência.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Fig. 5 – Grafo e as matrizes de incidência e de adjacência respectivas

As funções booleanas também podem ser representadas em forma de matrizes booleanas e ternárias. Quando uma função booleana está em DNF então as colunas da matriz correspondem aos argumentos da função e as linhas determinam as conjunções elementares. Por exemplo, para a função seguinte é bastante fácil construir as respectivas matrizes booleana e ternária:

$$x_1x_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2\bar{x}_3 = x_1 \vee x_2x_3$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{T} = \begin{bmatrix} 1 & - & - \\ - & 1 & 1 \end{bmatrix}$$

Notemos que a matriz booleana **B** pode ser obtida a partir da matriz ternária **T** através da substituição das componentes que possuem valor "-" com todas as combinações possíveis de 0 e 1, e da eliminação posterior de linhas iguais.

Um sistema de funções booleanas **y=f(x)** pode ser especificado com a ajuda de um par de matrizes **X** e **Y**. Se as funções booleanas estão em DNF então as colunas da matriz **X** determinam um sistema de intervalos juntamente com as conjunções elementares respectivas, e as linhas da matriz **Y** correspondem às DNFs, i.e. cada elemento y_i^j é igual a 1 quando a conjunção elementar *j* entra na DNF *i*.

O seguinte exemplo ilustra a representação de um sistema de DNFs com as matrizes **X** e **Y**:

$$y_1 = \bar{x}_1 \vee x_2x_3$$
$$y_2 = x_1\bar{x}_2\bar{x}_3 \vee x_2x_4$$
$$y_3 = \bar{x}_1 \vee \bar{x}_1\bar{x}_4 \vee x_2x_3$$

$$\mathbf{X} = \begin{bmatrix} 0 & - & 1 & - & 0 \\ - & 1 & 0 & 1 & - \\ - & 1 & 0 & - & - \\ - & - & - & 1 & 0 \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

IV. APLICAÇÕES PRÁTICAS DE PROBLEMAS COMBINATÓRIOS

Muitos problemas práticos reduzem-se à resolução de vários problemas combinatórios típicos. Consideremos alguns exemplos destes problemas práticos destacando os que surgem na área do desenvolvimento de circuitos digitais.

É conhecido que uma das fases da síntese e optimização arquitectural consiste na distribuição de operações no tempo, i.e. na determinação do início da execução de cada operação com base na análise da sequência e interacção de todas as operações [9]. A possibilidade da resolução deste problema com dadas restrições temporais pode ser verificada aplicando o método da determinação do caminho mais comprido num grafo. Consideremos um grafo orientado e interpretado, cujos vértices representam as operações e cujos arcos correspondem às dependências entre as operações. O peso de cada arco é igual ao atraso da operação colocada na sua origem. Os arcos complementares especificam as restrições temporais e as restrições máximas $U_{ij}^{\max}$ definem o intervalo máximo de tempo que pode decorrer entre duas operações *i* e *j*. Sendo assim, para que o problema tenha solução, o caminho mais comprido (o caminho com o peso maior) neste grafo entre os vértices v_i e v_j deve ser $\leq U_{ij}^{\max}$.

Na fase seguinte da síntese arquitectural efectua-se a atribuição das operações aos recursos existentes. Um dos objectivos a atingir nesta fase consiste na diminuição da área do circuito permitindo que várias operações distribuidas no tempo utilizem o mesmo hardware [9]. Neste caso duas operações chamam-se compatíveis caso não executem simultâneamente e possam ser implementadas em recursos do mesmo tipo. Constroi-se o grafo de compatibilidade, cujos vértices correspondem às operações e cujas arestas ligam os pares de operaçãos compatíveis. Sendo assim o problema da distribuição óptima de recursos reduz-se à partição do grafo num número mínimo de cliques.

Para a minimização do número de condições lógicas numa FSM (Máquina de Estados Finitos) usa-se a informação complementar sobre as suas alterações possíveis no processo de execução. Designemos por • (x_i) o conjunto de estados cujas transições dependem da condição lógica x_i, i=1,..,L. Colocamos em correspondência a cada conjunto • (x_i) um vector $\mathbf{g}_i = (g_{i1},...g_{iL})$, e representamos todos os vectores por uma matriz **G**, onde cada elemento g_{ij} é igual a 1 se em todos os estados de $A(x_i)$ $x_j = 1$, g_{ij} é igual a 0 se em todos os estados de $A(x_i)$ $x_j = 0$, e g_{ij} é igual a "-" se nos estados de $A(x_i)$ a condição x_j pode tomar tanto o valor 1 como o valor 0. Com base na matriz **G** construimos um grafo com os vértices correspondentes às condições lógicas. Os dois vértices x_i e x_j são ligados por uma aresta caso $g_{ij} \neq g_{ji}$. Depois de colorir o grafo com o número mínimo de cores podemos unir (de um modo especial [10]) aquelas condições lógicas que correspondem aos vértices da mesma cor.

Na fase da síntese lógica de circuitos efectua-se a minimização de funções booleanas. O objectivo é construir um sistema de funções que sendo equivalente ao inicial corresponde aos requisitos especificados (por exemplo, a complexidade mínima ou o atraso de propagação mínimo, etc.).

O método da determinação das partições mínimas de um conjunto aplica-se na fase de mapeamento de elementos lógicos em células de uma dada biblioteca. Por exemplo, temos um sistema de funções booleanas em DNF com L argumentos, N funções e B conjunções diferentes. Designamos com L^i, i=1, ..., B, o número de argumentos na conjunção *i*, então $L_{max} = \max_i L_i$. É preciso implementar este sistema de funções booleanas em PLAs(s,t,q) (PLA com s entradas, t saídas e q linhas intermédias) e L > s, N > t, B > q, L^{max} <= s. Consideremos o conjunto de todas as conjunções elementares diferentes $E=\{e_1,...,e_B\}$. Então o problema reduz-se ao encontro de partição mínima no conjunto E que para cada bloco E^u faz cumprir as seguintes restrições: o número de argumentos em E^u < s, o número de funções que dependem de E^u <= t e $|E^u| < q$ [11].

Frequentemente surge o problema de decomposição de uma FSM em várias sub-FSMs mais simples. Um dos objectivos que se coloca consiste em minimizar a quantidade de ligações entre sub-FSMs, o que é equivalente à minimização do número de transições entre os estados pertencentes a sub-FSMs diferentes [6, 12]. Para resolver este problema é preciso cortar o grafo da FSM inicial em subgrafos minimizando a quantidade de sub-FSMs e o número de arestas eliminadas.

Suponhamos que temos um conjunto de vectores booleanos e precisamos de construir um array AND (o mais simples possível) que implemente estes vectores. Pede-se também arranjar os vectores de entrada respectivos que serão transformados em dados vectores de saída. Notamos que a estrutura de um array AND pode ser especificada com a ajuda de uma matriz **B**, e a sua funcionalidade descreve-se com a equação $\mathbf{Y} = \overline{\mathbf{B} \times \mathbf{X}}$ [7]. Sendo assim, temos de encontrar as matrizes booleanas **B** (com o número mínimo de colunas *k*) e **X** (com o mesmo número de linhas *k*) que satisfazem a equação $\mathbf{Y} = \overline{\mathbf{B} \times \mathbf{X}}$. Substituimos a matriz **Y** com a sua negação **Y'** e analisamos a equação $\mathbf{Y'} = \mathbf{B} \times \mathbf{X}$. Neste caso cada coluna da matriz **Y'** é igual à disjunção de várias colunas da matriz **B** (a matriz **X** indica quais). Portanto o problema reduz-se à determinação da base disjuntiva mínima.

Às vezes surge o problema de determinação de causas a partir dos efeitos que estas provocam (por exemplo, em diagnóstico). Neste caso as causas abrangem os fenómenos cuja observação é difícil ou mesmo impossível. Por outro lado, os efeitos são directamente detectáveis. Contudo a detecção dos efeitos tem um custo associado portanto o seu número deve ser restringido. Sendo assim, surge o problema de determinar o subconjunto mínimo de efeitos cuja observação permite descobrir a respectiva causa. A relação entre os conjuntos de causas e efeitos pode ser representada em forma de uma matriz booleana **X** onde cada elemento x_i^j é igual a 1 se a causa *j* provoca o efeito *i*, e é igual a 0 caso contrário. Então o problema reduz-se a determinar o teste diagnóstico mínimo para a matriz **X** [7].

V. MÉTODOS DE RESOLUÇÃO DE PROBLEMAS COMBINATÓRIOS

Na sua forma mais simples a resolução de um problema combinatório consiste no exame de todas as variantes possíveis. Cada variante é examinada uma por uma e logo que se verifique que uma destas é a solução, o processo de pesquisa termina. Em algumas situações este método simples é o único que pode ser aplicado. Por exemplo, é preciso encontrar o número mínimo num conjunto de *n* números. É evidente a inevitabilidade de examinar os *n* elementos.

O conjunto entre cujos elementos se procura a solução é finito, portanto ao examinar exaustivamente os seus elementos podemos sempre encontrar a solução ou concluir que esta não existe. Portanto qualquer problema lógico pode ser resolvido em princípio, mas isto não significa que possa ser resolvido em tempo aceitável. Por exemplo, sabemos que a solução de um problema pode ser representada por uma função booleana desconhecida de 10 variáveis. Existem 2^{2^n} de funções booleanas diferentes de *n* argumentos, logo o método de revisão completa vai exigir o exame de $2^{2^{10}} = 2^{1024} > 10^{256}$ variantes!

Sendo assim o método de revisão completa só pode ser aplicado em situações muito simples. No caso geral devemos reduzir de alguma maneira a quantidade de

variantes a considerar utilizando informação complementar sobre o problema em causa.

Uma das técnicas gerais da resolução de problemas combinatórios consiste na construção de uma árvore de pesquisa. A raíz da árvore corresponde à situação inicial e os restantes vértices representam todas as situações que podem ser atingidas no processo de pesquisa. Os arcos correspondem aos passos do processo decisório. Na maioria dos casos a árvore é desconhecida inicialmente, e só se constroi durante o processo de pesquisa da solução. Ao atingir uma dada situação devemos determinar todas as variantes possíveis da continuação da pesquisa, i.e. todos os arcos de saída. É evidente que é necessário reduzir de algum modo o seu número para que a solução seja encontrada o mais depressa possível. Normalmente esta redução é baseada na eliminação dos passos que não vêm a alterar a situação actual, na rejeição das variantes já analisadas anteriormente e em parar de procurar aquelas soluções cujo custo no ponto actual já se tornou maior do que o custo de alguma das soluções já encontradas.

Este método simples sendo universal é pouco eficaz, i.e. só pode ser aplicado na prática caso os dados iniciais do problema sejam bastante simples. Uma das técnicas que aumentam significativamente a eficâcia deste método é a redução. Nesta técnica uma situação actual substitui-se por uma outra situação mais simples, o que permite reduzir a quantidade de cálculos necessários para analisar o conjunto de variantes resultantes desta situação. As maneiras concretas de redução são extremamente diversas. Por exemplo, é preciso encontrar a cobertura mínima de uma matriz booleana. Neste caso a matriz inicial pode ser diminuida através da eliminação de linhas dominantes a colunas dominadas (se procuramos a cobertura composta por colunas) [7], i.e. a simplificação da situação consiste na diminuição da matriz. Um exemplo clássico desta técnica é o da determinação das raízes de uma equação pelo método da bisecção do intervalo, em que a simplificação da situação consiste na diminuição do intervalo no qual se procura a raíz da equação.

Contudo nem sempre se pode recorrer à técnica de redução, sendo necessário considerar também o processo da decomposição de situações. Neste caso uma das situações torna-se crítica e por esta razão deve ser substituida por um conjunto de situações mais simples, cada uma das quais deve ser examinada. É de notar que convém decompor no menor número possível de situações mais simples. O processo de decomposição é representado por uma árvore de pesquisa de cujos vértices saem vários arcos. Por exemplo, temos uma função booleana $f(x_1,...,x_n)$ e pede-se para encontrar um vector booleano $\mathbf{x}$ que faz com que a função fique igual a 1. De acordo com o teorema de Shannon podemos decompor uma situação corrente em duas, escolhendo uma variável x_i e supondo que numa situação esta variável fica igual a 1 e noutra fica igual a 0:

$$f(x_1, ..., x_n) = x_i f(x_1, ..., 1, ..., x_n) \vee \overline{x}_i f(x_1, ..., 0, ..., x_n)$$

A seguir analisamos a primeira situação. Se esta nos levar à solução, o processo da pesquisa para; caso contrário teremos de examinar a segunda situação.

As vezes consegue-se ordenar as situações para as quais se decompõe a situação corrente, i.e. pode-se analisar algumas características do problema e com base nesta informação escolher uma situação preferível que tem maiores probabilidades de nos levar à solução. Isto melhora significativamente a eficiência dos algoritmos. Por exemplo, quando procuramos um vector ortogonal a cada linha de uma dada matriz e não podemos reduzir uma situação actual, a única saída consiste no exame sequencial dos valores possíveis de todas as componentes desconhecidas deste vector. Contudo, para encontrar a solução o mais rápido possível, recomenda-se escolher primeiro a componente que corresponde à coluna mais determinada da matriz, i.e. à coluna que tem o número mínimo de valores "-"[7].

Em muitos problemas práticos nem sempre é possível encontrar a solução óptima, pelo que os métodos aproximados têm ampla difusão. Nos métodos aproximados a busca que se efectua é bastante reduzida devido à eliminação das variantes menos promissoras. Contudo entre as variantes eliminadas pode estar a óptima. A eficiência dos métodos aproximados é avaliada segundo a qualidade da solução obtida e o tempo gasto para a obter.

VI. IMPLEMENTAÇÃO DE ALGORITMOS COMBINATÓRIOS

Os algoritmos de resolução de problemas combinatórios podem ser implementados em software, em hardware ou de uma maneira mista. Devido à necessidade do exame de muitas variantes no processo de solução, a implementação em software requer consumos significativos de tempo. Os modelos considerados são de bastante difícil representação na memória dos computadores, portanto devemos aplicar métodos especiais para a organização, armazenamento e processamento de dados específicos, tais como grafos, etc.

Pode-se implementar os algoritmos combinatórios em hardware mas isto também revela algumas dificuldades. Por um lado cada problema é único em termos do conjunto de operações que requer. O tipo de operações é normalmente bastante reduzido e estas são aplicadas às estruturas uniformes de dados (por exemplo, linhas e colunas de matriz). Por outro lado, é muito difícil propor algum dispositivo universal que possa ser usado para a resolução de qualquer problema arbitrário. Isto inviabiliza a construção de hardware dedicado porque este seria muito caro e utilizado de uma maneira ineficiente.

Contudo o uso de dispositivos reconfiguráveis, tais como FPGAs (*Field Programmable Gate Arrays*), seria justificado e eficaz. É evidente que a implementação em hardware reduz significativamente o tempo necessário para a resolução de problemas respectivos. Neste caso podemos usar as FPGAs de duas maneiras diferentes:

1. Como um co-processador reconfigurável que pode ser reprogramado para a resolução de problemas diferentes. Sendo assim, a reconfiguração consiste em implementar um conjunto de operações reduzido. A construção de um co-processador deste tipo não é facil, mas é possível [13, 14].
2. Como um dispositivo especial que mapeia directamente uma determinada estrutura de dados. Por exemplo, um grafo pode ser modelado por um sistema de flip-flops interligados que correspondem aos vértices do grafo. Uma ligação entre dois flip-flops representa um arco entre os vértices respectivos. Tais estruturas permitem resolver muitos problemas combinatórios, por exemplo os de encontro de caminhos num grafo. É possível especificar as estruturas de maneira a poderem ser guardas numa biblioteca expansível.

Os algoritmos combinatórios podem também ser realizados de uma maneira mista, sendo umas partes implementadas em software e outras em hardware, i.e. num dispositivo baseado em FPGA e ligado ao barramento do computador. Uma estrutura deste tipo pode aproveitar das vantagens de implementações diferentes.

VII. CONCLUSÕES

Neste artigo apresentamos alguns exemplos de problemas combinatórios típicos e consideramos vários modelos matemáticos usados normalmente para a sua especificação e resolução. Foi demonstrado que estes modelos são mutuamente convertíveis. Analisámos alguns exemplos práticos que surgem na área de projecto de dispositivos digitais e requerem a resolução de problemas combinatórios. Consideraram-se também os principais métodos utilizados na fase de desenvolvimento de algoritmos de resolução de problemas combinatórios. Estes algoritmos podem ser realizados quer em software, quer em hardware. A implementação em software requer normalmente muitos recursos do computador e consome bastante tempo. Existem muitos problemas que nem sequer podem ser resolvidos devido à sua complexidade elevada (a maioria dos problemas combinatórios não são resolúveis em tempo polinomial). A implementação em hardware permite reduzir o tempo necessário para a execução, mas neste caso cada aplicação exige um circuito dedicado. Para resolver este problema pode-se usar hardware reconfigurável, tal como FPGAs. Concluímos que o melhor resultado será obtido caso combinemos de uma maneira razoável a implementação em software, com a utilização de hardware reconfigurável baseado em FPGAs.

REFERÊNCIAS

[1] • .D.Zakrevski, "Combinatorial Theory of Logical Design", Avtomatica i Vychislitelnaya Tekhnica, Nº2, 1990, pp. 68-79 (em russo).

[2] A.D.Zakrevski, "Graph Coloring and Decomposition of Boolean Functions ", Logical Design, Nº. 5, 2000 (em russo).

[3] R.Diestel, "Graph Theory", 2nd ed., Springer, 2000, p.310.

[4] A.Aho, J.Hopcroft, J.Ullman, "Data Structures and Algorithms", Addison-Wesley, 1983.

[5] M.Golumbic, "Algorithmic Graph Theory and Perfect Graphs", Academic Press, 1980.

[6] V.A.Sklyarov, "Synthesis of Finite State Machines Based on Matrix LSI", Minsk: Science and Techniques, 1984 (em russo).

[7] A.D.Zakrevski, "Logical Synthesis of Cascade Networks", Moscow: Nauka, 1981 (em russo).

[8] A.D.Zakrevski, "Combinatorial Problems over Logical Matrices in Logic Design and Artificial Intelligence", Electrónica e Telecomunicações, vol. 2, Nº 2, 1998, pp. 261-268.

[9] Giovanni De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, Inc., 1994.

[10] S.I.Baranov, L.N.Zhuravina, V.A.Skliarov, "Computer Aided Design", Minsk, High School, 1981 (em russo).

[11] S.I.Baranov, V.A.Sklyarov, "Digital Devices Based on Programmable Matrix LSI", Moscow: Radio and Communications, 1986 (em russo).

[12] G.D.Hachtel, F.Somenzi, "Logic Synthesis and Verification Algorithms", Kluwer Academic Publishers, 1996.

[13] I.Skliarova, A.B.Ferrari, "Exploiting FPGA-based Architectures and Design Tools for Problems of Reconfigurable Computations", Proceedings of the SBCCI 2000 XIII Symposium on Integrated Circuits and System Design, Manaus, Brazil, September 2000, pp. 347-352.

[14] I. Skliarova, A.B.Ferrari, "Development tools for problems of combinatorial optimization", Proceedings of the 4th Portuguese Conference on Automatic Control (CONTROLO'2000), Guimarães, Portugal, October 2000, pp. 552-557.

Diário da Assembleia da República Electrónico.

Joaquim Sousa Pinto, Pedro Almeida

Resumo – **Neste artigo apresentamos o projecto do Diário da Assembleia da República Electrónico, que foi desenvolvido no âmbito da disciplina de projecto do curso de Licenciatura em Engenharia Electrónica e Telecomunicações , leccionado na Universidade de Aveiro.**

Abstract –**This article is to present the project Diário da Assembleia da República Electrónico, that was developed as part of the course Engenharia Electrónica Telecomunicações at the University of Aveiro.**

I. INTRODUÇÃO

Hoje em dia o computador é cada vez mais uma ferramenta de trabalho que não podemos desprezar.

O seu uso como fonte de informação está a implementar-se rapidamente e torna-se necessário criar sistemas de informação avançados.

Estes sistemas podem funcionar quer a nível local, dentro de uma intranet, quer a nível global, através de um site da Internet.

Um aspecto interessante destes sistemas é o facto de ser distribuído, permitindo assim que a informação esteja dividida por vários computadores.

No âmbito deste trabalho foi desenvolvido um sistema que possibilita a pesquisa de expressões dentro de um conjunto de documentos. Após apresentada a listagem dos documentos que satisfazem a expressão de pesquisa, podemos então seleccionar um dos documentos e visualizar o seu conteúdo. Quando se apresenta o documento as palavras que foram utilizadas na expressão de pesquisa aparecem destacadas para possibilitar uma rápida identificação do texto que se desejava encontrar.

O sistema permite também a possibilidade de ver a imagem digitalizada do documento.

II. IMPLEMENTAÇÃO

Para construir este sistema foi necessário utilizar várias linguagens de programação e de serviços que são disponibilizados pela MICROSOFT, estando estes incorporados no sistema operativo WINDOWS 2000 SERVER. De seguida vamos descrever muito brevemente as tecnologias que foram utilizadas para uma melhor compreensão do trabalho.

Tecnologias utilizadas:

- HTML (Hypertext Markup Language)

  O HTML é uma linguagem de marcas utilizada na World Wide Web que permite visualizar hipertexto. É necessário um programa, um browser, para poder interpretar e apresentar correctamente o conteúdo de um documento HTML.

- URL (Uniform Resource Locator)

  O URL indica-nos o endereço de um recurso acessível na rede.

- ASP (Active Server Pages)

  ASP's é uma tecnologia que permite que se possam executar scripts do lado do servidor. Do lado do cliente visualiza-se somente HTML puro, permitindo assim que os detalhes da implementação de um sistema só sejam "vistos" do lado do servidor. Através do uso de ASP's podemos aumentar as potencialidades de um servidor web pois é possível comunicar com bases de dados registadas no servidor, executar scripts, e utilizar objectos criados pelo programador.

- Linguagens de Script (JavaScript, VBScript)

  O JavaScript é uma linguagem de programação que permite dar alguma funcionalidades a uma página HTML. É uma linguagem derivada da linguagem C, não possuindo, no entanto, a rigidez associada com uma "typed language", em que as variáveis possuem um tipo definido e são previamente definidas. A capacidade de processamento de JavaScript depende dos browsers. Com JavaScript podemos, por exemplo, validar informação que é introduzida em formulários, libertando assim o servidor de verificar o conteúdo dos vários campos do formulário. Este processo permite que o servidor não seja constantemente sobrecarregado com pedidos do cliente para validar dados. O JavaScript tem

muitas mais funcionalidades como aproveitar certos eventos para alertar o utilizador de alterações que estão a ser feitas no documento, ou simplesmente animar um documento.
O VbScript é também uma linguagem de script derivada da linguagem Visual Basic.

- COM/ACTIVEX™

COM (Component Object Model) é uma arquitectura de software standard baseada em interfaces, e desenhada para ter código separado em objectos "self-contained". Estes componentes binários (nível mais baixo de troca de informação/ dados /código) estão preparados para funcionar com qualquer cliente COM. A tecnologia COM é independente da linguagem de programação. Quando componentes COM trabalham entre vários computadores ligados em rede, os componentes são denominados como distributed COM (**DCOM**).

Sendo assim, **ActiveX** é uma tecnologia que integra componentes de software, num ambiente de rede, independentemente da linguagem em que foram criados. Um control ActiveX, anteriormente conhecido como control OLE ou control OCX, é implementado tendo como suporte a tecnologia COM. No nosso caso, desenvolvemos componentes em C++, e integramo-los em páginas web.

Utilizando controls ActiveX, permitiu encapsular o código, protegendo deste modo a propriedade intelectual dos componentes desenvolvidos.

Para desenhar componentes ActiveX, utiliza-se a ajuda de Class Libraries. Existem duas livrarias para desenhar controls ActiveX.

  - Microsoft foundation classes (MFC)
  - Active Template Library (ATL) (O objectivo desta livraria é executar apenas o código para a aplicação, sendo por isso desnecessário, associar-se a outras livrarias.)

- IIS (Internet Information Service)

O IIS permite criar um site da Internet com todas as funcionalidades que lhe estão associadas. Através da utilização deste serviço é possível utilizar ASP's na construção de sites WEB.

Este serviço disponibiliza ainda a possibilidade de criar directorias virtuais, que podem estar localizadas em outro disco ou computador, que podem ser utilizadas com se fizessem parte do nosso site.

- Indexing Service

Este serviço permite indexar um conjunto de ficheiros e formar um catálogo com os mesmos. Após criado o catálogo e com ajuda das linguagens de programação mencionadas acima , podemos pesquisar o mesmo. Este serviço pode ser configurado de modo a permitir que nem todas as palavras dentro de um documento façam parte do catalogo.

Quando efectuamos uma pesquisa no catálogo o serviço indica-nos os documentos que satisfazem a expressão de pesquisa utilizada.

III – COMPONENTES

Foi necessário contruir componetes que permitissem a contrução do site e a gestão do mesmo. Para a gestão do sistema foi construido um componente cujo nome é TXT2HTM. Para a construção do site foram desenvolvidos outros dois componetes com o nome SHOWPAGE, e MAGICVIEWER.

A. Componente TXT2HTM.

O projecto **Diário da Assembleia da República electrónico**, para o seu funcionamento, necessita das diversas páginas do diário no formato HTML, para proceder à sua indexação, e à posterior visualização.

Para que através de uma simples página se possa novamente reconstruir o diário, as páginas no formato HTML, deverão ter como marcas informativas (*meta tags),* o número do diário, da legislatura, da série, da sessão, e as datas da sessão e do diário. As finalidades destas marcas é facilitar a acessibilidade e organização nas diferentes páginas de cada diário. Estas marcas possibilitam também a navegação dentro de um diário nas diferentes páginas, e dentro das diversas legislaturas, sessões e séries nos diferentes diários.

Como os diários da Assembleia da República são públicados em papel, irá ser necessário convertê-los para ficheiros HTM. Assim o primeiro passo será converter as páginas do diário em papel, para o formato digital, ou seja digitalizar as páginas para obter uma imagem e depois fazer um reconhecimento de caracteres, converte-los assim, para ficheiros de texto. Os formatos fornecidos para a imagem e o texto são os formatos TIF e TXT, respectivamente.

O segundo passo, irá ser realizado com a ajuda do programa ***TXT2HTM***, que como o próprio nome indica, será a partir dos ficheiros de extensão TXT, anteriormente referidos, criar ficheiros HTM(L), onde serão inseridas as *metas tags* legislatura, série, sessão, número do diário, data da sessão e data do diário.

Os objectivos desta componente, são então os seguintes:

- Conversão de um ficheiro no formato TXT para um no formato HTM, ou seja, pegar na informação do

ficheiro TXT, e colocá-la dentro das marcas <BODY> </BODY> de um ficheiro HTM, com o mesmo nome.

- Inserir *meta tags* no ficheiro HTM, com a informação sobre o diário (número, data sessão,...).

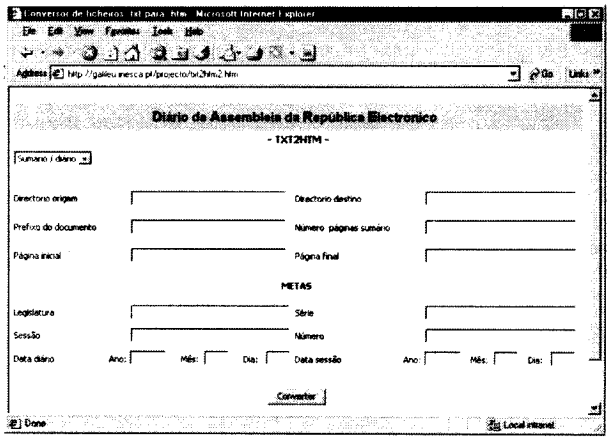


Fig.1 Interface do programa TXT2HTM.

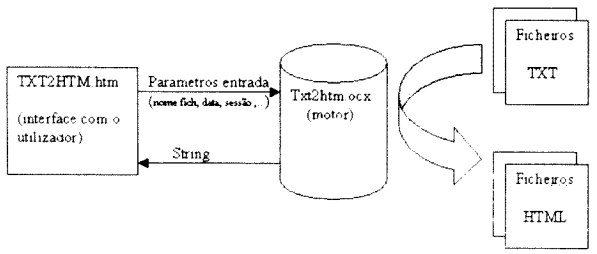


Fig.2. Modelo do Objecto.

B. Componente SHOWPAGE.

No âmbito do projecto **Diário da Assembleia da República Electrónico**, sempre que efectuamos uma pesquisa, é necessário apresentar o resultado da mesma de uma forma que o utilizador possa identificar rapidamente no texto pelo que pesquisou. Para tal, foi desenvolvido um módulo de software construído em visual C++, sendo este um projecto do tipo Active Template Library. Este tipo de projecto permite criar um ActiveX que vai ser incorporado dentro de uma ASP, permitindo um melhor funcionamento do sistema. Este ActiveX, em conjunto com uma stylesheet, permite realçar as palavras que desejamos encontrar. O ActiveX também permite obter valores que vão ajudar na construção de um cabeçalho que identifica o documento e que tem valores que permitem ajudar na navegação ao longo dos diários pesquisados.

Mais tarde, esta página terá também opções avançadas que permitirão imprimir um diário por completo ou seleccionar as páginas a imprimir.

Os objectivos do componente showpage são:

- Realçar uma ou mais palavras que foram utilizadas na expressão de pesquisa.

- Dado um documento de HTML com uma formatação predefinida, retirar deste documento informações que são relevantes para a apresentação da página em causa, e obter dados que permitam certas facilidades de navegação na apresentação da página. Estes dados são obtidos a partir do cabeçalho do documento, onde se encontram campos do tipo <META> com descrição dos dados que pretendemos obter (data , numero, ...).

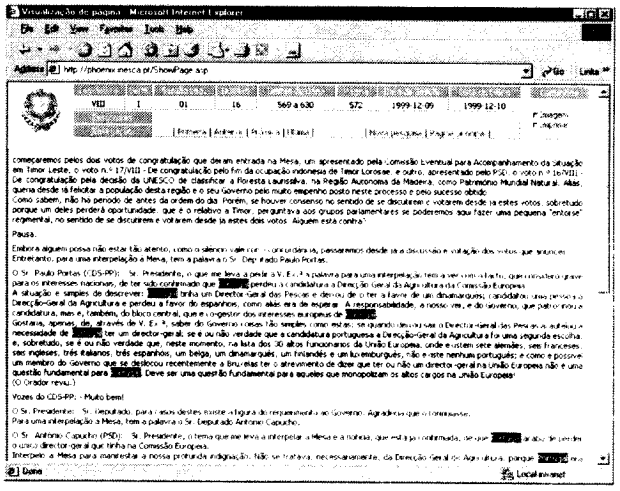


Fig. 3. Exemplo da página SHOWPAGE.ASP.

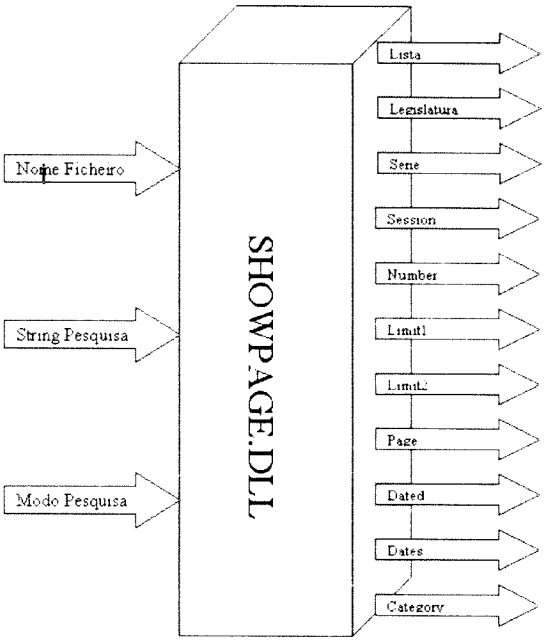


Fig. 4.Modelo do Objecto.

Este módulo de software é um ActiveX que foi desenvolvido através de uma ATL. Depois de compilado o programa, gera-se uma dll que é necessário registar no sistema, para que seja possível utilizar o objecto que foi criado. Normalmente este passo é executado automaticamente no Visual C++ 6.0, mas caso seja necessário instalar este sistema noutro computador é necessário recorrer a um ficheiro de instalação que regista a dll no sistema em causa.

O diagrama representa o modelo do objecto que foi criado. Este objecto representa o funcionamento do programa, tendo este parâmetros de entrada e parâmetros de saída. Associado a este objecto encontram-se métodos que podem ser invocados sobre ele, devolvendo cada método informação relativa a pedidos sobre o documento manipulado.

Para melhor compreensão do funcionamento deste sistema vamos descrever de seguida os vários métodos associados ao módulo.

- Método lista

  Este método inicialmente preenche tabelas com expressões que vão ser pesquisadas dentro do documento HTML. Depois de preenchidas, são comparadas as palavras armazenadas, com as palavras do texto em que estamos a pesquisar. Após concluída a pesquisa, retorna o texto que foi pesquisado de forma a que seja possível realçar as palavras que se encontravam na tabela.

- Métodos legislatura, série, session, number limits1, limits2, page, dated, dates e category.

  O funcionamento destes métodos é muito semelhante. Eles procuram o meta correspondente no ficheiro HTML e retornam o valor que se encontra nesse campo.

Após concluída a aplicação esta é inserida em uma página ASP. A inserção do objecto na página asp, é feita através do seguinte código:

```
Set objecto=
Server.CreateObject("showpage.mostra.1")
```

Após criado o objecto, para aceder aos vários métodos do mesmo, utilizamos a notação:

```
text_doc=objecto.lista(Nome_ficheiro,Str,Modo
_Pesquisa)
legis=objecto.legislatura()
serie=objecto.serie()
sess=objecto.session()
number=objecto.number()
limits1=objecto.limit1() * 1.0
limits2=objecto.limit2() * 1.0
page=objecto.page() *1.0
dated=objecto.dated()
dates=objecto.dates()
category=objecto.category()
```

Vamos agora descrever as funcionalidades associadas à página *showpage.asp* .

A maior parte das funcionalidades desta página estão associadas à tabela de navegação. Esta permite que o utilizador se possa situar dentro de um diário e que consulte o mesmo de uma forma eficaz, já que tem opções de navegação. Caso o utilizador pressione uma das opções do cabeçalho, como por exemplo *página seguinte*, vão ser introduzidos novos valores em campos escondidos da página, e vamos voltar a chamar a mesma página, showpage.asp, variando os parâmetros necessários para aparecer a página que pretendemos consultar. Na situação mencionada atrás, em que o utilizador pressiona o texto página seguinte, o utilizador vai voltar a ver a página showpage.asp, mas agora com o documento da página seguinte ao que estava a consultar. Caso apareçam nesse documento palavras que o utilizador pesquisava estas vão aparecer realçadas, isto é, a expressão de pesquisa mantém-se quando o utilizador navega ao longo do documento.

No caso de o utilizador pressionar página anterior o funcionamento é muito semelhante.

Se o utilizador pressionar página inicial então é apresentada a primeira página do diário e através de uma função em JavaScript desactivamos a opção de ir para a página anterior visto que não tem sentido voltar atrás da primeira página.

Quando o utilizador pressiona ultima página o funcionamento é semelhante, isto é desactivamos a opção de ir para a página seguinte, já que neste caso não faz sentido ir para a página seguinte.

Caso o utilizador carregue sobre imagem então vai ser chamada outra ASP que se encarrega de apresentar a imagem do diário original.

C. Componente MAGIC VIEWER.

No âmbito do projecto **Diário da Assembleia da República Electrónico** sempre que efectuamos uma pesquisa, e visualizamos uma página, temos uma opção que nos permite ver o documento original digitalizado. Para que tal seja possível foi necessário construir um ActiveX que nos permitisse visualizar TIF's. Isto deve-se ao facto de o Internet Explorer 5.0 não possuir filtros para

abrir imagens deste formato. Este ActiveX é posteriormente incluído dentro de uma página da WEB, permitindo assim que possamos observar ficheiros deste formato dentro do Internet Explorer 5.0 . Sendo assim torna-se mais fácil a visualização dos documentos, sem ter de recorrer a programas externos para abrir as imagens. A utilização deste formato de imagem deve-se ao facto de este ser de qualidade bastante elevada e de tamanho reduzido, permitindo assim poupar recursos quer a nível de espaço de disco quer a nível de tempo de processamento. Este facto é bastante importante já que o número de imagens guardadas em disco é muito elevado. O facto deste programa obter as imagens a partir de um URL permite que o sistema seja distribuído, podendo as imagens estar guardadas numa máquina distinta de onde guardamos os ficheiros de texto.

Os principais objectivos deste componente, são então os seguintes:

- Abrir um ficheiro de imagem que se encontra num computador remoto, e apresenta-lo no computador do utilizador.
- Permitir que o utilizador visualize uma imagem em formato TIF dentro do Internet Explorer 5.0 .

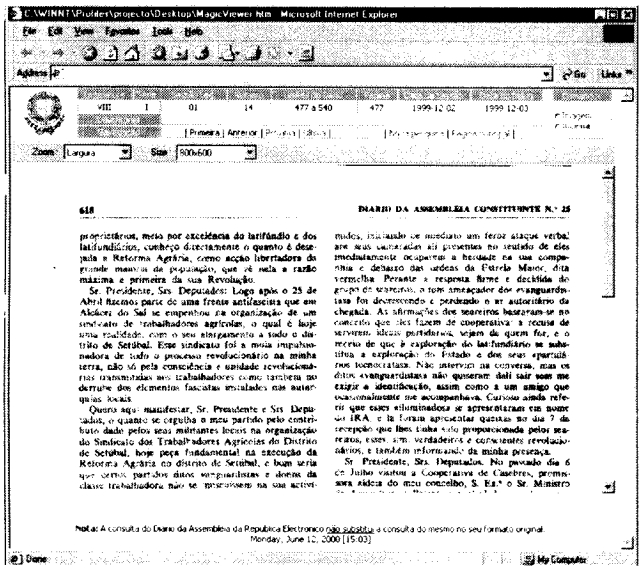


Fig 5. Exemplo da utilização do componente MAGICVIEWER.

É possível ao utilizador alterar o modo como o documento é apresentado. Disponibilizamos duas listas de opções: uma lista para zoom e outra para tamanho.

Na lista de zoom disponibilizamos seis opções diferentes. Quatro destas opções são para visualizar a página com diferentes ampliações. As outras duas servem para ajustar a página em largura ou em altura. Na lista de tamanho permitimos que o utilizador altere o tamanho do objecto de modo a poder visualizar melhor a página de acordo com as definições do browser.

Este módulo de software é um ActiveX Control que foi desenvolvido através de Visual Basic 6.0. Depois de compilado gera-se um *ocx* que é necessário registar no sistema para que seja possível utilizar o objecto que foi criado.

Existem duas maneiras distintas de instalar este software no computador de um utilizador. Pode-se instalar o programa através de um programa de instalação normal ou através da Internet.

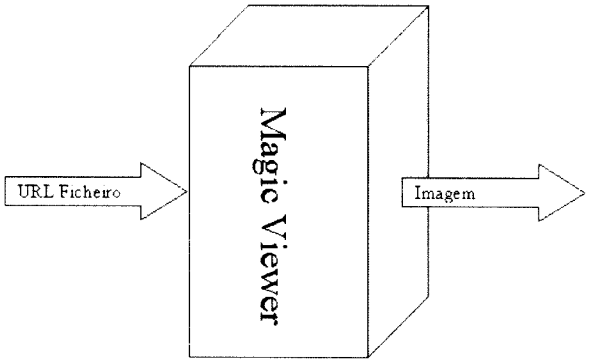


Fig. 6. Modelo do objecto

De seguida vamos descrever o funcionamento deste objecto para melhor compreensão do funcionamento deste sistema. Inicialmente fornecemos um URL ao objecto. A partir deste URL efectuamos um pedido HTTP ao servidor com o nome do ficheiro pretendido e criamos um ficheiro temporário no disco do utilizador para poder visualizar a imagem pretendida. Dentro deste objecto temos ainda incorporado um outro objecto. Este objecto é o objecto da Wang que permite visualizar TIF's e cujo nome é imgedit.ocx . Fornecemos então a este objecto, o nome do ficheiro que pretendemos abrir e apresentamos a imagem.

IV CONCLUSÕES E TRABALHO FUTURO

No geral a construção do site da Diário da Assembleia da república electrónico está concluída. As ferramentas necessárias para o seu bom funcionamento estão desenvolvidas e pensamos que o resultado global é bastante satisfatório, havendo no entanto ainda a necessidade de desenvolver algum trabalho a nível de design e de apresentação da informação.

REFERÊNCIAS

1. Skanson ,Jan , C++ *from the beginning*, Addison Wesley, 1997
2. Kamin, Samuel N., Programming with class: A C++ Introduction to Computer Science, McGraw-Hill,1996
3. Stroustrup, Bjarne, *The C++ Programming Language*, Addison Wesley, 1986
4. Skylyarov, Valery, The Revolutionary guide to Turbo C++, WROX Press, 1992
5. Brophy, Keith , Teach yourself ActiveX Control programming with Visual Basic 5 in 21 days, 1997

HiParaGraphs, uma Linguagem de Especificação de Algoritmos de Controlo Paralelos e Hierárquicos

Andreia Melo, Valery Sklyarov, António Ferrari

Resumo – **Este artigo apresenta um método de especificação utilizado para descrever o comportamento de uma unidade de controlo digital. É uma linguagem formal que resultou da evolução da sua antecessora, os HGSs – Hierarchical Graph-Schemes. Denominada *HiParaGraphs* – Hierachical & Parallel Graphs, esta linguagem pretende acrescentar novas facilidades de paralelismo aos algoritmos de controlo, além de melhorar a especificação anterior com a introdução de novos parâmetros e diferentes tipos de nodos.**

Abstract - **This paper presents a specification method for describing the behavior of digital control units. It is a formal language that results from an evolution of its predecessor, the HGSs – Hierarchical Graph-Schemes. It is called HiParaGraphs – Hierarchical and Parallel Graphs, and the main objective was to add new facilities, mainly related to parallelism.**

I. INTRODUÇÃO

O trabalho desenvolvido na área da especificação e projecto de sistemas complexos e reactivos tem resultado em vários formalismos, tais como os Statecharts [1] e as redes de Petri [2]. Neste artigo descreve-se uma linguagem de especificação cujo objectivo é integrar algumas propriedades dos métodos já existentes e acrescentar novas funcionalidades. Os sistemas reactivos são caracterizados por reagir continuamente a eventos, isto é, a estímulos internos e externos. Estes sistemas são utilizados em inúmeras áreas, tais como em telecomunicações, sistemas operativos, sistemas de aviação, etc. Vamos aqui debruçar-nos sobre a parte de controlo de um dado sistema digital, também de características reactivas, a que chamaremos unidade de controlo.

A dificuldade reside na descrição do comportamento reactivo de forma clara e realista e ao mesmo tempo formal e rigorosa. O comportamento de um sistema reactivo é um conjunto de sequências de entrada permitidas e eventos de saída, condições, acções e eventualmente algumas restrições temporais.

Os Grafos Hierárquicos e Paralelos (HiParaGraphs - *Hierarchical & Parallel Graphs*) são uma linguagem de especificação gráfica formal de algoritmos de controlo que se baseiam no formalismo da máquina de estados finitos (FSM – *Finite State Machine*). Um HiParaGraph descreve algoritmicamente o comportamento de unidades de controlo digitais, sendo composto por diferentes tipos de nodos ligados entre si, por intermédio de arcos direccionados, que estabelecem o fluxo de execução do algoritmo. Desta forma, a sua construção é análoga à de um fluxograma.

A linguagem de especificação que irá ser aqui descrita resultou de uma evolução de linguagens anteriormente propostas, às quais se adicionou alguma funcionalidade. Inicialmente denominada GSs (*Graph-Schemes*) [3], esta linguagem especificava algoritmos de controlo simples que depois evoluíram para os Esquemas de Grafos Hierárquicos (*Hierarchical Graph-Schemes* – HGSs) [4]. Neste passo foi acrescentada a hierarquização dos algoritmos de controlo, composta por um algoritmo principal e vários sub-algoritmos que são executados um de cada vez, ou seja, um algoritmo invoca outro e este quando terminar a sua execução retorna àquele que o invocou. O passo posterior foi a especificação do paralelismo onde se acrescentou aos HGSs a possibilidade de invocar dois ou mais sub-algoritmos simultaneamente de forma a serem executados ao mesmo tempo. Este é o ponto actual da evolução e é onde pretendemos adicionar novas formas de sincronismo no âmbito da execução paralela. Como por exemplo, os sub-algoritmos podem iniciar a sua execução e não retornar à parte invocadora, podem ser invocados indirectamente a partir de dois ou mais sub-algoritmos que estejam a ser executados, ou até a parte invocadora pode ter a possibilidade de terminar um algoritmo que invocou, qualquer que seja o estado deste. Estas são algumas das características da linguagem de especificação gráfica que será descrita ao longo deste artigo, os HiParaGraphs.

Este artigo está dividido em sete secções. A primeira corresponde à presente introdução. Na segunda secção descreve-se brevemente as características gerais da linguagem. A secção três é dedicada à especificação linear dos circuitos de controlo. Na quarta secção é apresentada a funcionalidade hierárquica deste formalismo. O paralelismo dos HiParaGraphs é abordado na secção cinco. A sexta secção descreve as novas funcionalidades da ferramenta de software que suporta este método de especificação. A secção sete contém as conclusões.

II. CARACTERÍSTICAS GERAIS DOS HIPARAGRAPHS

Ao formalismo clássico das FSMs estão associados os diagramas de transição de estados. Estes diagramas não fornecem a noção de hierarquia e de modularidade, não

permitindo, consequentemente, a utilização de técnicas de decomposição descendente ou montagem ascendente durante o desenvolvimento de um sistema. O número de transições num diagrama pode ser bastante elevado. Por exemplo, um evento que causa transições iguais num grande número de estados, como é o caso de uma interrupção de elevada prioridade, é representado por um elevado número de arcos iguais, resultando assim numa quantidade enorme de transições e complicando bastante a representação. Outra desvantagem deste método é o crescimento exponencial do número de estados quando o sistema apenas cresce linearmente.

Os Statecharts [1] resolvem estes problemas fornecendo uma representação modular, hierárquica e concorrente. A comunicação entre FSMs é também uma característica deste método.

As redes de Petri [2] são extremamente eficientes na descrição de sistemas concorrentes e paralelos, utilizando uma representação simples, também à base de estados (*places*) e de transições. Graficamente os *places* são circunferências e as transições são linhas. A interligação entre estes dois componentes é efectuada por intermédio de arcos direccionados. Uma desvantagem da representação visual deste método é a utilização de um número muito reduzido de símbolos que implica que a detecção das várias situações de sequenciação, sincronismo, paralelismo e concorrência numa descriçao complexa seja efectuada não de uma forma directa mas sim após uma análise da rede. A hierarquia e a modularidade também não são explicitamente representáveis.

Os HiParaGraphs resultam do estudo e da integração de métodos formais já existentes. Nasceram das propriedades sequenciais e simples dos diagramas de transição de estados e evoluíram de forma a combinar a hierarquia, modularidade e comunicação dos Statecharts com o paralelismo das redes de Petri e dos Algoritmos de Controlo Lógicos [6]. Utilizando uma notação gráfica e intuitiva, permitem o desenvolvimento top-down do circuito pretendido.

Como elementos gráficos temos nodos, cuja forma depende da função, e arcos direccionados. Os nodos encapsulam operações de teste em conjuntos de entradas e acções nas saídas. Os arcos direccionados interligam os vários tipos de nodos. Com apenas estes recursos visuais estamos aptos a descrever algoritmos de controlo lineares e sequenciais. No entanto, se considerarmos um algoritmo de controlo como um conjunto de sub-algoritmos, necessitamos de nodos com novas funcionalidades de forma a prever situações não só de sequenciação, como era o caso anterior, mas também de sincronismo e paralelismo. Por outro lado, os vários sub-algoritmos podem ser organizados segundo uma hierarquia e assim será necessária a especificação de invocações.

Resumidamente, a actualização desta linguagem baseou-se na adição de novos tipos de nodos (*Sync*, e *Switch*) e na utilização de parâmetros de entrada em sub-algoritmos, relativamente aos HGSs, que enriqueceram este método, fornecendo ao algoritmo novas funcionalidades que se revelaram interessantes durante o projecto das unidades de controlo.

Um HiParaGraph possui quatro categorias de nodos: **iniciação**, **terminação**, **activação** e **teste**.

Os nodos de iniciação são denominados *Begin*. Como são sempre os primeiros nodos de cada sub-algoritmo, poderão mostrar uma lista de parâmetros de entrada, cuja activação determinará o início da execução do respectivo algoritmo. Desta forma acrescentou-se a especificação de sincronismo.

Os nodos de terminação são *End* (obrigatório, para determinar o final de um sub-algoritmo) e *Halt* que pode ser usado para suspender a execução de um sub-algoritmo no caso de ser detectada uma situação de excepção.

Tal como o nome indica, os nodos de activação são utilizados para activar sinais. Os vários tipos de nodos que se enquadram nesta categoria são os Operacionais, que são usados tanto para activar sinais de saída como para invocar sub-algoritmos. A estes nodos acrescentaram-se indicadores de activação quer em sinais de saída quer em invocações de sub-algoritmos. Estes, por sua vez, podem ser especificados utilizando parâmetros de entrada e desta forma podem ser activados a partir de um ou mais sub-algoritmos diferentes durante a sua execução, isto é, a invocação de um dado procedimento poderá não depender unicamente de um mas de um conjunto de estados de sub-algoritmos diferentes. Para este efeito, ou seja, o sincronismo no caso de algoritmos paralelos, são utilizados os nodos *Sync*. Os nodos do tipo *Dummy* são um caso particular dos Operacionais pois como não actuam sobre qualquer sinal, representam um estado de espera. Finalmente, os *Assign* utilizam-se para devolver o resultado (verdadeiro ou falso) das funções.

Os nodos de teste são usados para testar o valor lógico dos sinais de entrada, a que chamamos condições lógicas. Nesta classe temos os nodos do tipo Condicionais com Condição Lógica, que testam as condições lógicas e os Condicionais com Função Lógica que são responsáveis pela invocação de uma função lógica. Os *Switch* encapsulam uma rede de nodos com condições lógicas pois verificam uma dada sequência destes sinais. A funcionalidade de cada tipo de nodo será descrita mais à frente com maior detalhe.

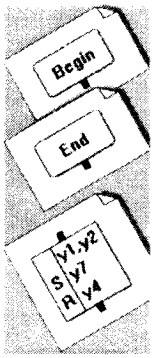
III. ESPECIFICAÇÃO NÃO HIERÁRQUICA

Os circuitos reactivos que estamos aqui a considerar, tal como já foi mencionado na introdução deste artigo, são aqueles que constituem a parte de controlo de um sistema digital que se pode dividir em unidade de controlo e unidade de execução. A primeira é a parte do sistema que pretendemos especificar o comportamento e cuja função é produzir uma sequência de operações na unidade de execução. Estas operações (elementares), são chamadas microoperações. $Y=\{y_1,\ldots,y_n\}$ é o conjunto de microoperações induzidas pelos sinais binários $y_1,\ldots,y_n$ da

unidade de controlo. Para activar a microoperação y_n (n=1,...N) o sinal y_n=1 deverá ser activado. Por vezes algumas microoperações são executadas em simultâneo (no mesmo ciclo de relógio) na unidade de execução. A sequência de activação das microoperações é determinada por funções de transição, isto é, funções Booleanas α_{ij} (i,j=1,...T) de variáveis Booleanas $X=\{x_1,...x_L\}$. Estas variáveis correspondem às variáveis de entrada da unidade de controlo, também denominadas condições lógicas, e podem ser alteradas pelas microoperações.

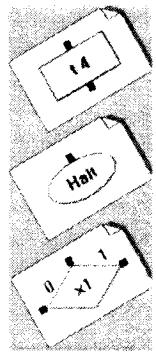
Um HiParaGraph é uma linguagem gráfica que dispõe de vários tipos de nodos para especificar facilmente o comportamento de um algoritmo de controlo. Recorrendo a dois tipos de nodos, teste e activação, um HiParaGraph é capaz de testar o valor binário de um sinal de entrada ou de uma expressão Booleana de variáveis de entrada e activar ou desactivar os valores das saídas. Os nodos possuem pontos de conexão onde se ligam os arcos direccionados permitindo assim estabelecer ligações com outros nodos. No entanto, existe uma restrição relativamente às ligações que consiste numa única ligação de um ponto de saída de um nodo. Os pontos de entrada podem estar ligados a um ou mais nodos. Algumas regras básicas são a obrigatoriedade de um nodo se situar num caminho existente entre o nodo *Begin* e o nodo *End*. Desta forma estamos a garantir que se encontra correctamente ligado e que não pertence a um ciclo infinito. As saídas dos nodos condicionais não se devem ligar à própria entrada pois desta forma estamos a assumir que existe um estado de espera que não está representado explicitamente no algoritmo e para este efeito existe um nodo específico denominado *Dummy*.

Os vários tipos de nodos que constituem um HiParaGraph não hierárquico são os seguintes:



Os nodos *Begin* e *End* representam os pontos de entrada e de saída do algoritmo de controlo, respectivamente, e a sua presença num HiParaGraph é obrigatória e única.

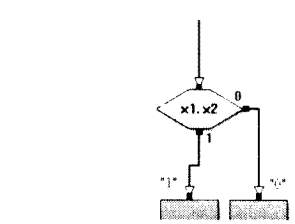
Originalmente [3], [4] os nodos operacionais eram utilizados para especificar a activação dos sinais de saída da unidade de controlo. No seu interior são colocadas listas de microoperações, que correspondem a esses sinais de saída. Além disto foi acrescentada a possibilidade de utilizar indicadores de activação nestes sinais de forma individual. Isto permite-nos activar (indicador SET - "S") ou descativar (indicador RESET - "R") um dado sinal durante um ou mais ciclos de relógio, ou durante um período de tempo correspondente a um ou mais estados do circuito. Os indicadores são colocados à esquerda da lista de microoperações que irá ser afectada. O número de ciclos de relógio durante o qual este nodo permanecerá activo pode também ser aqui especificado. É indicado por um número inteiro precedido pela letra "t" e por defeito é igual a 1.

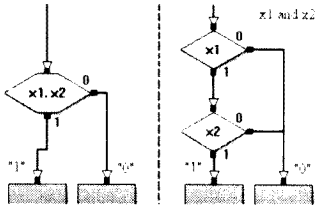


Os nodos *Dummy* são um caso particular dos nodos operacionais pois não contêm microoperações e são utilizados apenas para introduzir um estado de espera. Por defeito o seu tempo de activação é um ciclo de relógio, mas tal como no caso anterior, podem ser especificados múltiplos de ciclos de relógio. No exemplo da figura ao lado temos no topo um nodo *Dummy* que ficará activo durante 4 ciclos de relógio.

Os nodos *Halt* indicam o final da execução do algoritmo no caso de este ter de ser interrompido numa situação de erro ou de excepção. Quando a execução do algoritmo atinge este nodo assume-se que o próximo estado da execução corresponde a ele próprio. Difere do nodo *End* pois, no caso de uma hierarquia, o *End* significa retorno enquanto o *Halt* significa paragem. Os nodos condicionais possuem uma funcionalidade configurável pois podem conter quer uma condição lógica pertencente ao conjunto $X=\{x_1,...,x_L\}$, que corresponde a uma entrada do circuito de controlo quer um subconjunto de condições lógicas às quais é aplicada uma função Booleana (por exemplo, AND, OR). Este nodo possui um ponto de conexão de entrada e dois pontos de saída que correspondem aos valores lógicos 0 e 1 do sinal binário da entrada ou do resultado da função Booleana de variáveis de entrada nele especificadas. Os nodos condicionais que possuem conjunções e disjunções de variáveis de entrada (Fig. 1) são úteis quer para encapsular por si só vários nodos condicionais num só, quer para facilitar o teste do valor lógico de uma condição mais extensa, composta por conjunções e disjunções (somas de produtos, produtos de somas, etc).

A especificação de funções Booleanas nos nodos condicionais permite diminuir o número de nodos por algoritmo e consequentemente facilita a tarefa de verificação e de síntese. Também por este motivo foi criado mais um tipo de nodo de teste a que chamamos *Switch*. O seu comportamento compara-se ao de um descodificador com um número de entradas variável. O número de variáveis de entrada que podem ser colocadas neste nodo pode ser 1, 2 ou 3 e por isso teremos 2, 4 ou 8 saídas, respectivamente.





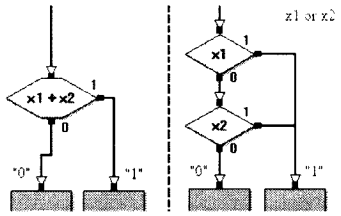


Fig. 1 – Encapsulamento de funções Booleanas AND e OR em nodos condicionais com duas variáveis.

A utilização destes nodos reduz significativamente o número de nodos condicionais numa construção equivalente. A introdução de uma variável de entrada implica a replicação de nodos condicionais já existentes, complicando assim a compreensão do algoritmo. Um nodo *Switch* com 3 entradas é equivalente a uma rede de 7 nodos condicionais, tal como mostra a Fig. 2. Como se pode concluir, uma construção deste tipo iria tornar o algoritmo de controlo mais complexo e sujeito a erros. A verificação das regras de especificação é simplificada pelo uso deste tipo de nodos visto que encapsulam uma função conhecida, que não precisa de ser construída manualmente, e que à partida cumpre todas as regras dos HiParaGraphs.

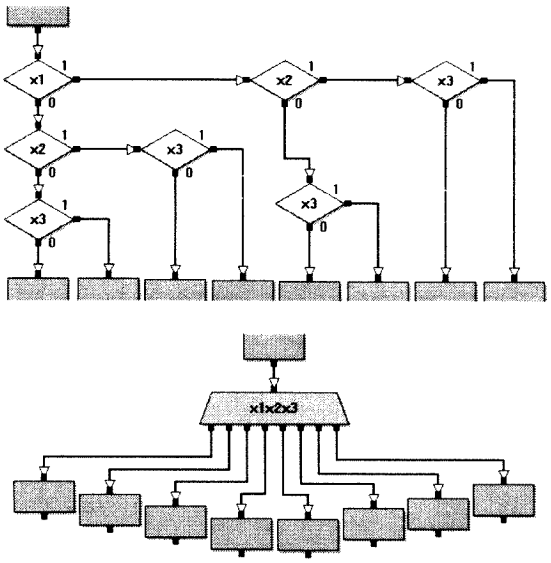


Fig.2 – Nodo Switch – construção equivalente com nodos condicionais.

O exemplo da Fig. 2 pretende mostrar a simplificação obtida pela utilização de um *Switch* de três variáveis. À saída deste nodo temos, da esquerda para a direita, todas as combinações binárias possíveis, de 0 a 7, respectivamente. A rede de nodos condicionais representada acima torna-se assim evitável eliminando-se uma possível fonte de erros e facilitando a tarefa de verificação automática.

Quer os nodos condicionais quer os nodos *Switch* podem ser ligados de forma a construírem funções mais complexas que deste modo são interpretadas de forma modular na própria especificação do algoritmo.

Os algoritmos de controlo especificados através de um HiParaGraph são sequenciais e podem ser modelados por uma máquina de estados finitos. Por exemplo, se considerarmos uma máquina de Moore, podemos associar os nodos de iniciação, activação e terminação aos vários estados do circuito, sendo as transições de estado efectuadas nos flancos ascendentes ou descendentes do sinal de relógio. O cálculo do estado seguinte depende dos nodos de teste existentes no algoritmo. A sequenciação, está representada na Fig. 3, onde é comparada com a especificação equivalente de uma rede de Petri. Estes algoritmos são equivalentes se considerarmos que o *place* (circunferência do topo da Fig.3b)) da rede de Petri que possui o *token* corresponde ao nodo operacional que activa a microoperação y_1. Os traços horizontais da rede de Petri representam transições, dependentes de condições, assim como os nodos de teste de um HiParaGraph. Na Fig. 3b), se a condição da primeira transição for verdadeira o *token* passará para o *place* seguinte, dando-se assim a transição de estado, caso contrário o *token* permanecerá no mesmo *place*. No caso do HiParaGraph, Fig.3a), se a condição lógica x_1 for verdadeira, do nodo operacional que activa y_1 passa-se para o nodo operacional que activa y_2. Por outro lado, se a condição for falsa, o estado activo do circuito continuará a ser o nodo operacional responsável pela activação de y_1.

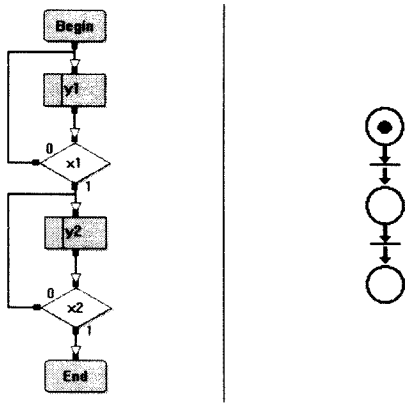


Fig. 3 – Especificação de sequenciação a) num HiParaGraph e b) numa rede de Petri.

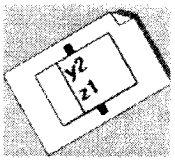
IV. ESPECIFICAÇÃO HIERÁRQUICA

A especificação hierárquica de um algoritmo de controlo justifica-se quando o comportamento de uma unidade de controlo é descrito por um algoritmo extenso, de tal forma que se poderia dividir em módulos, facilitando tanto a compreensão da especificação como a implementação do respectivo circuito. A descrição formal das máquinas de estados finitos hierárquicas (HFSM – *Hierarchical Finite State Machine*) [5] diz que apenas um nível hierárquico está activo em cada instante. Um dado algoritmo que é implementado como uma FSM invoca outro que será executado num nível hierárquico inferior.

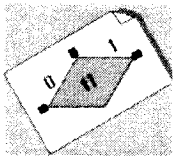
Quando este terminar a sua execução retorna ao algoritmo que o invocou, ou seja, ao nível imediatamente superior.

Os HiParaGraphs, herdando algumas características dos HGSs [4], disponibilizam uma descrição modular de algoritmos de controlo. Quando especifica uma hierarquia um HiParaGraph é composto por um conjunto de sub-algoritmos. Os vários módulos podem possuir características diferentes e por isso foram divididos em dois tipos: **funções** e **procedimentos**. Esta divisão baseou-se apenas na utilização ou não de parâmetros de saída, respectivamente. Tal como é usual em algumas linguagens de programação, como por exemplo o Pascal, as funções devolvem valores ao algoritmo onde são invocadas e os procedimentos apenas executam um conjunto de tarefas sem devolver ou calcular resultados.

Assim, um HiParaGraph, descrevendo uma unidade de controlo hierárquica, necessita de adicionar novos tipos de nodos ao conjunto apresentado na secção anterior.

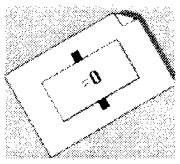


Para representar as invocações a procedimentos (macrooperações) são utilizados nodos operacionais, tais como foram descritos anteriormente, aos quais se adiciona uma macrooperação do conjunto $Z=\{z_1, z_2, \ldots, z_N\}$. Estes nodos podem simultaneamente especificar a activação de sinais de saída.



A utilização de funções deve-se à necessidade de poder influenciar o fluxo de execução do algoritmo invocador a partir de um valor que é determinado e devolvido pela função, tal como uma condição lógica, decidindo assim qual o próximo estado activo do circuito. Aos nodos de teste que se utilizam para invocar funções chamam-se nodos condicionais com funções lógicas e contêm um elemento do conjunto $F=\{f_1, f_2, \ldots, f_J\}$.

Para devolver o resultado (verdadeiro ou falso) de uma função são usados dois nodos de activação do tipo Assign



que devem pertencer à especificação da função, um com o valor 0 e outro com o valor 1. Ambos devem ser ligados ao nodo End, pois o retorno do resultado deverá ser a última operação efectuada por este sub-algoritmo.

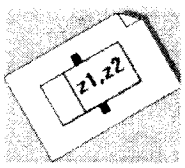
Recursividade

Os vários sub-algoritmos podem invocar outros e até mesmo eles próprios. Desta forma um HiParaGraph poderá especificar **algoritmos de controlo recursivos**. Não existe uma regra que evite a especificação de algoritmos infinitamente recursivos mas, no entanto, esta situação pode ser detectada mediante a construção de um HiParaGraph equivalente, não hierárquico, ao qual se extraem todos os nodos de activação (excepto aqueles utilizados para invocar sub-algoritmos). Isto resulta numa árvore de invocações onde facilmente se detectam ciclos, que podem corresponder a situações de recursividade infinita. Se existir sempre um caminho a partir de um dado nodo para o *End*, significa que a hierarquia especificada não é infinitamente recursiva. Os possíveis ciclos de invocações presentes nesta construção podem eventualmente ser interrompidos por nodos condicionais, ou seja, a execução cíclica pode depender de alguma variável de entrada. Neste caso a execução correcta do algoritmo depende do modelo de implementação escolhido e dos recursos de hardware disponíveis. Por exemplo, um modelo de implementação de uma HFSM que utilize um *stack* está limitado, no que diz respeito à recursividade, devido à sua profundidade. Se o número de invocações recursivas ou até mesmo o número de níveis hierárquicos da máquina for maior do que a profundidade disponível do *stack*, então a dada altura da execução do circuito de controlo chegaremos a uma situação de *stack overflow*. O conhecimento prévio da profundidade do *stack* e a construção do HiParaGraph equivalente, representando a árvore de invocações, poderá evitar esta situação.

V. ESPECIFICAÇÃO PARALELA HIERÁRQUICA

Para especificar o comportamento de sistemas reactivos, como é o caso das unidades de controlo que estamos a considerar, pode-se utilizar vários tipos de linguagens com diferentes filosofias e objectivos, mas preferencialmente orientadas ao estado. Este é o caso dos HiParaGraphs cujo objectivo é descrever o comportamento de um sistema que suporte hierarquia e paralelismo. Combinando algumas propriedades dos Statecharts [1], redes de Petri [2] e Algoritmos de Controlo Lógicos [6], obtivemos uma forma de integrar numa linguagem gráfica as características disponibilizadas por cada uma destas linguagens.

Da definição formal de HFSM apenas um nível hierárquico está activo em cada instante e tal como foi dito na secção anterior, quando pretendemos invocar um



procedimento colocamos uma macrooperação dentro de um nodo operacional.

No entanto, se em vez de apenas uma macrooperação forem especificadas duas ou mais dentro do mesmo nodo, isto significa que no instante em que este nodo for atingido elas serão invocadas simultaneamente. Assim, os procedimentos deverão ser executados em paralelo. Esta situação está representada na Fig. 4, onde também é comparada com uma construção equivalente numa rede de Petri. No exemplo da Fig. 4b), se a condição atribuída à transição for verdadeira, o *token* transitará do seu *place* para ambos os *places* que existem após a transição, ficando assim os dois activos simultaneamente. A Fig. 4a) possui a representação equivalente de um HiParaGraph, onde o nodo operacional especifica a invocação de duas macrooperações simultaneamente. Após este nodo serão activados os nodos *Begin* de cada um dos procedimentos. Neste caso o algoritmo invocador

prossegue a sua execução, mas poderia não o fazer mediante a utilização de um indicador.

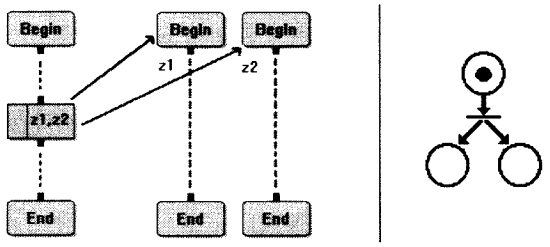
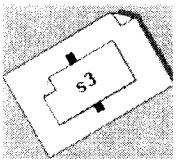


Fig. 4 – Especificação de paralelismo a) num HiParaGraph e b) numa rede de Petri.

Tal como as microoperações podem ter indicadores de activação associados, as macrooperações podem ser especificadas de forma a que o algoritmo que as invoca espere ou não pela sua terminação para poder prosseguir a execução. Assim, uma macrooperação afectada pelo indicador W (*wait*), utilizado nos HiParaGraphs, significa que o algoritmo invocador deve esperar pelo seu retorno. O indicador W é atribuído individualmente a cada macrooperação, o que significa que quando colocado num nodo operacional, pode afectar mais do que uma macro, ou um subconjunto daquelas que lá existem. Neste caso o algoritmo deve esperar pelas macrooperações com W mas caso as restantes ainda não tiverem terminado a sua execução, ele prosseguirá.

Não só os nodos operacionais que possuem macrooperações são os responsáveis pela invocação de procedimentos. Os nodos do tipo *Sync* são utilizados para invocações indirectas que especificam situações de sincronismo, tal como nas redes de Petri (ver Fig. 5).

Nestes nodos são colocados sinais pertencentes ao conjunto $S=\{s_1, s_2, \ldots, s_M\}$ e que são utilizados na comunicação entre sub-algoritmos.



Os nodos *Sync* foram criados para que seja possível invocar um procedimento ou função quando se atingem determinados estados de sub-algoritmos diferentes. Para tal é também necessário especificar parâmetros de entrada nos sub-algoritmos invocados desta forma. Tal como mostra a Fig. 5a), os nodos *Sync*, pertencentes a procedimentos ou funções diferentes, activam os sinais s_2 e s_4, eventualmente em instantes de tempo diferentes. O novo procedimento só inicia a sua execução quando os sinais que são especificados nos seus parâmetros de entrada estiverem ambos activos, ou seja, espera pela activação dos dois nodos *Sync* para poder iniciar a sua execução. Os algoritmos que possuem os nodos do tipo *Sync* prosseguem a sua execução normalmente, pois estão apenas a actualizar o valor de um sinal de saída.

Por outro lado, na Fig. 5b), a rede de Petri possui uma transição que, embora a sua condição seja verdadeira, só pode disparada quando os *places* que a precedem possuírem *tokens*, tal como está representado na figura. Assim, esta transição equivale àquela que existe a seguir ao nodo *Begin* da Fig. 5a). Neste caso, tal como mostra a figura, pressupõe-se a existência de dois sub-algoritmos diferentes pois não é possível num HiParaGraph existir dois estados activos no mesmo sub-algoritmo.

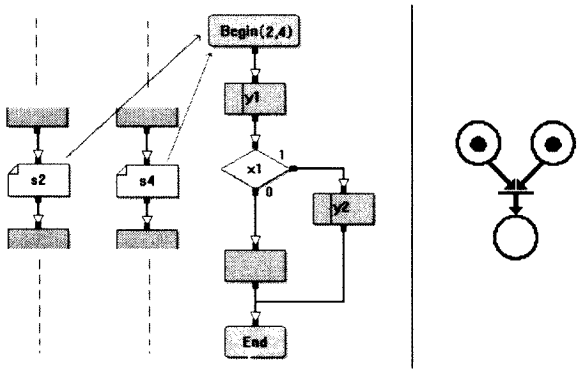


Fig. 5 – Especificação de sincronismo a) num HiParaGraph e b) numa rede de Petri.

VI. SOFTWARE – EDIÇÃO GRÁFICA

A aplicação de software que suporta esta linguagem de especificação chama-se GraphBuilder. A sua versão anterior [7], [8] foi melhorada de forma a integrar as novas características da especificação relativas ao paralelismo.

A janela principal desta aplicação encontra-se na Fig. 6, mostrando um HiParaGraph já construído. Além da especificação esta ferramenta integra também outras facilidades, nomeadamente a verificação, marcação de estados (Mealy ou Moore), optimização, conversão para código VHDL comportamental sintetizável e depuração. No entanto, estas tarefas não serão aqui explicadas.

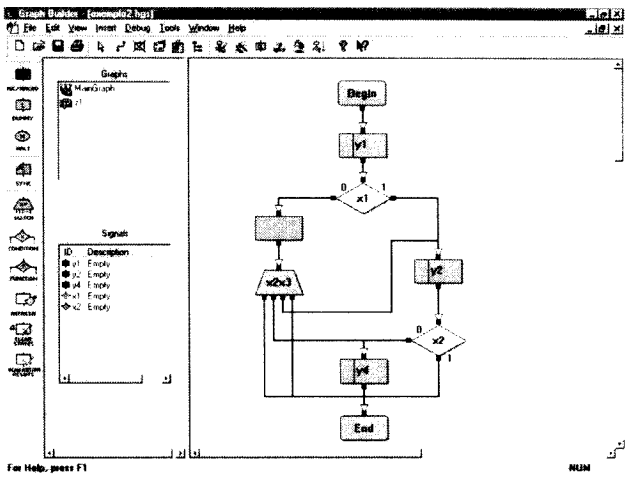


Fig. 6 – Janela principal da ferramenta GraphBuilder.

Nas secções anteriores foram apresentados os diferentes tipos de nodos que constituem um HiParaGraph. Todos eles pertencem a uma hierarquia de classes implementáveis em Visual C++ e cuja estrutura está representada na Fig. 7. Esta estrutura hierárquica foi modificada relativamente às versões anteriores, pois com o aumento do número de nodos foi necessária uma divisão de classes de acordo com a funcionalidade de cada um.

Como estamos na presença de uma especificação hierárquica, a primeira classe representa a hierarquia e chama-se *CHierGraph*. Uma hierarquia é composta por sub-algoritmos e por isso esta classe inclui a *CGraphScheme* que por sua vez possui vários nodos (no mínimo 2, *Begin* e *End*). Assim, esta inclui a classe *CNode* e a partir daqui são derivados os quatro tipos de nodos de que dispomos: iniciação – *CInitiationNode*, terminação – *CTerminationNode* , teste – *CTestNode* e activação – *CActivationNode*. Como temos apenas um tipo de nodo para iniciar os vários sub-algoritmos, deriva-se a classe *CBeginNode* (nodo *Begin*) da *CInitiationNode*. No caso dos nodos de terminação temos as classes *CEndNode* (nodo *End*) e *CHaltNode* (nodo *Halt*) derivadas da *CTerminationNode*. Para teste deriva-se da classe *CTestNode* a *CLogCondNode* (nodos condicionais com condições lógicas), *CLogFuncNode* (nodos condicionais com funções lógicas) e *CSwitchNode* (nodo *Switch*). Finalmente, para activação, deriva-se da classe *CActivationNode* a classe *COpNode* (nodos operacionais), *CDummyNode* (nodos *Dummy*), *CAssignNode* (nodos *Assign*) e *CSyncNode* (nodos *Sync*).

Todas estas classes estão armazenadas numa DLL (*Dynamic Link Library*) que é carregada quando a aplicação é iniciada.

Os sub-algoritmos de um HiParaGraph são entidades autónomas e podem ser reutilizados em diferentes algoritmos e consequentemente em diferentes unidades de controlo. Assim, a ferramenta GraphBuilder permite gravar não só um HiParaGraph completo (objecto da classe *CHierGraph*), mas também os seus componentes, isto é, permite a gravação individual de cada sub-algoritmo (objectos do tipo *CGraphScheme*).

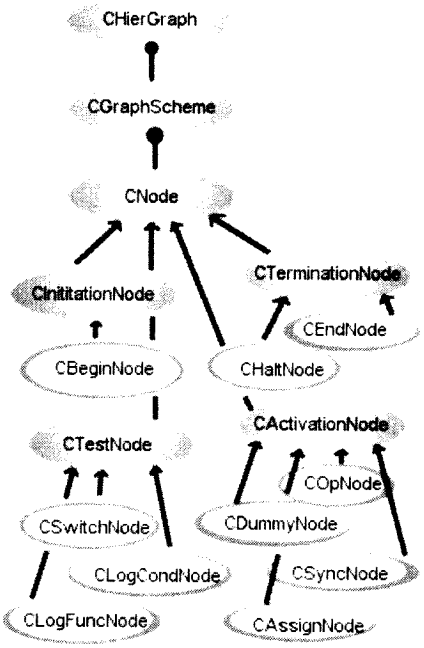


Fig. 7 – Hierarquia de classes para implementação de um HiParaGraph.

VII. CONCLUSÕES

A especificação de algoritmos de controlo através de HiParaGraphs suporta explicitamente hierarquia (o que não acontece com as redes de Petri e os Algoritmos de Controlo Lógicos) e paralelismo. Embora os Statecharts possuam também estas características, não existe uma forma automática de sintetizar e implementar os circuitos especificados por este método. Isto não acontece com os HiParaGraphs que além de tentarem combinar as características e vantagens dos vários métodos formais mencionados, são sintetizáveis e implementáveis. Embora não tenha sido descrito neste artigo, a ferramenta de software que suporta este método, integra uma aplicação que é capaz de converter os HiParaGraphs em código VHDL comportamental sintetizável. Essa aplicação, denominada SynGraph, utiliza um *template* em código VHDL que pode ser usado para sintetizar máquinas de estados finitos, ao qual efectua as alterações necessárias para descrever o comportamento do circuito desejado. O conjunto de ferramentas de software desenvolvidas para suportar este método de especificação permite modificar e expandir facilmente os algoritmos de controlo construídos em qualquer fase do projecto do circuito [9], [10].

Com este método pretende-se automatizar o projecto de unidades de controlo partindo de uma especificação facilmente extensível e sintetizável. Os HiParaGraphs são uma descrição gráfica formal independente do modelo de implementação do circuito.

REFERÊNCIAS

[1] David Harel, "Statecharts: A Visual Formalism for Complex Systems", Science of Computer Programming, Nº8, pp. 231-271, 1987.

[2] Tadao Murata, "Petri Nets: Properties, Analysis and Applications", Proceedings of the IEEE, Vol. 77, Nº4, pp.541-590, Abril 1989.

[3] Samary Baranov, "Synthesis of Microprogrammed Automata", Energy Publishing Company, 1974 (em russo).

[4] Sklyarov, V., "Hierarchical Graph-Schemes", Latvian Academy of Science, Automatics and Computers, Riga, 1984, N 2, pp 82-87.

[5] Arnaldo Oliveira, "Modelos, Métodos e Ferramentas para Implementação de Unidades de Controlo Virtuais", Dissertação de Mestrado em Engenharia Electrónica e de Telecomunicações, Universidade de Aveiro, Julho 2000.

[6] A. D. Zakrevskij, "Parallel Algorithms for Logical Control", Institute of Engineering Cybernetics of NAS of Belarus, Minsk, 1999.

[7] A. Oliveira, A. Melo, V. Sklyarov, "Specification, Implementation and Testing of HFSMs in Dinamically Reconfigurable FPGAs", actas de FPL'99, 9th International Workshop on Field Programmable Logic and Applications, Glasgow, Reino Unido, pp. 313-322, Agosto 1999.

[8] Andreia Melo, "Especificação, Optimização e Teste de Algoritmos de Controlo Hierárquicos", Dissertação de Mestrado em Engenharia Electrónica e de Telecomunicações, Universidade de Aveiro, Janeiro 2000.

[9] Sklyarov, V., Monteiro, R. S., Lau, N., Melo, A., Oliveira, A., Kondratjuk, K., "Integrated Development for Logic Synthesis Based on Dynamically Reconfigurable FPGAs", *Field-Programmable Logic and Applications*, 8th International Workshop, FPL '98, Tallinn, Estonia, Agosto 1998.

[10] Sklyarov, V., Lau, N., Oliveira, A., Melo, A., Kondratjuk, K., Ferrari, A. B., Monteiro, R. S., Sklyarova, I., "Synthesis tools and design environment for dynamically reconfigurable FPGAs", *SBCCI98 XI Brazilian Symposium on Integrated Circuit Design*, Búzios, Rio de Janeiro, Brazil, Setembro 1998.

A Coprocessor for Traffic Scheduling and Schedulability Analysis in FTT-CAN

Ernesto Martins, José Alberto Fonseca

Resumo - **A complexidade e a flexibilidade do sistema de escalonamento on-line de mensagens usado num barramento CAN, é normalmente limitada pelo baixo desempenho dos microcontroladores usados. Uma solução possível para ultrapassar esta limitação consiste em transferir as tarefas de escalonamento para hardware dedicado.**

Este artigo descreve um coprocessador para escalonamento de tráfego e análise de escalonabilidade. O escalonamento pode ser programado segundo três diferentes critérios, e o número de mensagens a escalonar bem como os parâmetros respectivos podem ser alterados dinamicamente. O coprocessador é ainda capaz de analisar a escalonabilidade de um grupo de mensagens, suportando assim mecanismos de admissão controlada. Embora tenha sido desenhado de forma a suportar o protocolo FTT-CAN, pode ser adaptado também a outros barramentos de campo com escalonamento centralizado. Este artigo não apresenta ainda qualquer estimativa do desempenho do coprocessador.

Abstract - **The low-processing power microcontrollers typically used within CAN nodes, usually place tight limits on the complexity and flexibility of on-line message scheduling systems. One solution to break this barrier is to transfer the scheduling task to a hardware implementation.**

A preliminary version of a traffic scheduling and schedulability analyser coprocessor is proposed in this paper. This coprocessor, which will be implemented on a low-cost FPGA, can generate message schedules for the node CPU, leaving it just with the dispatching task. Adding to this, scheduling can be made to follow one of three different policies, and the number of messages to be scheduled and their respective parameters can be changed dynamically. To support on-line admission control of new messages, the coprocessor implements a schedulability analyser function. The coprocessor was designed to support the FTT-CAN protocol, but it can be adapted to any other fieldbus using centralized scheduling.

Since the coprocessor is still in an early design stage, no measures of performance are yet available.

I. INTRODUCTION

The FTT-CAN (Flexible Time-Triggered communication on CAN) protocol presented in [1] and developed further in [2], adopted originally a centralized planning technique to schedule the synchronous traffic on the CAN network.

The planning scheduler [3] was proposed as a means to trade-off between the purely static and dynamic scheduling policies. Specifically it tries to achieve a part of the timeliness characteristics of static scheduling, and some of the flexibility of dynamic scheduling, without having its time overhead. Basically what the planning scheduler does is to build static schedules for consecutive fixed duration periods of time called the plans. The creation of a plan table is overlapped with the dispatching of the previous plan. The bus time is usually divided in fixed duration time slots called the Elementary Cycles (ECs). The EC duration is the basic time unit in which message periods are expressed. Plans are always made up of an integer multiple of ECs.

The plan size affects two important properties of the scheduling system: its run-time overhead and its response time to changes in the message set. Moreover these are affected in opposite directions, as the former decreases with the plan size while the latter increases [2].

The run-time overhead is usually a sensitive issue due to the limited processing power generally available in CAN network nodes, in particular in the FTT-CAN master node where the scheduler executes. In addition, the latter node has to execute on-line admission control procedures which, usually, require relevant processing power. This limitation sets a lower bound on the plan size, beyond which scheduling might not be feasible. Now, the problem arises when this lower bound is still too high to guarantee a satisfactory response time to changes in the message parameters.

In an attempt to decouple the basic scheduler requirements from the capabilities of the node CPU, a specialized coprocessor dedicated to this task was developed. The Planning Scheduler Coprocessor [4], or PSCoP as it was named, was implemented in an FPGA and supports up to 8 messages. The two internal plan tables available in this first version can be configured with up to 16 ECs. The coprocessor is accessed by the CPU as a peripheral, and is capable of scheduling a plan in less than 1% of the time needed to dispatch it [4].

In this paper we propose a new scheduling coprocessor inspired on the same basic PSCoP architecture, but with a whole new functionality, including a schedulability analyser.

The next section starts with a presentation of the coprocessor features, and explains the reasons which led us to include some of them. Section III, which is the main part of this article, gives an overview of the coprocessor

architecture, first by presenting the interface to the microcontroller, and then by discussing the internal architecture with all its functional blocks and execution mechanisms. Finally, section IV describes the present status of this development, and concludes the paper.

II. DEFINING THE COPROCESSOR FEATURES

The Planning Scheduler Coprocessor represented our first effort towards a specialized hardware solution for message scheduling in a fieldbus system. An important result obtained from this experience concerns the speed gap between the coprocessor and the microcontroller, which was found to be much higher than what we thought at first. This suggests that we can introduce additional functionality in the coprocessor (constrained, off course, by the FPGA size) and still manage to have a performance level compatible with the time constraints imposed by the system.

Also based on the high relative performance of PSCoP, it was decided that the new design should work as a dynamic scheduler coprocessor instead of working as a planning scheduler. The notion of Elementary Cycle (EC) was kept, and, instead of building a complete plan table, the coprocessor was thought to build just one EC-schedule at a time.

Having a single EC-schedule output, guarantees that all changes in the message set are accounted for in the next EC-schedule, thus ensuring the highest responsiveness. However, preventing the coprocessor to build several EC-schedules in advance as PSCoP is able to, doesn't allow it to anticipate the missing of deadlines as in the planning scheduler. To solve this problem we introduced a schedulability analysis capability in the coprocessor. Therefore, every time the message set is changed, we can run the analysis to check if the set is still schedulable. Scheduling is resumed only if the analysis indicated a schedulable set.

A response time-based schedulability analysis function can be implemented in the coprocessor using the timeline method [2]. Since this method consists basically in building EC-schedules - exactly what a scheduler coprocessor is supposed to do - its design can be made almost without any additional complexity.

In PSCoP, scheduling is ruled by a static priority-based policy. Pursuing a goal of maximum flexibility, the new coprocessor was defined supporting a few other scheduling policies between which operation can be changed dynamically. Also, besides the period, phase and message duration, there is now explicit support for the deadline and priority parameters. All parameters can be changed during operation.

The following list summarizes the coprocessor features:

- Dynamic scheduler & schedulability analyzer;
- Three scheduling policies: rate monotonic (RM), deadline monotonic (DM), and Priority-based;
- Support for 32 variables with 8-bit parameter resolution;
- Variables parameters are: phase (Ph), period (P), deadline (D), priority (Pr) and message duration (C);
- Scheduling policy, variable set and parameters can all be changed dynamically;
- One, 32-bit, EC-schedule output register.

III. COPROCESSOR ARCHITECTURE

Starting from the feature set described in the previous section, an architecture was defined. In this section we describe such an architecture. Section *A* presents the CPU interface and discusses the coprocessor programmer's model as seen by the node microcontroller. Section *B* describes the architecture at the register-transfer level.

A. Node CPU Interface

A.1. Variable's Parameters Register Slots

These registers (see figure 1) hold the parameters of each variable to be scheduled. There are 32 slots of registers for a maximum of 32 variables. Each slot contains 5, 8-bit registers, where the parameters of each variable should be written to prior to any scheduling or schedulability analysis operation.

The variable's priority, optionally written to the priority register (PriReg), has a range of 0 (highest priority) to 255. The values written to the phase (PhaReg), period (PerReg) and deadline (DeaReg) registers are expressed in number of ECs. A register slot has no variable allocated to it if its period register is cleared. The values written to the message duration register (DurReg) should be expressed in a normalized unit given by NECd = (EC Duration / 255). If a message has a duration of Δt time units, its

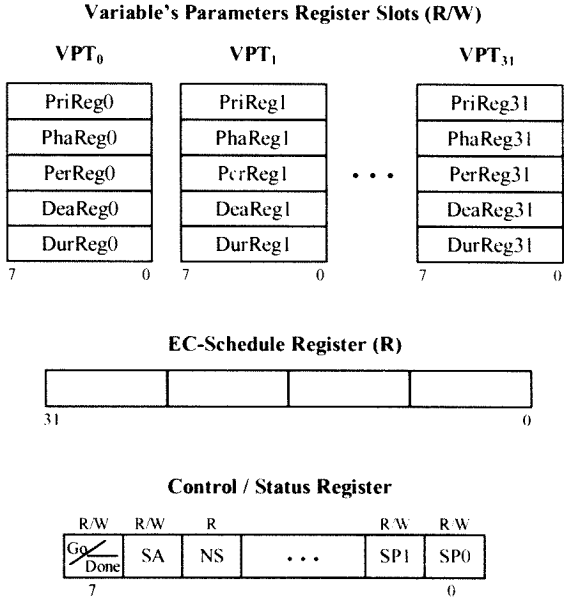


Figure 1 - Coprocessor programmer's model.

corresponding DurReg register should have the value $\lceil \Delta t / NECd \rceil$. The deadline value written in the DeaReg register is only relevant for schedulability analysis.

A.2. EC-Schedule Register

At the end of a scheduler operation this 32-bit register contains the schedule for an entire EC. Each variable allocated in that EC is specified by a 1, in the bit position corresponding to its register slot. Bit number i corresponds to the variable assigned to register slot i. This is the same coding method used in PSCoP.

A.3. Control and Status Register

This 8-bit register is used by the microcontroller to control the coprocessor. Only 5 bits are used.

Go/Done - The microcontroller sets this bit to command the coprocessor to generate an EC-schedule (if SA=0) or perform a schedulability analysis (if SA=1). The coprocessor resets this bit when done.

SA - When set, enables the schedulability analysis mode in the coprocessor.

NS - After the coprocessor is instructed to do a schedulability analysis, NS becoming 1 means that the set is not schedulable.

SP[0,1] - These two bits specify the scheduling policy to be used by the coprocessor, according to the assignments in Table 1.

SP1	SP0	Scheduling Policy
0	0	Priority-based
0	1	Rate Monotonic
1	0	Deadline Monotonic
1	1	not used

Table 1 - Scheduling policy control bits.

B. Architecture Overview

At a high level of description the architecture of the new coprocessor doesn't differ much from the one adopted in PSCoP [5]. This is true because, despite dealing now with a dynamic scheduler, we can still divide the scheduling function in the same two separate activities identified in the planning scheduler algorithm. These are, the action of placement of transactions in the ECs, and the function of keeping track of the instants in time when each variable must be produced. In the new coprocessor this work is carried out by the EC-Schedule Builder (ECSB) and the Variable's Production Timer (VPT), respectively.

Figure 2 depicts the coprocessor architecture with one ECSB and 32 VPTs connected through an internal bus. Each variable to be scheduled is allocated to one VPT unit which holds the variable's period (P), initial phase (Ph), deadline (D) and priority (Pr) parameters. VPTs are thus

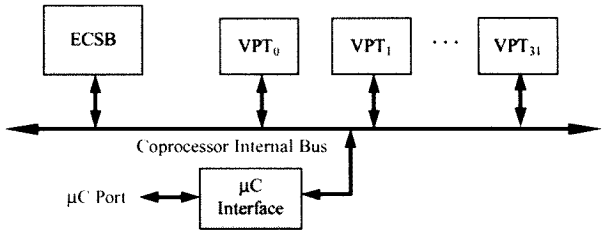


Figure 2 - Coprocessor architecture.
ECSB- EC-Schedule Builder. VPT - Variable's Production Timer.

the physical image of the variable's parameters register slots of the coprocessor programmer's model.

Global timing information received from the ECSB allows all VPTs to be synchronised while keeping track of the EC-schedule currently being generated. When a VPT detects that the scheduling for a particular EC where its variable should be produced has started, it signals the ECSB requesting the allocation of the associated transaction. Based on the transactions' duration (C) and the remaining EC time left, the ECSB unit decides to allocate or reject the transaction. If the transaction is accepted, further requests for allocation in the same EC (from other VPTs) are evaluated, otherwise the current EC-schedule is finished and the Go/Done bit in the Control/Status register is cleared.

B.1. The Variable's Production Timer

The internal structure of each VPT unit is represented in figure 3. The registers PriReg, PerReg and DeaReg of the coprocessor programmer's model (figure 1) are physically implemented in a register file inside the VPT.

Like in PSCoP, there are two presettable decrement counters, the Allocation Counter (AC) and the Deadline Counter (DC). AC is used to keep track of the variable's release time. DC is used in the schedulability analysis to evaluate if the variable can be produced within the specified deadline. Unlike PSCoP, nothing is done here to detect missed deadlines during normal scheduler operation - the user is expected to use the schedulability analysis capability in order to guarantee this never happens.

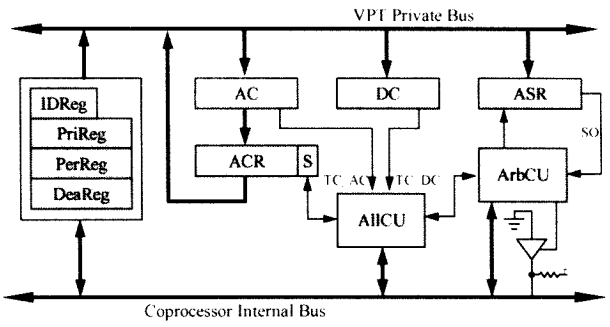


Figure 3 - Variable's Production Timer internal structure.
AC - Allocation Counter; DC - Deadline Counter; ACR - Allocation Counter Register; S - State Flag Register; ArbSReg - ASR - Arbitration Shift Register; AllCU - Allocation Control Unit; ArbCU - Arbitration Control Unit.

Also, there is no dedicated register for the phase parameter. What is mapped in the PhaReg position of the register slot is the AC counter, which is thus initially set with the phase value as it should be.

The AC counter is decremented by one every time the ECSB starts a new EC-schedule. When AC reaches zero, indicating the variable should be produced, the Allocation Control Unit (AllCU) requests the variable allocation to ECSB.

Because more than one VPT can request allocation in the same EC, a mechanism must exist to decide which request to serve first. In the new coprocessor this selection mechanism is based on a serial self-selection arbiter distributed by all the VPTs. Contention in this kind of arbiter is resolved using dynamic priority vectors which can be derived from message parameters such as their period or priority, making it easy to implement well-known scheduling policies, and to switch dynamically between them as desired. The serial self-selection arbiter is implemented by the Arbitration Shift Register (ArbSReg) and the Arbitration Control Unit (ArbCU) in each VPT, and a common tristate bus contention line.

The VPT and the ECSB have two operating modes: the Scheduler Mode (or S-Mode) and the Analyser Mode (or A-Mode), which correspond, respectively, to the coprocessor functions of scheduler and schedulability analyser.

In the middle of its normal scheduler operation, the coprocessor can be asked to do a schedulability analysis, and then to resume scheduling from the point where it was interrupted. To do this the VPTs must save their state in S-Mode before switching to A-Mode. The state information to save, which will be needed on return from A-Mode, is the current value in AC and a bit indicating whether or not the VPT has a pending request. The resources which hold these two pieces of information during a schedulability analysis are, respectively, the Allocation Counter Register (ACR) and the State Flag Register (S) in figure 3.

B.2. VPT Operating Modes

Figure 4 depicts the operation flowcharts of the VPT in its two modes.

<u>S-Mode</u> - In the S-Mode (Fig.4-a), after all registers have been initialized, the VPTs sit idle in state S_{1S}, waiting for ECSB to start building the first EC-schedule. When the Go/Done bit is set in the Control/Status register, ECSB initiates a series of cycles in which it receives the IDs of VPTs requesting the allocation of their variables in the current EC. With a collection of these IDs, ECSB builds the EC-schedule.

The VPT uses its AC counter as shown, with the phase and period parameters, to keep track of the EC where its variable should be allocated. When the processing for that particular EC-schedule begins, the VPT raises a request. Next an arbitration cycle decides which among the VPTs requesting allocation, can send its ID to ECSB. For a

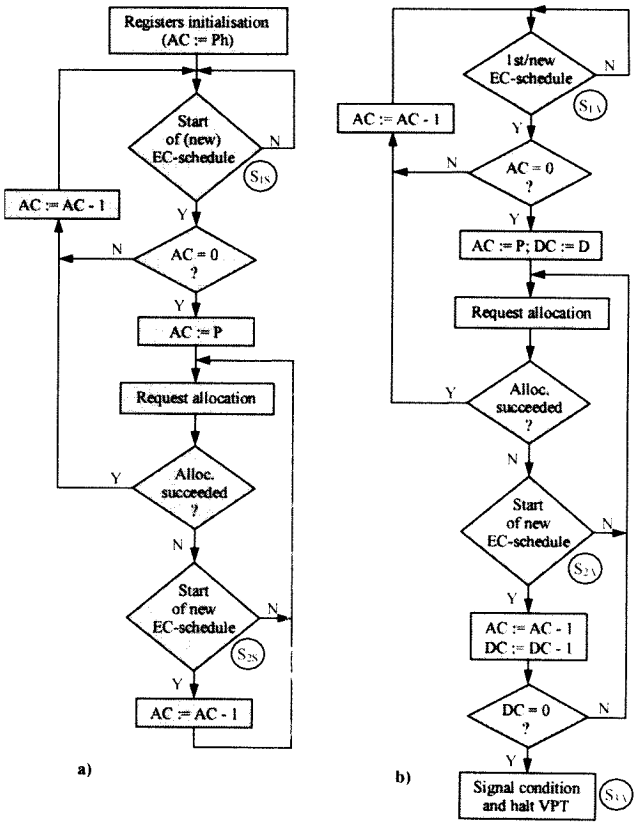


Figure 4 - VPT operation. a)- Flowchart in EC-Scheduler mode (S-Mode); b)- Flowchart in Schedulability Analysis mode (A-Mode).

variable to be successfully allocated, its VPT must not only win the arbitration, but also it must receive a positive acknowledge from ECSB indicating that the variable's transaction fits in the EC (or its remaining time left). The sending of this acknowledge as well as the transmission of the ID, are both part of the arbitration protocol.

Many arbitration cycles may be generated as part of each EC-schedule processing. If a VPT was not able to allocate its variable in the first arbitration cycle, it waits for the next while keeping its request active. Eventually the current EC may be closed before the VPT is able to allocate successfully. In this case the VPT continues asserting its request while waiting for the next EC, where the variable allocation is tried once more.

Between the processing of EC-schedules, VPTs are idle in either states S_{1S} or S_{2S}. In this condition the variable set may be changed by the CPU (the coprocessor idle state is indicated by a zero in bit Go/Done in the Control/Status register). All parameters of existing variables can be changed. In particular, changes in the variable's period are guaranteed to be made without any spurious transient. Variables are deleted by clearing the period register which deactivates the corresponding VPTs. Adding new variables activates other VPTs which, after being initialized, sit idle in state S_{1S}.

<u>A-Mode</u> - As can be seen from figure 4-b, the VPT flowchart in A-Mode is very similar to the S-Mode

flowchart. The difference is basically the utilisation of the Deadline Counter (DC).

In testing the schedulability of the message set, what the coprocessor does is to check that the response time experienced by each message is always below its respective deadline value. Theory shows that it is enough to do this response time analysis at time zero only [2]. In other words, if we guarantee that the first allocation of each variable is made within its deadline, then we can assure the schedulability of the set.

To check individual response times we build what is known as the timeline. This consists basically in constructing the EC-schedules in the normal way, until all variables are allocated at least once. If we succeed in doing this without any missed deadline, then the set is guaranteed to be schedulable.

To implement this method, the coprocessor needs little more than the scheduling functionality. In particular the VPT needs just to use the DC counter to monitor the delay between the variable's release and its allocation. If this delay exceeds the variable's deadline, ECSB is signaled.

Transitions between S-Mode and A-Mode - Figure 5 details how the VPTs switch between the two operating modes. Bit SA in the Control/Status register controls these transitions. If SA is zero, the coprocessor is in S-Mode. If SA is set the coprocessor changes to A-Mode, and all VPTs sit idle in S_{1A}. If now the Go/Done bit is set, a schedulability analysis is performed. After the analysis is completed, some VPTs will be in S_{1A}, some in S_{2A}, and, if the set turned out to be non-schedulable, one or more VPTs will end in state S_{3A}. If SA is now cleared, the state of all VPTs prior to the change to A-Mode will be restored, and the coprocessor returns to S-Mode.

If the set turned out to be non-schedulable we might want to change some parameters in the message set and run another schedulability analysis. The changes in the message set (e.g. changes in the parameters, deletion of messages) can also be made with the coprocessor in A-Mode.

To do another schedulability analysis all VPTs must, however, be first initialised in state S_{1A} with AC=0. One way to do this is to toggle the coprocessor to the S-Mode and then back to A-Mode using the SA control bit. After

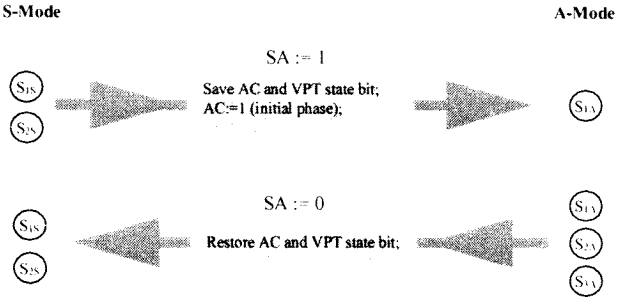


Figure 5 - Transitions between S-Mode and A-Mode in the VPT operation.

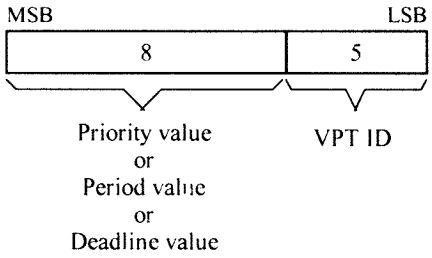


Figure 6 - Arbitration vector.

this operation all VPTs will be in state S_{1A} ready to start another schedulability analysis.

B.3. Arbitration between VPTs

The self-selection arbitration system serializes the arrival of allocation requests to ECSB, based on the criteria programmed in bits SP[1,0] of the Control/Status Register.

The arbiter relies on a contention bus line with a recessive and a dominant state, through which VPTs with pending requests try to send their respective IDs to the central ECSB. Each VPT uses a different priority vector to contend on this shared bus line. A bit-wise arbitration of all vectors from contending VPTs is serially performed during the arbitration cycle, resulting in the highest vector being transmitted.

Priority vectors have 13 bits as shown in figure 6. The most significant 8 bits are taken from the priority, period or deadline register, as dictated by the scheduling policy in use. The least significant 5 bits are equal to the VPT identifier, which is hardwired in the IDReg register. Using the VPT ID as part of the priority vectors guarantees the uniqueness of all vectors while eliminating the need for a separate transaction to identify the winning VPT.

B.4. The EC-Schedule Builder

This module builds EC-schedules in scheduler mode, and controls the coprocessor operation in analyser mode. Figure 7 shows the ECSB architecture.

As we saw above, the ECSB builds EC-schedules by accepting requests for allocation from the VPTs. The start of processing of a new EC-schedule (in scheduler or analyser mode) is communicated to all VPTs by a global signal, which the VPTs use to decrement their internal counters, and know therefore when they should request the release of their respective variables.

If there are requests, ECSB triggers a first arbitration cycle, at the end of which it acquires the ID of the highest priority VPT. The ID is captured in the register AddLatch, and used as an index to the table MDLut. This is a RAM look-up table where the DurReg registers, holding the duration of each transaction, are physically mapped. The transaction duration of the corresponding variable is thus retrieved from this table to check if the transaction fits in the EC, or its remaining free time (if we are already past the first allocation). To check this, ECSB keeps track of

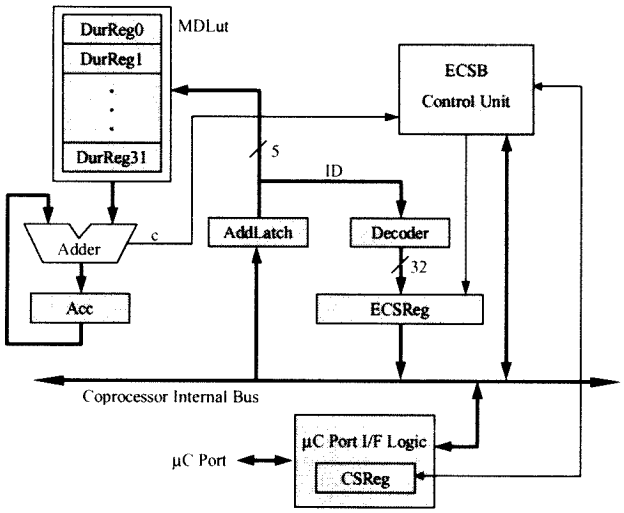


Figure 7 - EC-Schedule Builder architecture and µC Interface. MDLut - Message Duration Look-up table; ECSReg - EC-Schedule Register; CSReg - Control/Status Register.

the total EC time already booked by previous transactions in the Acc Register. The new transaction time is thus accumulated to the value in this register. Due to the normalization of transaction times, if the time just added exceeds the time left in the EC, this addition results in an overflow condition.

In case there is no overflow, the variable is allocated in the EC. This allocation is recorded in the EC-Schedule Register (ECSReg) by the Decoder module using the ID in the AddLatch, which sets the bit located in the bit position corresponding to the VPT just served. A signal sent at the end of the arbitration cycle, tells the VPT that the allocation was made, allowing it to withdraw its allocation request. In case there are more pending requests from other VPTs, the ECSB initiates another arbitration cycle.

This sequence is repeated until ECSB finds an allocation request whose transaction exceeds the remaining time in the EC. In this case the variable is not allocated, and the EC-schedule is closed. No more arbitration cycles are generated, which means that pending requests are deferred until the next EC-schedule is processed. In scheduler mode the Go/Done bit is reset and the coprocessor goes idle.

In analyser mode the ECSB works basically the same way, with the exception that it does not stop after completing an EC-schedule. In this mode the ECSB simulates the construction of successive EC-schedules. It stops doing this only if a VPT detects the missing of a deadline, or if all VPTs succeed in allocating their variables at least once. The former condition is communicated to ECSB by the assertion of a bus line. It indicates that the set is not schedulable and results in the activation of the NS flag in the Control/Status Register.

The latter condition is detected by ECSB, using the EC-Schedule Register. In analyser mode this register is not cleared between the processing of successive ECs, so that the allocation of all variables is detected when all bits become set. In this case the set is flagged as schedulable by a zero in NS.

Note that, contrary to the VPT modules, the ECSB does not need to save any state information before switching to analyser mode.

IV. PROJECT STATUS AND CONCLUSIONS

In this paper we described a coprocessor capable of traffic scheduling and schedulability analysis. It was conceived as an evolution of PSCoP, and as such it is particularly adapted to the FTT-CAN protocol.

At the time of writing the coprocessor is still in the early stages of development, with only the VPT unit almost past the logic design stage. So, for now, no quantitative performance figures are available yet to assess the coprocessor's feasibility. As soon as all design entry is complete we expect to obtain an estimate of its performance through simulation. Afterwards, a first prototype with all the characteristics described, will be implemented on a XC4010XL FPGA. Hopefully it will include 32 VPTs, depending on the FPGA resources required by each.

For the coprocessor to achieve its goal, it must be able to run the schedulability analysis fast enough, typically in a fraction of the time it takes to dispatch an EC-schedule. In addition, in case the message set is not schedulable, the microcontroller must have time to change whatever it is needed to make it schedulable. Building the timeline implies generating EC-schedules until all variables are allocated at least once. If no deadline is violated in this process, then the set is said to be schedulable. The time to complete the timeline will be dictated by the highest of all deadlines, by the number of variables allocated in each EC and, of course, by the speed of the design itself.

While a worst case for the execution time of this analysis is now difficult to predict, the high performance room obtained for PSCoP, indicates that such a schedulability analysis function has all chances to work within the time constraints imposed by most applications used with fieldbuses.

REFERENCES

[1] Almeida, L., Fonseca, J., Fonseca, P.; "A Flexible Time-Triggered Communication System Based on the Controller Area Network" Proc. FeT '99 - Fieldbus Systems and their Applications Conf., Germany, Sept. 1999.

[2] Almeida, L.; "Flexibility and Timeliness in Fieldbus-Based Real-Time Systems", PhD Thesis, University of Aveiro, Portugal, November 1999.

[3] Almeida, L., Pasadas, R., Fonseca, J.; "Using The Planning Scheduler to Improve Flexibility in Real-Time Fieldbus Networks" IFAC, Control Engineering Practice Vol. 7, Nº 1, pp. 101-108, Janeiro de 1999.

[4] Martins E., Neves P., Fonseca J.; "PSCoP - A Planning Scheduler Coprocessor", WIP Proceedings of the IEEE International Workshop on Factory Communication Systems, ISEP Porto, September 6-8 2000, pp.27-30.

[5] Martins E., Neves P., Fonseca J.; "Architecture of a Fieldbus Message Scheduler Coprocessor based on the Planning Paradigm", submitted for publication in Microprocessors and Microsystems.

An Introductory Course on Human Computer Interaction

Beatriz Sousa Santos, Joaquim Sousa Pinto

Resumo - **Apresenta-se resumidamente o programa, métodos de ensino e bibliografia da disciplina de Interfaces Humano Computador que tem sido oferecida desde 1993/94 como opção da Licenciatura em Electrónica e Telecomunicações da Universidade de Aveiro.**

Abstract – **An introductory course on Human Computer Interaction offered as an elective to the Engineering degree on Electronics and Telecommunications at the University of Aveiro is presented. Its contents, teaching methods and bibliography are briefly described.**

I. INTRODUCTION

When computers first appeared on the commercial scene in the 1950s they were cumbersome and very difficult to use; since then dramatic decreases in the cost of computing resources have resulted from new technological advances; in less than thirty years computers changed from being huge machines (housed in large air-conditioned rooms) to much smaller machines, including ones that can be carried around. The development of the first personal computers in the 1970s was a landmark because these machines provided interactive computer power for individual users at low cost. These changes have opened up a wide range of new possibilities for the way in which computers can be used. Computing is becoming ubiquitous.

Ordinary people began using computer systems, and for computers to be widely accepted and used effectively they need to be designed for the needs and capacities of the people for whom they are intended. However, computer professionals, namely software engineers, have been extensively trained in ways of thinking that are very different from those of most possible users of computer systems and tend to judge their work by criteria that may have little to do with the needs and constraints of the end user. For example, efficiency of code and flexibility of architecture are relevant engineering goals but are not enough to guarantee the success of a computer system in being accessible to, and supportive of, a particular type of user; thus a carefully designed user interface is a determinant factor.

In the 1980's a marketing brochure (for the Aple's Macintosh) focused, for the first time, decidedly on the human interface. This was an important evolution since publicity and attention was brought to computer-human interface as a selling point; it signalled a new attitude on the part of the computer industry toward this aspect of software products. The motivation for this change in attitude, and the then new and still growing interest in computer-interaction, came mainly from the fact that much more resources could be dedicated to the human computer interface without sacrificing functionality, performance or cost effectiveness. In fact, the user of computer technology is now often a person who is technologically uneducated, unsophisticated and unmotivated, rather than a professional technologist. Acceptable and appealing interfaces, as well as desired functionality, must be offered. Certain kinds of users simply will not buy or use systems which require too much of an investment in learning. To serve this kind of user, products must be very easy to learn.

At an individual level, user interfaces change many people's lives: doctors can make more accurate diagnoses, children can learn more effectively, graphic artists can explore more creative possibilities and pilots can fly airplanes more safely. Some changes however are disruptive; too often, users must cope with frustration, fear and failure when they encounter excessive complexity, incomprehensible terminology, or chaotic layouts.

Effective systems generate positive feelings of success, competence, mastery and clarity in the user community. The users are not encumbered by the computer and can predict what will happen in response to each of their actions. When an interaction system is well designed, the interface almost disappears, enabling users to concentrate on their work, exploration or pleasure.

There is a endless desire for increased productivity among computer users and organisations. Small improvements in productivity on a task can add up to significant organisational benefits when the number of users and volume of transactions are high, thus human performance in the use of computer systems will remain a rapidly expanding research and development topic in the coming decades. The interdisciplinary field of Human Computer Interaction combines the data gathering methods and intellectual framework of experimental psychology with the powerful and widely used tools developed from computer science and engineering. This is the environment where any professional who develops S/W applications has to move. Thus, nowadays, a basic education in Human Computer Interaction is very important.

In 1991, the document produced by the ACM/IEEE-CS Joint Task Force, "Computing as a Science" [1], considered Human Computer Interaction, along with Computer Graphics, as constituting Human Computer Communication, one of the nine sub-areas of Computer

Science. In 1994, the National Science Foundation in the United States of America subsidised a report on "New Directions in Human-Computer Interaction, Research and Practice" which identifies Human Computer Interaction as a fundamental field of S/W Engineering and reports difficulties in integrating it in the curricula of American Universities. Note that in Portuguese Universities, by that time, there were already courses on this subject, the one described in this work and another at Instituto Superior Técnico in Lisbon [2].

Currently, the draft of the Computing Curricula 2001 [3] includes some hours of Human Computer Interaction on the body of knowledge considered essential for all undergraduate programs in Computer Science and Engineering.

In the following sections a brief description of the general objectives, contents, bibliography and teaching methods of a course on Human Computer Interaction lectured at the University of Aveiro will be presented.

II. GENERAL CONTEXT AND OBJECTIVES OF THE COURSE

The course described in this work has been an elective offered to fifth year students of the Engineering degree on Electronics and Telecommunications, since 1993/94. During some years (until 1996/97), a few MSc students were also allowed to attend it.

Currently (since 1999/2000), the fifth year is organised in several areas and this course belongs to the core of the Information Systems area, along with courses on Data Bases and Distributed Architectures. As every student of this area attends simultaneously all core courses, it is possible to coordinate the activities performed at those courses, namely to have common practical assignments.

The specific objective of this course on Human Computer Interaction is to introduce the students to what is this multi-disciplinary field, addressing the main issues of the design, implementation and evaluation of human computer interfaces. Principles and guidelines are presented but also theoretical and experimental foundation from which they are obtained.

This course should also serve more general objectives, such as training students to be able to use adequate working methods as well as correct technical writing and presentation.

III. GENERAL CONTENTS

This course was designed to have 26 hours of lectures. Ten topics are addressed:

1- Introduction to the problem of designing the User's Interface: definition, general methodology and principles of design;
2- User's Profile and modelling: Human Information Processing System (HIPS): psychological and physical characteristics of the user; mental models.
3- Conceptual models: principles and guidelines to the development of conceptual models;
4- Dialogue styles: characteristics, experimental results applicability and principles and guidelines for the design of the main dialogue styles and their combined use;
5- User interface evaluation: significance, evaluation methods, their characteristics and applicability;
6- Input/Output devices: main existing devices and ergonomic issues;
7- Screen layout and design and colour usage: experimental results, principles and guidelines;
8- User's documentation: experimental results, principles and guidelines for the design of manuals and help on-line;
9- Software for User Interfaces: Windows Management Systems (WMS) and User Interface Management Systems (UIMS) and technologies for the Web;
10- Advanced applications: Computer-Supported Cooperative Work.

In the first lecture, corresponding to topic n.1, a general introduction to the area is made. Besides presenting the definitions and goals of Human Computer Interaction, the great importance of the user's interface is stressed: a significant part of the effort to produce an interactive application is devoted to the user's interface to which the success and acceptation of that application is strongly related. A general description of the main principles and guidelines that should be followed in the design of the user's interface is presented.

Topic n.2 is concerned with the "human part" of the interaction process. The importance of knowing the profile of the intended users is stressed and the characteristics to be considered in its definition are introduced. An overview of the most relevant aspects of the Human Information Processing System is presented stressing that they have important implications for design and that users, in spite of sharing common capabilities and limitations, are individuals with differences which should not be ignored. Some models of the user are introduced and mental models are presented.

Topic n.3 introduces conceptual models: experimental results are described and principles and guidelines to follow in the design of conceptual models for user interfaces are introduced; metaphors are also addressed.

Topic n.4 introduces a classification of dialogue styles (seven styles: direct manipulation, menus, question and answer, fill-in-forms, function keys, command languages and natural languages) as a taxonomy which helps understanding the characteristics and applicability of each style; this classification is useful in spite of the fact that, in general, several styles are used simultaneously and the frontiers between them are often blurred.

Topic n.5 is concerned with evaluation; it is introduced as an absolute necessity nowadays. A range of methods is presented along with their applicability: expert reviews, usability testing, surveys and continuing assessment.

In topic n.6 an introduction to the most common input/output devices is made stressing the increased concern on human factors that has steered the development of new

versions of old devices. Issues related to keyboards, pointing devices, devices for speech recognition and generation as well as displays are briefly described. Some novel devices are also referred.

Topic n.7 deals with screen layout and design as well as the problem of using colour effectively; some principles and guidelines are presented and the power and dangers of using colour are addressed (related to the psychophysical characteristics of the Human Visual System and cultural issues).

Topic n.8 addresses the design of supplementary materials, in both paper and online form, always needed to help users. Several types of user's written or online documentation are introduced, as well as the main principles and guidelines that should be followed in their design.

Topic n.9 introduces the general issues related to Windows Management Systems and Graphical User Interface Management Systems.

Finally in topic n.10, the design and evaluation of new technologies to support the social processes of work, often among distant partners, are introduced.

This is the general content of the *curriculum* adopted in this course as it has been applied in 1999/2000 and 2000/2001. It has evolved, according to the experience obtained each time it was taught. This evolution is expected to continue; several subjects already considered important are still missing in this *curriculum* and others are becoming relevant as information visualization or virtual reality the new paradigm for interaction. However this *curriculum* is already heavy as it is and the introduction of new topics should be carefully analysed.

On the second semester of the current academic year (2000/2001) another course on Human Computer Interaction will be lectured to 3rd year students of Computers and Telematics Engineering. These students have different background and interests; this will imply some changes in the way this course is lectured. The authors believe the main changes will be needed in the teaching methods, practical assignments and evaluation methods. Since this new course will have 39h of lectures, some more topics will be addressed, however the general content and bibliography seem to support equally well these new students.

IV. BIBLIOGRAPHY

There are several books that cover the contents of this course and can be used as textbooks. However, until now several books have been used to support the course, recommending to the students one or two for each subject, according to the approach they use. Other references are used to support specific topics (web sites or papers), which are not included in this bibliography. In the next section the general usage of this bibliography is described and some comments are provided.

A. General usage of bibliography

The first bibliographic reference (of the list presented in the next section), the text by Shneiderman is the one having a general organization which is believed to better support the current overall structure of the course; it is a comprehensive and up-to-dated text, however most topics are also addressed, with the right approach, in the texts by Preece et al. and Dix et al.. The former addresses in much more detail the issues related to user interface evaluation. The reference by Mayhew is not as comprehensive, however it gives a much more thorough view of the principles and guidelines that should be observed in the user's interface design; this characteristic makes it most useful and it has been used as the main reference to support several topics concerned with dialogue design.

The book by Nielson has been used to support practical assignments on evaluation.

The book by Tufte is generally very interesting and has been used specifically for the topic of colour usage.

The textbook by Hearn and Baker has been used to support the study of input/output devices.

B. Commented bibliography

Shneidermen, B., *Designing the User Interface, Strategies for Effective Human-Computer Interaction*, 3rd ed., Addison Wesley, 1998

Provides a complete and current introduction to user interface design, namely the principles and practices needed to design an effective interaction. It includes human factors for interactive software, tested methods to develop and assess interfaces, interaction styles and design considerations such as effective messages, consistent screen design and appropriate colour.

Dix, A., J. Finlay, G. Abowd, B. Russell, *Human Computer Interaction*, Prentice Hall, 2nd ed., 1998

It is a textbook, providing a multidisciplinary perspective of the subject. It covers the basic psychology and computer technology involved in the interface between humans and computers, as well as usability and advanced topics. This edition includes new material on virtual reality and ubiquitous computing, design for the Internet and information visualization among other topics.

Mayhew, D., *Principle and Guidelines in Software User Interface Design*, Addison Wesley, 1992

This book differs from the previous two texts; it is a practical text that provides available principles and guidelines for the design of user interfaces, plus research data and theory from which they are drawn. Based on extensive review of research and practice in human-computer interaction, it discusses software user interface in detail.

Preece, J., I. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey, *Human Computer Interaction*, Addison Wesley, 1994
Offers a comprehensive account of this multidisciplinary field illustrating the powerful benefits of a user-oriented approach to the design of modern computer systems. It balances technical and cognitive issues required for understanding the interplay between people and computer, particularly in new fields as multimedia, virtual environments and computer supported cooperative work.

Nielsen, J., *Usability Engineering*, Academic Press, 1993
It is a guide to the methods of usability engineering and helps non technical people improve the systems so that they are not only free of errors but also easier, more pleasant to use and more efficient.

Tufte, E., *Envisioning Information*, Graphics Press, 1990
This book reveals the design strategies for enhancing the dimensionality and density of portrayals of information. General principles of information design, which are universal, help to identify and explain design excellence – why some displays are better than others. It includes several hundred displays of complex information and a section on colour usage.

Hearn, D., M. P. Baker, *Computer Graphics*, 2nd ed., Prentice Hall, 1994
This can be considered a textbook on Computer Graphics; it includes a chapter on input/output devices and colour models using a different approach from the one generally used in Human Computer Interaction texts.

V. TEACHING METHODS

Elective courses of the degree in Electronics and Telecommunications Engineering have 2 hours of lectures and 2 hours of practical classes per week. This results, for a typical semester, in ≈26 hours of lectures and 26 hours of practical classes.

Every year (until 1996/97) a few MSc students, who wished to get some background on the area, were allowed to attend this course. These students were submitted to a different evaluation; they had more demanding implementation assignments, as the ones described in [4-10]. Evaluation, for all the other students, has always been based on an examination and practical assignments; however practical assignments have been of different nature along the years. In the first editions, team projects were developed involving several students (≈8). This type of assignment was interesting, however implied that some one would play the part of project leader and integrator of the different parts developed by the students [11,12]. MSc students, under the supervision of the teacher, have carried out this job. Later on, these assignments were replaced by two practical assignments to be executed by groups of two students: one involving the evaluation of user interfaces of S/W applications, web sites or even non-computational systems (as a video recorder or a video camera) and a second assignment involving the design and implementation of a user interface.

Since 1999/2000, the final assignment involving design and implementation of a user interface has been common to the courses on Data Bases and Distributed Architectures. This organization allows the students to have an integrated view of the design and implementation of a more sophisticated application, including management of the information as well as the server and client parts, "behind" the user interface.

The first practical classes have been used to introduce HTML and Visual Basic and to explore and modify some simple dialog boxes. This year, as several students needed to use XML for their graduation projects, this language was also introduced. These initial classes prepare students to implement their own final assignments. Another practical class is devoted to the evaluation of web sites. The remainder practical classes are used to discuss and develop the final assignments.

Students have to submit a report corresponding to each practical assignment; they also have to make short presentations. Since most of the students usually feel very uncomfortable with these tasks, some time is spent explaining how to write a technical report and make oral presentations [13].

VI. WHAT FUTURE?

The contents, bibliography and teaching methods of an introductory course on Human Computer Interaction at the University of Aveiro, was presented. This course was designed to be an elective course for the fifth year of a degree on Electronics and Telecommunications Engineering and has been offered since 1993/94. In the academic year of 2000/2001 it will become a mandatory course of a degree on Computers and Telematics Engineering and will be lectured at the 3rd year. In order to adapt the presented course to the new type of students some modifications will be needed. We believe that this modifications will occur mainly on the teaching methods, evaluation and practical assignments since the general *curriculum* and bibliography seem *a priori* to support equally well the needs of both types of students; however only teaching experience will validate this assumption.

REFERENCES

[1] Tucker, A., B. Barnes, R. Aiken, K. Barker, K. Bruce, J. T. Cain, S. Conry, G. Engel, R. Epstein, D. Lidtke, M. Mulder, J. Rogers, S. Spafford, A. Turner, *Computing Curricula 1991- Report of the ACM/IEEE-CS Joint Curriculum Task Force*, 1991

[2] Gomes, M.R., J. Brisson Lopes, M. J. Fonseca, "A Disciplina de Interfaces Homem Máquina no IST", *1º Workshop Computação Gráfica Multimédia e Ensino. CGME'99*, Leiria, 1999, pp. 7-11

[3] *Computing Curricula 2001 - Draft of the Report of the ACM/IEEE-CS Joint Curriculum Task Force*, http://www.computer.org/education/cs2001/report

[4] Andreia Barbosa de Melo, Beatriz Sousa Santos, C. Ferreira, J. Sousa Pinto, "Software Application for Data Visualization and Interaction in a Location Routing Problem", *Revista do Dep. de Electrónica e Telecomunicações da Universidade de Aveiro*,Vol. 2, nº4, Janeiro 1999, pp. 471-476

[5] P. Marrafa, B. Sousa Santos, "S/W de Monitorização e Gestão da Produção de uma Tecelagem de Etiquetas", *Actas do 8º Encontro Português de Computação Gráfica*, Coimbra, 1998, pp. 133-147

[6] H. Zagalo, J. Sousa Pinto, Beatriz Sousa Santos, J. A. Martins, "Ambiente de Navegação Virtual para Suporte de Educação à Distância", *Actas da 1ª Conferência das Telecomunicações*, Aveiro, 1997, pp. 53-56

[7] C. Ferreira, Beatriz Sousa Santos, M. Eugénia Captivo, J. Clímaco, C. C. Silva, "Multiobjective Location of Unwelcome or Central Facilities Involving Environmental Aspects - A prototype of a Decision Support System", *JORBEL, Belgian Journal of Operations Research, Statistics and Computer Science*, December, 1996, Vol.36, No.1-2, pp.159-172

[8] J. Fonseca, C. Ferreira, Beatriz Sousa Santos, M. Eugénia Captivo, J. Clímaco, J. A. Rafael, "Localização de Equipamentos Hospitalares: um Modelo e Protótipo de um Sistema de Apoio à Decisão", *Actas do 4º Congresso Português de Engenharia Biomédica, BioEng'96*, Aveiro, Abril de 1996, pp. I.3.1-I.3.6

[9] Silvia de Francesco, C. Loff Barreto, Beatriz Sousa, Santosa, J. A. Rafael, "User Interface for a Dentist Office Management Tool", *Revista do Dep. de Electrónica e Telecomunicações da Universidade de Aveiro*,Vol. 1, nº5, Janeiro 1996, pp. 473-477

[10] G. Paiva Dias, J. A. Rafael, Beatriz Sousa Santos, "Interface de Utilisador para Sistemas de Gestão de Imagem Médica", *Revista do Dep. de Electrónica e Telecomunicações da Universidade de Aveiro*, Vol. 1, nº3, Janeiro 1995, pp. 193-198

[11] S. Pinto, Beatriz Sousa Santos, C. Ferreira, J. A. Rafael, "Interface de Utilizador para um Sistema de Apoio à Decisão em Problemas de Cobertura", *Revista do Dep. de Electrónica e Telecomunicações da Universidade de Aveiro*, Vol. 1, nº4, Setembro 1995, pp. 363-366

[12] C. Carvalhal, Beatriz Sousa Santos, C. Ferreira, J. A. Rafael, "User Interface for a Decision Support System Based on Factor Analysis", *Revista do Dep. de Electrónica e Telecomunicações da Universidade de Aveiro*, Vol. 1, nº3, Janeiro, 1995, pp. 257-261

[13] D. Beer (ed.), *Writing and Speaking in the Technology Professions: A Practical Guide*, IEEE Press, 1991

ACKNOWLEDGEMENTS

The authors wish to thank Prof. José Alberto Rafael for his collaboration in this course during the academic years of 1994/95 and 1995/96.

Interacção com um Robô através de Linguagem Natural Falada *

Nuno M. Ferreira, Nuno F. Ferreira, Mário Rodrigues*
António Teixeira, Francisco Vaz, Luís Seabra Lopes

*Faculdade de Ciências da Universidade do Porto

Resumo - **O desenvolvimento de um sistema capaz de interagir com um ser humano através da voz é algo que tem sido alvo de muitos estudos e projectos. Este tipo de sistemas são aliciantes pois permitem ao Homem, através de uma forma simples e intuitiva, comunicar com a máquina. A interface de voz com um robô – o Carl -, em desenvolvimento no âmbito do projecto CARL, é apresentada neste artigo. Apresenta-se sobretudo as tecnologias existentes, as ferramentas disponíveis e a interligação de todos os subsistemas constituintes de uma interface de voz: síntese, reconhecimento e processamento de linguagem natural.**

Abstract – **Developing a system capable of using spoken language for interaction with humans has been pursued by several studies. Such systems are interesting by allowing humans to communicate with machines in a simple and intuitive way. The spoken language interface with a robot (named Carl), in development as part of the CARL project, is presented in this paper. We present, mainly, existing technologies, available tools and connection of all interface subsystems: speech synthesis, speech recognition and natural language processing.**

I. INTRODUÇÃO

A comunicação oral entre o Homem e a máquina é um assunto que tem interessado e fascinado engenheiros e cientistas ao longo de várias décadas. Para muitos, a possibilidade de conversar livremente com uma máquina representa o último desafio para perceber como se processa a comunicação por voz entre as pessoas. Além do desafio que este assunto proporciona, máquinas com interface de voz começam a tornar-se uma necessidade. Num futuro próximo, redes interactivas permitirão um acesso fácil a uma grande quantidade de informação e serviços, o que afectará a maneira como as pessoas trabalham e planeiam o seu dia-a-dia. Hoje em dia, tais redes estão limitadas ás pessoas que sabem trabalhar e têm acesso a computadores. Infelizmente estas pessoas representam uma pequena parte da população, mesmo nos países mais desenvolvidos. O avanço no processamento de voz é necessário para que o cidadão comum possa usar uma grande variedade de aparelhos utilizando somente a sua principal forma de comunicação, a voz. A necessidade de interfaces que usem a voz obriga a desenvolver métodos que permitam o seu reconhecimento automático.

II. PROJECTO CARL

O projecto CARL (Communication, Action, Reasoning and Learning in Robotics)[18] visa o desenvolvimento de um robô capaz de entender, através de um interface amigável, instruções sob a forma de conceitos familiares ao Homem. Conforme é ilustrado na Figura 1, construir um robô com esta capacidade é visto como um problema de integração de quatro domínios:

- interacção Homem-máquina;
- percepção e capacidades sensório-motoras;
- capacidade de decisão;
- aprendizagem.

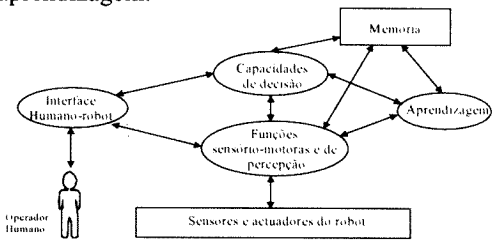


Figura 1 - Diagrama de blocos do projecto CARL.

Apesar destes quatro domínios já terem sido bastante estudados, a sua integração foi pouco tentada.

A filosofia do projecto CARL é baseada na hipótese de que é possível construir um robô inteligente, delineando a estrutura geral e organização da representação do mundo real e dos módulos de execução de tarefas, desde as funções de controlo de baixo nível até às capacidades de decisão de alto nível.

Numa arquitectura deste tipo, os símbolos desempenham um papel muito importante, pois estes vão ser a representação que a máquina terá do mundo. A tarefa de definir símbolos é um problema clássico na programação de robôs. A incorporação de mecanismos de aprendizagem em arquitecturas robóticas vai ser explorada com o intuito de simplificar o problema de programação. O projecto CARL visa uma aprendizagem do robô, não apenas ao nível de conceitos, mas também de tarefas.

É essencial que o robô tenha um interface de linguagem natural, pois este é considerado o único interface aceitável

* Projecto de 5º Ano e Estágio final de Licenciatura em Matemática Aplicada à Tecnologia, Universidade do Porto.

resolver o problema dos símbolos, de modo a tornar o robô de simples utilização [14].

A. *Um robô chamado Carl*

O projecto CARL está a utilizar a plataforma móvel PIONEER 2-DX da ActivMedia Robotics [17]. a qual está dotada, entre outras coisas, de:

- três rodas (duas rodas motrizes e uma direccional);
- 16 sonares repartidos em dois conjuntos de 8 (um conjunto à frente e outro atrás);
- pára-choques com sensores de colisão;
- um computador baseado no processador Pentium 266 MMX da Intel, com 64 MB de memória RAM e com o sistema operativo Linux instalado;
- 1 saída de áudio (para a coluna de som);
- 1 entrada de som (para o microfone);
- 1 ligação para uma bússola;
- uma ligação *robot-to-desktop* para comunicação com um PC, entre outras possíveis ligações a outros acessórios;
- também é acompanhado pelo *software* Saphira. Este permite ler informação proveniente dos sonares, controlar os motores, etc.

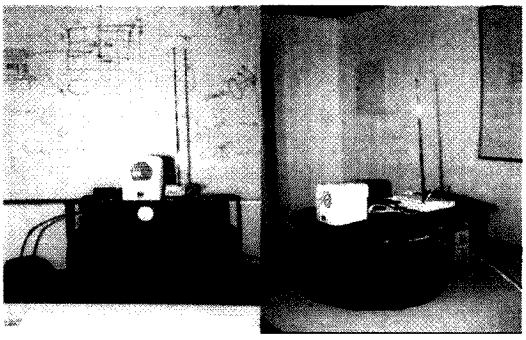


Figura 2 - Imagens do Carl.

III. DESCRIÇÃO GERAL DA INTERFACE

Podemos dividir a interface em quatro blocos principais, tal como se pode ver na Figura 3.

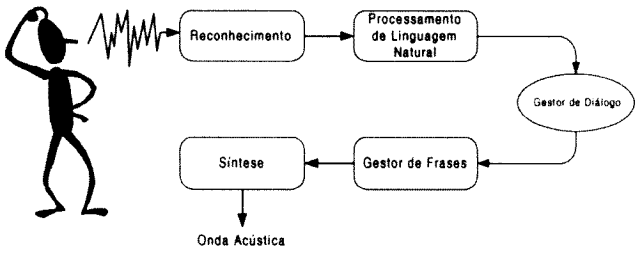


Figura 3 - Blocos da interface de voz

Temos o bloco de reconhecimento que converte o sinal acústico (sinal de voz) numa sequência de fonemas que formam cada palavra e posteriormente a frase. A frase reconhecida apenas nos traz informação disso mesmo - o que foi reconhecido (frase mais provável de ter sido dita pelo interlocutor). Esta informação em si não tem grande significado ao nível da máquina. Como exemplo, podemos ter como frases reconhecidas: *Meio-dia menos um quarto, onze horas e quarenta e cinco minutos, onze e quarenta e cinco*; todas estas frases, para nós, têm o mesmo significado mas para a máquina, são apenas um conjunto de palavras constituindo no seu conjunto uma frase.

É pois necessário atribuir um significado a estas, para que todas elas sejam compreendidas da mesma forma. Para o efeito existe o bloco de processamento de linguagem natural.

Para que a máquina (robô) possa exprimir através da voz *o que lhe vai na alma*, é necessário um bloco de síntese que transforme um conjunto de palavras (frase) num sinal acústico (voz) perceptível pelo ser humano.

Para gerir todo este processo de reconhecimento e síntese há a necessidade de um quarto bloco – o gestor de diálogo. Este bloco é responsável pelas regras de comunicação para que seja de facto possível dialogar de uma forma amigável com o robô.

Na figura aparece um quinto bloco – gestor de frases. Este bloco pode também ser considerado parte integrante do gestor de diálogo. Serve sobretudo para não tornar monótonos os diálogos, ou seja, evita que numa dada situação o robô diga sistematicamente a mesma frase.

IV. RECONHECIMENTO

O bloco de reconhecimento é o bloco mais crítico e onde se nos depararam a maioria dos problemas inerentes ao interface de voz. As dificuldades surgem quando temos um elevado nível de ruído ambiente: conversas paralelas à frase a ser reconhecida; mudança súbita de contexto por parte do orador; ruído dos motores do robô também pode afectar o processo de reconhecimento.

A escolha do reconhecedor a usar teve em conta as contribuições de ruído; a dimensão do vocabulário usado; a dependência/independência do orador [3]. O requisito principal para o reconhecedor é o de que permita ao utilizador a utilização de linguagem o mais próxima possível da que habitualmente utiliza para interagir com seres humanos. Para isso tem de permitir a utilização de discurso contínuo, ser independente do utilizador sem necessidade de treino, lidar com características da voz natural como as hesitações e responder em tempo real [3].

Um reconhecedor que vai ao encontro destas características é o *grapHvite* [11], desenvolvido pela Entropic (empresa actualmente parte da Microsoft). O *grapHvite* fornece várias ferramentas para o desenvolvimento de aplicações de reconhecimento de voz das quais destacamos a sua *Hidden (Markov Model) ToolKit Application Interface* (HTK API), denominada de HAPI [10] que contém todo um conjunto de rotinas em *C* para o processo de reconhecimento.

Além das rotinas da HAPI, um reconhecedor necessita basicamente de três fontes externas de dados:

- *Lattice*
- Dicionário
- Ficheiro de configuração

A *Lattice* irá conter todas as hipóteses de frases e palavras possíveis de serem reconhecidas. O dicionário contém todas as palavras usadas na aplicação, juntamente com a sua transcrição fonética. O ficheiro de configuração contém a localização de todos os ficheiros externos necessários, bem como todos os parâmetros de inicialização por defeito da HAPI.

O algoritmo de reconhecimento encontra-se na Figura 4.

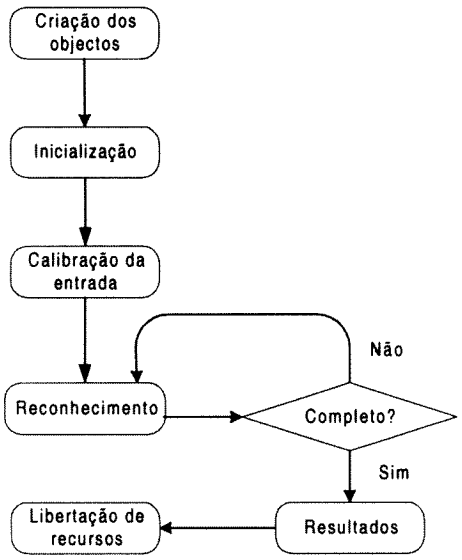


Figura 4 – Algoritmo de reconhecimento

Como resultado o reconhecedor pode devolver para além da frase reconhecida um nível de confiança na frase reconhecida. Na implementação efectuada, esse nível foi utilizado para rejeitar frases pouco prováveis.

V. PROCESSAMENTO DE LINGUAGEM NATURAL

O resultado do reconhecimento em si não traz informação útil para o robô. Há a necessidade de dar um novo formato a essa informação.

É feita uma análise semântica [1] à informação proveniente do reconhecedor, baseando-nos numa gramática específica APS [5] (mais detalhes sobre APS no Apêndice 1). Existe desta forma uma grande dependência entre a gramática da *lattice* e a usada para análise semântica.

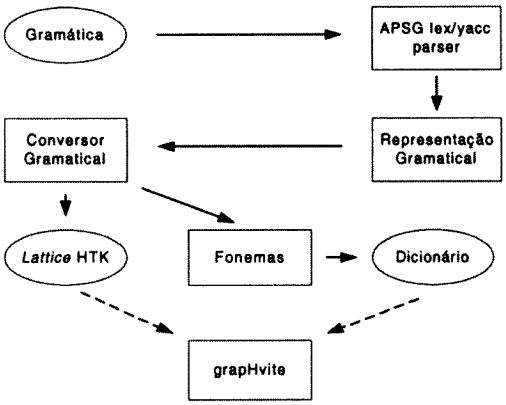


Figura 5 - Conversor gramatical

Para minimizar potenciais fontes de erro provenientes de alterações nas gramáticas, foi desenvolvido um conversor gramatical que, através de apenas uma gramática, gera a *lattice* e o dicionário para o reconhecimento, ficando desta forma a informação coerente nos três sítios, ou seja, todo o processo de compreensão da linguagem é controlada por apenas uma gramática. O processo de conversão é apresentado na Figura 5.

A análise semântica foi feita em conformidade com a representação HRCL (Human Robot Communication Language) apresentada em [14]. (ver Apêndice 2).

Um exemplo de reconhecimento e análise semântica é apresentado de seguida:

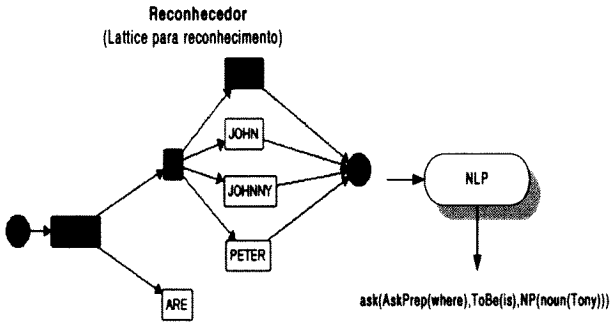


Figura 6 – Análise semântica

O interlocutor diz a frase "Where is Tony"; esta informação ao passar pelo módulo NLP, fica com o seguinte aspecto:

```
ask(AskPrep(where),ToBe(is),NP(noun(Tony)))
```

onde é indicado ao robô que o interlocutor está a perguntar algo "ask" e que a frase é constituída por uma preposição "where", por um verbo To *Be* "is" e por fim um sintagma nominal (NP) com um nome "Tony".

Para a parte de síntese, também foram usadas ferramentas de processamento de linguagem natural para geração aleatória de frases. A API usada para o processamento de linguagem natural foi desenvolvida na Universidade de Aalborg, Dinamarca, pela equipa de Tom

Brøndsted, tendo sido criadas por nós novas rotinas adequadas à aplicação em causa.

VI. SÍNTESE

O módulo de síntese de voz transforma uma mensagem proveniente do gestor de diálogo numa onda acústica. Além da habitual conversão de texto para voz é competência deste módulo transmitir informação paralinguistica, como a emoção. Pretende-se que consoante o tipo de mensagem a sintetizar (informação, pedido, mensagem urgente, ..) sejam utilizados parâmetros do sintetizador (como o tom, intensidade e velocidade) adequados. É nossa convicção que mesmo uma abordagem rudimentar será apreciada pelos utilizadores.

A. Requisitos

Os principais requisitos para o módulo de síntese são: a voz sintética tem de ser inteligível e o mais natural possível; deve ser possível manipular a velocidade, o volume e o tom (*pitch*); o sistema deve possuir boa capacidade de normalização de texto, para lidar com acrónimos, números, etc.; existir uma boa interface para programação (API); poder ser usada em Linux e, se possível, em Windows.

B. Sistema utilizado

Para síntese de voz estão disponíveis várias ferramentas. Foram testados dois sintetizadores: o Festival [2] e o *ViaVoice™ Outloud da IBM.*

Após alguns testes (ver [7]) foi escolhido o sintetizador oferecido pela IBM: o *ViaVoice™ Outloud* [3]. Nos testes este sintetizador revelou-se o mais rápido, de qualidade superior e exigindo recursos computacionais (memória e espaço em disco) muito inferiores ao Festival.

Este sintetizador permite alteração de inúmeros parâmetros da voz sintetizada, conseguindo assim atribuir emotividade e identidade à voz do robô.

A síntese não foi alvo de grande pesquisa da nossa parte, tendo sido implementado apenas um conjunto de frases pré-programadas para vários contextos e com um pouco de emoção. O tom, velocidade e intensidade utilizado para transmitir mensagens urgentes, como bateria fraca, é diferente do utilizado para simples informações.

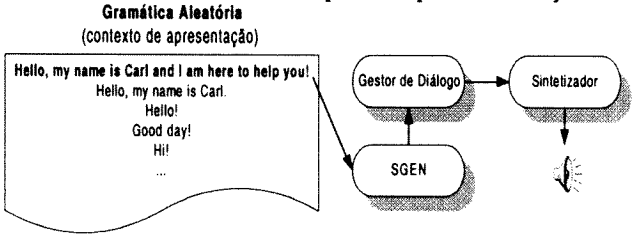


Figura 7 – Escolha de uma frase para síntese

Para melhorar a comunicação as mensagens geradas pelo robô devem variar. Esta variação pode ser conseguida através da utilização de um processo aleatório de escolha dos vários componentes das mensagens. Na implementação efectuada foi utilizado um gerador de mensagens aleatórias com base numa gramática baseado na ferramenta SGEN (Sentence GENerator) do CPK NLP [6]. Foi definida uma gramática para síntese, contendo várias sub-gramáticas para diferentes contextos de conversação. Destas sub-gramáticas é então escolhida uma frase aleatoriamente como o ilustrado na Figura 7.

VII. INTERLIGAÇÃO

A interface foi desenvolvida em *UNIX*, estando cada bloco atrás descrito, implementado como um processo único que comunica com os outros através de troca de mensagens ([7] cap. 5). O bloco de Processamento de Linguagem Natural, não se encontra estanque num processo, mas faz parte dos módulos de reconhecimento e gestão de diálogo. O diagrama de blocos da interface final pode ser visualizado na Figura 8.

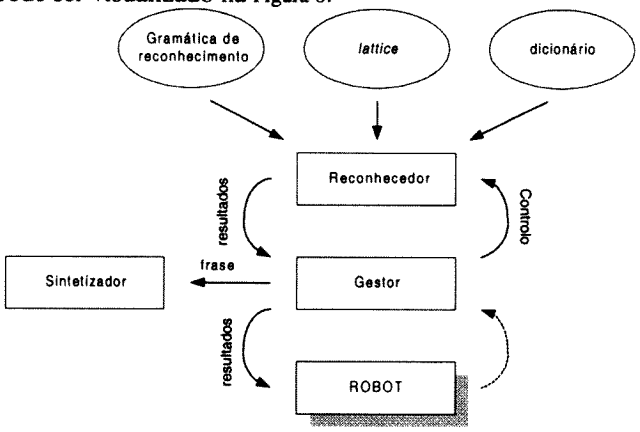


Figura 8 - Interface de voz

VIII. AVALIAÇÃO DO DESEMPENHO DO RECONHECEDOR

Existe um programa para avaliar o desempenho do reconhecedor. O método é o que normalmente é utilizado nas avaliações de sistemas de reconhecimento pelo *National Institute of Standards and Technology* (NIST) americano [19], pois só assim é que se pode comparar resultados com outros projectos/publicações. Este compara as frases ditas pelo orador com as frases reconhecidas. Para melhor perceber o funcionamento deste vai ser dado um exemplo:

```
o orador diz: "Teixeira is in Aveiro"
o robô reconhece: " Is Teixeira in Aveiro now"
```

O programa vai tentar, a partir da frase pronunciada, chegar à frase reconhecida e escolhe a maneira que seja menos penalizada. Para isso pode realizar 3 operações com as seguintes penalizações:

- substituir palavras (S), penalização: 10;
- inserir palavras (I), penalização 7;
- remover palavras (R), penalização 7.

Por exemplo, duas formas para atingir a frase reconhecida a partir da pronunciada seriam:

primeiro caso:

Original	Teixeira	Is	in	Aveiro		
Reconhe-cida	Is	Teixeira	in	Aveiro	Now	
Operação	I	S			I	
Penaliz.	10+	10+	0+	0+	7	=27

segundo caso:

Original		Teixeira	is	in	Aveiro		
Reconhe-cida	Is	Teixeira		in	Aveiro	Now	
Operação	I		R			I	
Penaliz.	7+	0+	7	0+	0+	7	=21

Entre estes dois casos o *software* escolheria o segundo caso. Para avaliar o melhor o programa utiliza um algoritmo de programação dinâmica.

IX. RESULTADOS

Foram feitos alguns testes ao interface, que nos permitiram tirar algumas conclusões acerca do trabalho realizado.

A. Primeiro teste de desempenho e efeito da distância ao microfone

Os testes basearam-se em pronunciar quatro vezes as frases abrangidas pela gramática e foram feitos para uma distância curta ao microfone e para uma distância de aproximadamente 2 metros do microfone. O microfone usado para reconhecimento, foi o microfone da Labtec®, LVA-7280 [20], que é um *array* de microfones direccional com um DSP que efectua cancelamento do ruído de fundo.

Os resultados apresentam-se de seguida em forma de gráfico. Apresenta-se a taxa de palavras correctamente reconhecidas WRR (do Inglês *Word Recognition Rate*), percentagem de inserções WIR (de *Word Insertion Rate*) e nível de confiança devolvido pelo reconhecedor para a frase CONF (de *Confidence).*

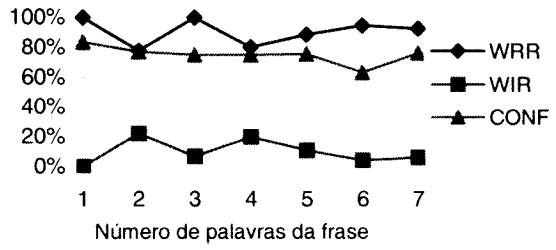


Figura 9 - Reconhecimento perto do microfone

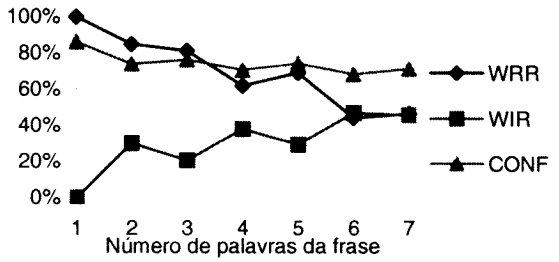


Figura 10 - Reconhecimento a 2 metros do microfone

Através da análise dos gráficos, verificamos que para distâncias curtas foram obtidos resultados aceitáveis. A taxa de reconhecimento de palavras (WRR) tem algumas oscilações mas tende a variar em torno de um valor constante, já para distâncias maiores esta tende a decrescer com o aumento do número de palavras.

B. Teste das gramáticas desenvolvidas

A gramática utilizada nos testes apresentados anteriormente foi posteriormente melhorada [13]. O teste às gramáticas visa quantificar a percentagem de frases gramaticalmente incorrectas aceites pela gramática original e pela gramática melhorada.

O teste consistiu em gerar uma frase aleatória da gramática. O orador decide se a frase é correcta do ponto de vista gramatical. O processo de geração aleatória e classificação de frases continuou até terem sido geradas 200 frases válidas.

Os resultados obtidos no teste das gramáticas são claros. Como se pode verificar na Figura 11, mais de 90% das frases permitidas pela gramática melhorada são gramaticalmente correctas, enquanto que menos de 60% das frases possíveis com a gramática original estão correctas do ponto de vista gramatical.

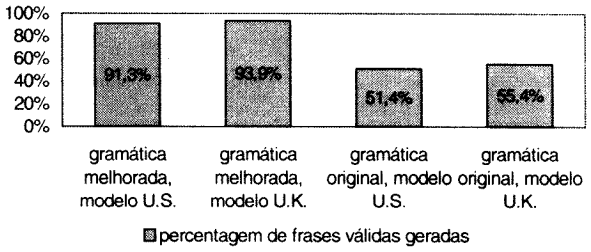


Figura 11 - Desempenho das gramáticas.

C. Teste aos modelos HMM

O teste aos HMM's [12] pretendeu medir o desempenho do reconhecedor, para um determinado orador e com diferentes modelos para inglês (U.K. e U.S.A). Deve ficar claro que o resultado deste teste não pode ser utilizado para saber qual o melhor modelo. Apenas se poderá dizer que o orador escolhido fala inglês mais parecido com o praticado no Reino Unido ou com o praticado nos Estados Unidos da América. Este teste é útil para determinar qual

o desempenho do reconhecedor com os modelos actuais e também para ilustrar a importância do treino dos modelos. Neste teste optou-se pela utilização de um microfone sem fios (Shure TCHS) para evitar a degradação dos resultados com a distância do interlocutor humano ao robô.

Para este teste, o gerador aleatório de frases foi utilizado, com o objectivo de obter sugestões de frases que garantidamente a gramática do reconhecedor está preparada para aceitar. Cada frase correcta foi proferida perante o robô. Isto foi feito para 200 frases correctas.

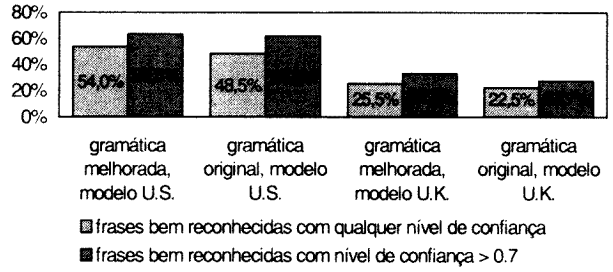


Figura 12 - Frases correctamente reconhecidas.

A percentagem de frases correctamente reconhecidas é apresentada na Figura 12. Como se pode ver, utilizando o mesmo conjunto de palavras, os modelos de voz para os Estados Unidos da América (modelo U.S.) obtiveram sensivelmente o dobro da percentagem de frases bem reconhecidas que os modelos de voz para o Reino Unido (modelo U.K.). Assim é de esperar que, ao treinar modelos para o inglês que se pratica em Portugal (ou pelo menos para as pessoas envolvidas neste projecto), a percentagem de frases correctas seja superior.

É igualmente importante analisar o número de palavras bem reconhecidas. Por vezes, mesmo que uma palavra seja mal reconhecida, esta não influencia o sentido da frase, ou o erro poderá não influenciar a conversa. Por exemplo:

```
o orador pergunta: "Where is the toilet ?"
e o robô reconhece: "Where is a toilet ?"
```

A frase não foi bem identificada, mas a ideia é a mesma.

O número de palavras bem reconhecidas no decorrer do teste encontra-se no gráfico da Figura 13.

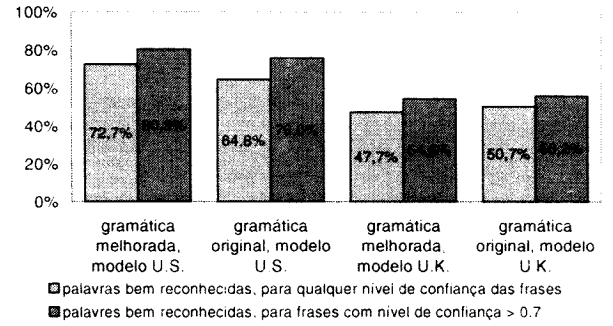


Figura 13 - Palavras bem reconhecidas (WRR)

Como seria de esperar, os modelo U.S. tiveram um melhor desempenho que os U.K e não houve diferenças relacionadas com as gramáticas. No entanto é de referir que a percentagem de palavras bem reconhecidas com modelos U.K. é cerca de 50%.

Esta medida (WRR) torna-se insuficiente, pois não é penalizada quando faltam palavras, quando estas são mal reconhecidas e quando são reconhecidas palavras que não foram ditas. Existe uma medida, designada nos quadros 1 e 2 por "Percent accuracy", que penaliza a percentagem de palavras bem reconhecidas do seguinte modo:

$$\text{Percent accuracy} = \left(\frac{T - S - I - R}{T} \times 100 \right)\%$$

onde T é o número total de palavras, S o número de palavras substituídas, I o número de palavras inseridas e R o número de palavras removidas.

O gráfico da Figura 14 mostra os resultados obtidos para frases com nível de confiança acima de 0.7. Em virtude de o robô só aceitar frases nestas condições, só é interessante analisar este caso. Neste gráfico a percentagem de "palavras correctas" é calculada com a medida "Percent accuracy". Ao somar para uma coluna, as diferentes percentagens (palavras correctas, substituídas, inseridas e removidas), o total pode não dar exactamente 100% devido a erros cometidos pela aproximação.

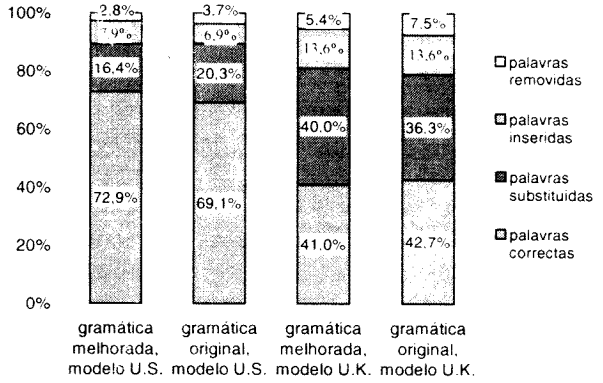


Figura 14 - Análise das palavras

Naturalmente a percentagem de palavras correctas com esta nova medida é inferior à revelada pela Figura 13.

Verifica-se que a percentagem de palavras correctas é superior quando o modelo utilizado é o U.S.. Este resultado seria de esperar, pois os resultados apresentados anteriormente apontam para um melhor desempenho deste modelo com o orador que efectuou o teste.

O número de palavras substituídas reflecte o número de palavras mal reconhecidas, isto é, o orador pronúncia uma palavra e o robô reconhece outra. É nesta medida que existe a maior diferença entre os modelos. Este resultado é natural pois o modelo U.S. ajusta-se melhor a este orador que o modelo U.K.. Ao observar os gráficos, repara-se que a percentagem de palavras substituídas é, no modelo U.K., quase igual à percentagem de palavras correctas. Este dado revela um desempenho fraco deste modelo com este orador. Para o modelo U.S. substituíram-se cerca de três vezes menos palavras que aquelas identificadas correctamente.

A inserção de palavras podem ter duas causas: (1) restrições impostas pela gramática; (2) ruído.

Pode-se perceber melhor como é que restrições impostas pela gramática podem conduzir à inserção de palavras através de um exemplo:

```
o orador diz: "This is the computer lab"
o robô reconhece: "Doty is in the computer lab"
```

A única palavra mal reconhecida foi "This". Devido a ter sido identificada a palavra "Doty" e a gramática não permitir a frase "Doty is the computer lab", foi acrescentada a palavra "in". Obviamente isto só acontece se a frase reconhecida, mesmo com a inserção, tiver um nível de confiança superior a qualquer outra alternativa. As palavras removidas podem provir, à semelhança das inseridas, de restrições impostas pela gramática.

D. Primeira demonstração

Como primeira demonstração do trabalho o robô foi auxiliado através de comandos por voz na tarefa de navegação pelo edifício do Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA) com resultados encorajadores. O filme desta primeira aventura do Carl foi apresentado no *ECAI Workshop on Service Robotics* [16], que decorreu em Berlim, na *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2000)* [15], no Japão, e durante a Semana da Ciência e Tecnologia de 2000.

X. CONCLUSÕES

Através dos resultados obtidos, verificamos que há ainda muito trabalho a desenvolver, especialmente quando o orador se encontra afastado do microfone. Os resultados indicam que:

- O *array* de microfones ainda não consegue resolver o problema de captação da voz com o interlocutor humano algo afastado do robô;
- O desenvolvimento de reconhecedores para a HRCL, linguagem em desenvolvimento para a interacção com o robô, é possível e facilitada pela utilização de ferramentas como o CPK NLP toolkit;
- Com o microfone sem fios, foi já possível que o robô desse o primeiro passeio recebendo ordens de um operador humano;
- Os modelos HMM ainda não são os mais adequados, sendo necessário o seu retreino para as condições de ruído e forma de falar dos utilizadores.

A. Trabalho Futuro

No futuro, que se espera próximo, pretende-se que sejam abordados, entre outros, os seguintes assuntos:

- a melhoria do desempenho do sistema de reconhecimento. Uma abordagem poderá passar por colocar um módulo de pré-processamento antes do reconhecedor. Para melhores resultados, os modelos do reconhecedor devem ser retreinados para os diferentes meios (salas, corredores) que o robô irá encontrar, sendo activados quando este os estiver a percorrer. Foram incluídas em [7] outras sugestões para um possível melhoramento do reconhecedor;
- avaliar novas hipóteses para o reconhecimento, como o sistema de reconhecimento *IBM ViaVoice*;
- evolução do vocabulário, frases reconhecidas e geradas pelo robô;
- melhoria da arquitectura e da integração com os subsistemas de navegação e aprendizagem do CARL;
- o sintetizador deve ser estudado profundamente de forma a ser possível dar uma identidade própria à voz do robô;
- avaliação exaustiva do desempenho do sistema. O sistema deverá ser testado em tarefas reais, como a de servir de guia a um visitante do IEETA;
- conversão do processo de reconhecimento e síntese para a utilização da língua portuguesa .

AGRADECIMENTOS

Este trabalho foi financiado pela FCT através do projecto CARL PRAXIS/P/EEI/ 12121/1998. Agradece-se a todos os que colaboraram no projecto CARL durante o ano lectivo de 1999/2000, em particular, ao Eng. Ricardo Ruivaco. Agradece-se, também, ao Departamento de Matemática Aplicada da Faculdade de Ciências da Universidade do Porto na pessoa da Doutora Ana Paula Rocha ter tornado possível o estágio do Mário Rodrigues no IEETA.

APÊNDICE 1. FORMALISMO APSG

As gramáticas APSG (Augmented Phrase Structure Grammar) [3] são gramáticas usadas para geração de *frames* semânticas através de frases definidas na gramática. O princípio básico de funcionamento da gramática é caracterizar cada palavra com as suas características de relevo, importantes à aplicação. Por exemplo a palavra *this* poderia ser categorizada como

```
{lex=this,cat=pron,type=demonstrative},
```

o que indica que esta palavra é um pronome *(pron)* do tipo *(type)* demonstrativo. Podemos atribuir as categorias que quisermos a cada palavra, por exemplo onde está *type*, poderia estar *tipo*.

O formato APSG inclui quatro tipos de regras: *axiom, structure building rules (B-rules), lexical rules (L-rules) e semantic rules (M-rules)*. Para a criação de uma gramática, começa-se por definir um *axiom*, que em termos linguísticos corresponde à frase *(sentence)*:

[{cat=s}].

Posteriormente é necessário definir as *B-rules* que são nada mais nada menos do que a estrutura das frases. Se tivermos uma frase constituída pelo sintagma nominal (noun phrase, NP), por um verbo, por exemplo *To Be*, e por outro verbo no gerúndio, teríamos a regra:

```
{cat=s, stype=frase1}
{
    {cat=np},
    {cat=verb,vtype=tobe},
    {cat=verb,vtype=gerund},
}.
```

onde *np* seria:

```
{cat=np,stype=person}
{
    {cat=noun,semtype=person}
}.
```

Com esta regra qualquer uma das frases seguintes seria aceite:

```
Tony is eating.
They are riding.
```

As *L-rules* são as regras mais simples e consistem basicamente na categorização das palavras usadas na gramática. Por exemplo as palavras *Tony* e *is* podem ser categorizadas como:

```
{lex=Tony,cat=noun,type=proper,semtype=person}.
{lex=is,cat=verb,vtype=tobe,nb=singular}.
```

Por palavras: *Tony* é um nome próprio de uma pessoa e a palavra *is* é um verbo do tipo *to be* no singular.

As *M-rules*, são regras semânticas, não necessárias à gramática, que permitem dar uma forma ou significado semântico às frases. Estas regras consistem em duas partes, separadas por '/' a primeira contendo o formato semântico a dar à informação, a segunda descrevendo a árvore sintáctica à qual se deve aplicar o formato em questão. A parte da condição pode conter *links* para outras *M-rules* definidas na gramática; estes *links* consistem em *labels* antecedidos por '#'. Ainda para o exemplo anterior, poderíamos ter:

```
(frase
        (#NP,ToBe($V),Gerund($G))
)
/
{cat=s}
[
        {cat=np}#NP,
        {cat=verb,lex=$V},
        {cat=verb,lex=$G}
].
```

o *link #NP* iria referir-se à estrutura:

```
(NP
    (noun($N))
)
/
{cat=np}
[
    {cat=noun,lex=$N}
].
```

Se a frase em questão fosse *Tony is sleeping*, o resultado semântico seria:

```
(frase(NP(noun(Tony)),ToBe(is),Gerund(sleeping)))
```

APÊNDICE 2. HUMAN-ROBOT COMMUNICATION LANGUAGE

Como já foi referido, o diálogo utilizando linguagem natural é a única forma prática de um utilizador não especialista especificar e ensinar uma tarefa a um robô. Para implementar essa comunicação, o robô terá de ser capaz de gerar e interpretar mensagens (frases). Internamente essas mensagens têm de ser representadas de uma maneira mais formal. A comunidade científica dos sistemas multi-agente desenvolveu ao longo dos anos diversas linguagens para comunicação entre agentes. Provavelmente a mais conhecida é a ACL, acrónimo de *Agent Communication Language* [8].

Para representar as mensagens trocadas entre o robô e o utilizador/professor humano inspiramo-nos na ACL. Em desenvolvimento encontra-se a linguagem HRCL, a nossa *Human-Robot Communication Language* [14]. Para a representação de conhecimento nas mensagens, utiliza-se Prolog, dado que esta linguagem é também a utilizada na implementação dos módulos de decisão e de representação de conhecimento. A ontologia está a ser definida tomando em consideração o tipo de mundo que antecipamos o robô "verá" (dados os sensores instalados) e as capacidades que se pretendem desenvolver. Para simplificar, apenas se considera, por agora, um sistema com apenas um único robô e um único utilizador humano. Algumas "mensagens" para início e fim de diálogos e sub-diálogos foram consideradas. A ideia é de que o robô seja capaz de guardar informação acerca do contexto com a ajuda de uma hierarquia de diálogos e sub-diálogos. Na Tabela I apresenta-se a parte mais importante da linguagem HRCL.

ask	ready	register
ask_if	next	dialogue
tell	rest	dialogue_accept
deny	discard	dialogue_reject
insert	sorry	dialogue_end
delete	standby	dialogue_end_accept
achieve	error	dialogue_end_reject

Tabela I

HUMAN-ROBOT COMMUNICATION LANGUAGE (HRCL)

REFERÊNCIAS

[1] J. Allen, *Natural Language Understanding*. Benjamin/Cummings Publishing Company, 1995.

[2] Alan W. Black, P. Taylor e R. Caley, *The Festival Speech Synthesis Systems (Edition 1.4, for Festival Version 1.4.0)*, 1999.

[3] T. BrØndsted *et al.*, *A Platform for developing Intelligent MultimediaApplications*, Tec Rep R-98-1004. Institute of Electronic Systems, Aalborg University, 1998.

[4] T. BrØndsted, "The Natural Language Processing Modules in REWARD and IntelliMedia 2000+", Em *LAMBDA 25*, Copenhagen Business School, pp. 91-108, 1999.

[5] T. BrØndsted, "Implementing a Task Specific Grammar for Recognition and Parsing using the CPK NLP Suite for Spoken Language Understanding".

[6] T. BrØndsted, "The CPK NLP Suite for Spoken Language Understanding", em *Proc. EuroSpeech,* 1999.

[7] N. Ferreira e N. Ferreira, *Interface Humano Máquina do CARL*, Relatório Final de Projecto 5º ano, Dep. Electrónica e Telecomunicações, Universidade de Aveiro, Julho de 2000

[8] M.R. Genesereth e S.P. Ketchpel, "Software Agents", *Communications of the ACM*, p 48-53, 1994.

[9] *IBM ViaVoice™ Outloud API Reference*, Setembro 1999.

[10] J. Odell, *The HAPI Book (for HAPI Version 1.4) – A description of the HTK Application Programming Interface*, Entropic Ltd, 1999.

[11] K. Power, R. Morton, C. Matheson e D. Ollason, *The grapHvite Book (for grapHvite 1.4 - reference release)*, Abril 1997.

[12] L. Rabiner e B.-H. Juang, *Fundamentals of Speech Recognition*, 1993.

[13] M. Rodrigues, *Colaboração no desenvolvimento de um interface de voz*, Relatório de Estágio, Dep. Matemática Aplicada, Faculdade de Ciências da Universidade do Porto, Setembro, 2000.

[14] L. Seabra Lopes A. Teixeira, "Teaching Behavioral and Task Knowledge to Robots through Spoken Dialogues", em *My Dinner with R2D2: Natural Dialogues with Practical Robotic Devices*, AAAI-2000 Spring Symposium Series, Stanford, Março 2000.

[15] L. Seabra Lopes e A. Teixeira, "Human-Robot Interaction through Spoken Language Dialogue", em *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Takamatsu, Japão, 2000.

[16] L. Seabra Lopes, K. L. Doty, F. Vaz e J. A. Fonseca. "Service Robotics and the Issue of Integrated Intelligence", em *Proc. ECAI Workshop on Service Robotics - Applications and Safety Issues in an Emerging Market*, Berlim, pp. 35-44, 2000.

[17] http://www.activrobots.com/robots/p2dx.html

[18] http://www.ieeta.pt/CARL

[19] http://www.nist.gov/speech/tests/index.html

[20] http://www.labtec.com/

Goodears – Robot orientado por um farol acústico

João Reis, Paulo Santos, José Neto Vieira e Luís Almeida

Resumo **– Este artigo descreve de forma sucinta o robot *Goodears*, um robot autónomo capaz de se orientar por uma fonte sonora. O sistema de orientação possui dois microfones espaçados de uma distância D_M. A direcção da fonte sonora é obtida com base no atraso relativo entre os sinais recebidos no microfones. Foram seguidas duas abordagens relativamente à caracterização da fonte sonora: previamente conhecida e desconhecida. No primeiro caso usaram-se *matched filters* combinados com uma PLLs tendo-se obtido muito bons resultados. O robot era efectivamente capaz de seguir uma fonte sonora com características pré-definidas mesmo emitindo com reduzidos níveis de potência e em presença de ruído. No segundo caso fez-se uma correlação cruzada entre os dois sinais tendo-se obtido resultados positivos embora inferiores aos da outra abordagem.**

Abstract **- This paper presents a brief description of the *Goodears* robot which is an autonomous robot capable of following an acoustic source. The robot has two microphones spaced by a given distance D_M. The direction of the sound source is computed based on the delay between the microphone signals. Two distinct approaches have been followed concerning the acoustic source: previously known and unknown. In the first case, a pair of *matched filters* was used, combined with one PLL each, resulting in a very good performance. The robot could effectively follow a sound source with pre-determined characteristics even when emitting with low power and in the presence of noise. In the second case a cross correlation was carried out between the signals of the two microphones. The results were positive although not as good as those of the first approach.**

I. INTRODUÇÃO

O robot *Goodears* foi desenvolvido por um grupo de dois alunos de Engª Electrónica e Telecomunicações no âmbito do seu projecto de fim-de-curso. O objectivo principal que presidiu à sua construção foi o de estudar mecanismos que permitissem determinar a direcção de uma fonte sonora, não baseados na intensidade, de forma a que um agente autónomo, um robot, se pudésse guiar por essa mesma fonte. Um robot equipado com tal sistema poderia, por exemplo, reagir a um chamamento acústico, deslocando-se em direcção ao chamador. Pretendia-se, ainda, desenvolver o trabalho apenas com dois microfones, à semelhança dos seres vivos que possuem dois ouvidos. Por último, pretendia-se averiguar a possibilidade de instalação do sistema numa plataforma de baixo consumo que permitisse o funcionamento a partir de um conjunto de baterias NiCad pequenas (AA). A plataforma utilizada foi um kit DSP baseado no processador TMS320C30 com o interface analógico adequado quer à ligação dos microfones quer ao controlo dos motores.

Relativamente à caracterização da fonte sonora, existem duas situações distintas, uma em que a fonte é conhecida e outra em que não é. O primeiro caso permite tirar partido das propriedades do som emitido pela fonte para o conseguir identificar e discriminar, mesmo imerso em ruído ambiente. O segundo caso é mais complexo, esperando que o robot reaja a um som qualquer. Para estas duas situações seguiram-se duas abordagens diferentes. No primeiro caso, usou-se um *matched filter* associado a cada sinal. A saída desse filtro era aplicada a uma PLL realizada em software que permitia obter uma sequência regular de pulsos. Posteriormente era medido o desfasamento temporal entre os pulsos das duas sequências. No segundo caso aplicou-se uma correlação cruzada entre os sinais dos dois microfones, medindo-se o atraso relativo dos dois sinais através do desvio do pico da correlação.

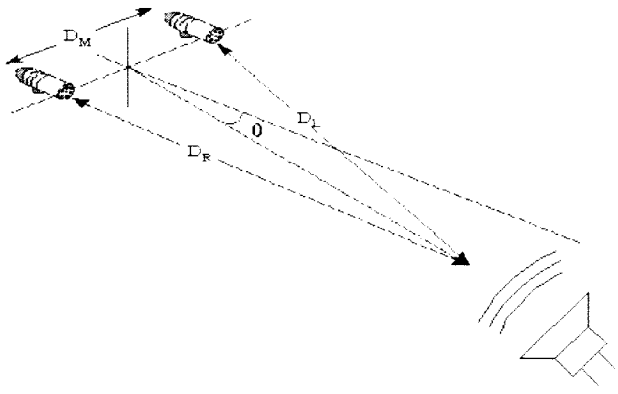


Fig. 1 – Visão geral do sistema com os dois microfones separados de uma distância D_M e a geometria de detecção da posição da fonte sonora.

Em ambas as abordagens, a determinação da direcção da fonte sonora (Sound Direction Of Arrival - SDOA) baseia-se no cálculo da diferença de tempo interaural, em inglês Interaural Time Diference (ITD), que consiste em determinar o desfasamento entre os sinais recebidos nos dois microfones (fig. 1).

Para distâncias D_L e D_R (de left e right) muito maiores que o espaçamento dos microfones D_M (fig. 1), o ângulo a que está a fonte pode ser aproximado por

$$\theta \cong \sin^{-1}\left(\frac{D_L - D_R}{D_M}\right) \quad (1)$$

ou em termos da ITD em amostras

$$\theta \cong \sin^{-1}\left(\frac{v_s ITD}{D_M}\right) \quad (2)$$

em que v_s é a velocidade do som. O cálculo da ITD é efectuado de forma diferente consoante a abordagem seguida, i.e. com fonte conhecida ou não. Em qualquer dos casos, é necessário que os sinais sonoros emitidos pela fonte tenham um conteúdo espectral relativamente largo.

Na secção seguinte é exposta a utilização da técnica de *matched filter*, utilizada na abordagem com fonte sonora conhecida. Na secção III expõe-se a utilização da correlação cruzada para obter a direcção de fontes sonoras desconhecidas. O artigo termina com algumas observações críticas relativas à implementação de ambas as abordagens, à contrução do robot e seu desempenho.

II. PRIMEIRO MÉTODO: MATCHED FILTER

Uma das técnicas de estimação de atraso temporal mais conhecidas baseia-se na utilização de *matched filters*. Esta abordagem pode ser facilmente implementada, embora a respectiva resolução seja limitada pela largura de banda do sinal transmitido. Neste método é emitido um sinal sonoro conhecido, com uma largura de banda grande (300-3400Hz) relativamente ao espectro audível. Este sinal, designado por *chirp de frequência*, será melhor descrito posteriormente. Na figura 2 encontra-se o diagrama de blocos relativo ao cálculo da ITD usando este primeiro método.

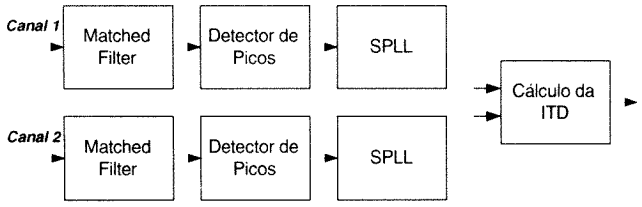


Fig. 2 - Diagrama de blocos do 1º método

Matched Filter

Em muitos sistemas é considerado o problema de recuperar um sinal com uma forma de onda desconhecida a partir de um sinal original, usando toda a sua informação tal como a largura de banda que ocupa, a altura de ocorrência, ou as suas propriedades estatísticas no domínio dos tempos. Contudo, em algumas situações, a forma do sinal que se espera receber é previamente conhecida. Nesse caso, podemos aplicar uma técnica denominada de *matched filtering*. Com esta técnica consegue-se melhorar substancialmente a relação sinal ruído na detecção do sinal. Basicamente, a técnica de *matched filtering* consiste em correlacionar o sinal de entrada, desconhecido, com o sinal esperado. Sempre que um padrão semelhante ao sinal esperado for reconhecido no sinal de entrada, o filtro gera um pico na respectiva saída.

Conceitos

Para se perceber melhor a técnica de *matched filtering* convém relembrar dois conceitos fundamentais da teoria do processamento digital de sinal: o somatório de convolução (3) e o de correlação cruzada (4).

$$y[k] = \sum_{n=0}^{N-1} h[n]x[k-n] \quad (3)$$

$$R_{xy}[\hat{o}] = \sum_{n=0}^{N-1} x[n]y[n+\hat{o}] \quad (4)$$

No primeiro caso é efectuada a multiplicação com acumulação da resposta impulsional $h[n]$ do sistema por uma versão invertida no tempo e atrasada do sinal de entrada $x[n]$. No segundo caso é efectuada a multiplicação com acumulação dos sinais $x[n]$ e $y[n]$ em que este último é atrasado no tempo τ amostras. A diferença mais relevante entre os dois somatórios é que no primeiro, i.e. na convolução, é necessário inverter no tempo um dos sinais originais.

Suponhamos agora que um sinal $x[n]$ é aplicado à entrada de um sistema linear que tem uma resposta impulsional $h[n]$ igual a uma versão de $x[n]$ invertida no tempo. A saída de tal sistema é dada pelo somatório de convolução que, neste caso, coincide com o somatório de autocorrelação do sinal $x[n]$. Um sistema linear assim é chamado de *matched filter*, uma vez que a resposta impulsional do filtro é uma versão do sinal a detectar.

Para facilitar a detecção do sinal $x[n]$ é conveniente que a respectiva função de autocorrelação seja tanto quanto possível semelhante a um impulso. Quando assim é, sempre que o sinal à entrada contém uma versão de $x[n]$, o filtro gera um impulso na saída. O problema da detecção do sinal pré-definido $x[n]$ é então reduzido à procura de impulsos na saída do filtro.

Um sinal que possui uma função de autocorrelação impulsional é o ruído branco. A sua função de autocorrelação com densidade espectral de potência σ, é dada por (5).

$$R_{xx}[\hat{o}] = \acute{o}\ddot{a}(\hat{o}) \quad (5)$$

Outros exemplos de sinais com autocorrelação impulsional são o *chirp de frequência* e uma sequência aleatória de ± 1.

Na prática estes sinais podem apenas ser aproximados, usando uma duração e banda limitada. Contudo, usando valores grandes mas realizáveis para estas grandezas é possível obter sinais com autocorrelação suficientemente

próxima do impulso ideal. Nestas condições a técnica de *matched filtering* tem um bom desempenho, permitindo detectar um sinal pré-definido na presença de ruído mesmo com relações sinal/ruído muito baixas.

Chirp de frequência

O sinal escolhido para ser emitido pelo farol acústico foi o *chirp de frequência*. Este sinal pode ser interpretado como uma sinusóide cuja frequência varia linearmente ao longo do tempo desde uma frequência inicial f_0 até à frequência f_1. Este sinal pode ser descrito como a modulação em frequência de um sinal em dente de serra e representado pela expressão (6) em que $\Psi(t)$ é a fase instantânea.

$$x(t)=A\cos(\psi(t))=\Re e\{Ae^{j\psi(t)}\} \qquad (6)$$

A derivada da fase em ordem ao tempo dá-nos a frequência angular e dividindo por 2π obtemos a frequência instantânea em Hz (7).

$$f_i(t)=\frac{1}{2\pi}\frac{d}{dt}\psi(t) \qquad (7)$$

Por outro lado, pretendemos que a frequência cresça linearmente a partir de um valor inicial f_0 tal como descrito em (8). O declive de $f_i(t)$ é 2μ, e a sua ordenada na origem é f_0. Se o sinal começar em $t = 0$, então f_0 é também a frequência inicial.

$$f_i(t)=2\mu t+f_0. \qquad (8)$$

A partir das expressões (7) e (8), por integração, pode-se obter a fase instantânea $\psi(t)$, resultando na expressão (9).

$$\psi(t)=2\pi\mu t^2+2\pi f_0 t+\phi \qquad (9)$$

Considerando que o *chirp* tem uma duração T, então, o *matched filter* terá como resposta impulsional o sinal dado por (10) que não é mais que o próprio *chirp* invertido no tempo.

$$h(t)=f(T-t) \qquad (10)$$

Detector de picos

O detector de picos tem como função detectar os impulsos gerados pelo filtro com uma energia acima de um dado limiar. Estes correspondem a detecções do sinal pré-definido imerso no sinal de entrada. A definição do limiar tem repercussões importantes. Se o limiar estiver demasiado baixo poderão existir detecções falsas. Se estiver demasiado alto, as ocorrências do sinal pré-definido no sinal de entrada poderão não ser detectadas. Uma vez que o melhor valor para o limiar não é conhecido *a priori*, utilizou-se um algoritmo adaptativo para estimar dinamicamente o seu valor (fig. 3). A saída do *matched filter* é passada por um rectificador de meia onda e depois por um filtro IIR passa-baixo de 1ª ordem com uma largura de banda muito baixa. O limiar para a detecção dos picos é definido para metade do valor de saída do filtro. Com este método o limiar de detecção é automaticamente ajustado quer quando, por exemplo, o robot se aproxima do farol acústico, quer quando se afasta.

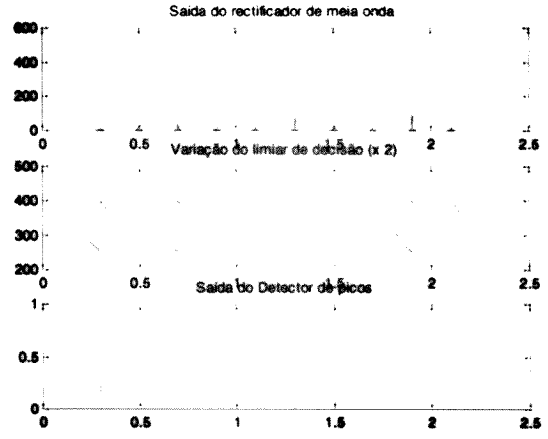


Fig. 3 Evolução do limiar adaptativo do detector de picos.

Software PLL (SPLL) e cálculo da ITD

Antes de fazermos uma explicação do funcionamento da PLL convém relembrar que o sistema é composto por dois canais audio. O sinal obtido em cada um destes canais é submetido independentemente a um *matched filter* e um detector de picos. As saídas de ambos os detectores contém uma sequência de impulsos cujo desfasamento temporal é usado para calcular a ITD conforme referido na secção I e figura 2. Contudo, dado que os dois canais são independentes, poderão ocorrer situações em que é gerado um impulso apenas num canal, podendo esta situação originar erros na determinação da ITD.

Para evitar estas situações, utilizou-se mais um aspecto pré-conhecido do sinal acústico emitido, i.e. o período de repetição do *chirp* (T_p). Assim, tendo em conta eventuais falhas de detecção do *chirp*, a saída do detector de picos é constituída por uma sequência de impulsos espaçados T_pN em que N representa o número instantâneo de períodos entre dois impulsos consecutivos (fig. 4). Esta sequência é aplicada a uma PLL que só sincroniza com a frequência pré-definida de repetição do *chirp* ($1/T_p$). Assim, sempre que a PLL se encontra sintonizada, i.e. *locked*, a respectiva saída é uma sequência de impulsos com um período regular igual ao período de repetição do *chirp*.

A figura 4 descreve a constituição e funcionamento da PLL realizada em *software*. A comparação de fase, i.e. a determinação da diferença entre o período instantâneo e o de referência, é efectuada sincronamente com os impulsos do sinal de referência gerado pela PLL. Deste modo é possível eliminar eventuais picos indesejáveis do sinal de entrada, tornando o sistema mais robusto.

Depois dos sinais de ambos os canais passarem pelas respectivas PLL's, sempre que estas estiverem sincronizadas podemos efectuar a medição do atraso de

uma das sequências relativamente à outra sem risco de cometer erros por ausência de impulsos num dos canais. A determinação deste atraso permite estabelecer o valor da ITD, para posterior localização da fonte acústica no plano azimutal.

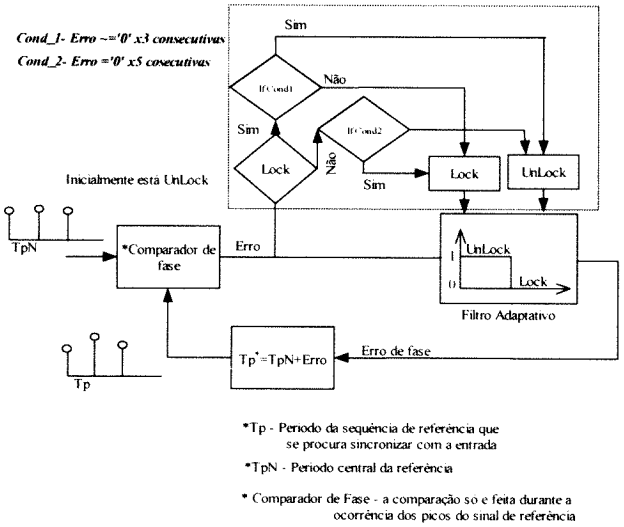


Fig. 4 - Diagrama de blocos da SPLL

III. SEGUNDO MÉTODO: CORRELAÇÃO CRUZADA

Teoria subjacente

Outro método para obter a direcção de uma fonte sonora consiste na utilização da correlação cruzada dos sinais dos dois microfones. Neste caso não existe nenhum sinal pré-definido pelo que o robot se deverá orientar para o som mais forte.

A correlação cruzada entre dois sinais (4) pode ser entendida como uma medida da semelhança entre esses sinais. A função de correlação apresenta um máximo no instante em que a semelhança entre os sinais é também máxima. Assim, o desvio do máximo da função de correlação relativamente à posição central $\tau = 0$ representa o atraso de um sinal em relação ao outro.

Aplicando uma janela temporal aos sinais de ambos os canais e calculando a respectiva função de correlação deverá ser possível determinar o desvio do respectivo máximo e daí calcular a ITD. O sinal, positivo ou negativo, desse desvio define a localização esquerda ou direita da fonte sonora no plano azimutal.

Também neste método se coloca a questão de determinação do máximo da função de correlação. De facto, sempre que os sinais correlacionados tiverem um baixo conteúdo espectral, i.e. uma largura de banda estreita, a função de correlação resulta numa forma alargada, cujo máximo pode ser difícil de determinar. Por outro lado, se os sinais tiverem um conteúdo espectral rico, a função de correlação apresenta um pico relativamente estreito, facilitando a determinação do máximo. Deste modo será de esperar que este método apresente melhores resultados sempre que os sons produzidos para orientação do robot contenham um espectro relativamente largo.

Implementação

Uma das dificuldades práticas associadas a este método consiste no próprio cálculo da função de correlação. De facto, a utilização directa da expressão (4) é muito pesada computacionalmente; exigindo um processador de grande capacidade de cálculo especialmente para janelas de dimensão elevada. Por outro lado, a utilização de janelas justapostas, i.e. sem sobreposição, pode resultar em variações grandes e súbitas nos sinais e, consequentemente, na função de correlação.

Para resolver estes problemas utilizou-se uma janela deslizante em que a janela seguinte, com N amostras de largura, é obtida a partir da anterior pelo deslocamento de uma amostra. Neste caso, comparando os cálculos das funções de correlação de duas janelas consecutivas, existem muitas operações comuns. Para cada τ todos os termos do somatório são os mesmos à excepção do primeiro produto da correlação anterior e último produto da correlação actual. Assim, a correlação para uma nova janela (n+1) pode ser obtida a partir da da janela anterior (n) conforme descrito em (11).

$$R_{xy}(n+1,\hat{o}) = \begin{cases} R_{xy}(n,\hat{o}) - x_{n-\tau}.y_n + x_{n+N}.y_{n+N+\tau} & \hat{o}<0 \\ R_{xy}(n,\hat{o}) - x_n.y_{n+\tau} + x_{n+N-\tau}.y_{n+N} & \hat{o}\geq 0 \end{cases} \quad (11)$$

Para além do cálculo recursivo da função de correlação, os requisitos de capacidade cálculo foram ainda mais reduzidos através da limitação dos valores de τ para os quais a função de correlação era efectivamente calculada. Por exemplo, se o valor máximo absoluto para a ITD corresponde a um desfasamento de 8 amostras, não adianta calcular a correlação para $|\tau| > 8$, poupando assim muito tempo de processamento.

IV. INDEFINIÇÃO FRENTE-TRÁS

A utilização de apenas dois microfones para determinar a direcção de uma fonte sonora com base na IDT não permite destinguir entre uma fonte localizada à frente e outra localizada atrás da linha que une os microfones. Assim, se um farol acústico *F* se encontrar localizado atrás da linha dos microfones o robot verá a imagem virtual *I* localizada à frente dessa linha.

Contudo, como a informação azimutal obtida a partir da IDT é utilizada para orientação do robot, causando um determinado movimento angular em direcção à fonte sonora detectada, o robot acaba sempre por se orientar para o farol acústico real. Este facto, aparentemente estranho, deve-se ao algorítmo de orientação baseado na minimização do erro angular entre a posição da fonte sonora detectada e a orientação do robot e é explicado abaixo.

Na exposição que se segue considera-se que o robot usa controlo diferencial de direcção e que os microfones se encontram sobre a linha que une as rodas do robot, linha esta perpendicular à respectiva orientação.

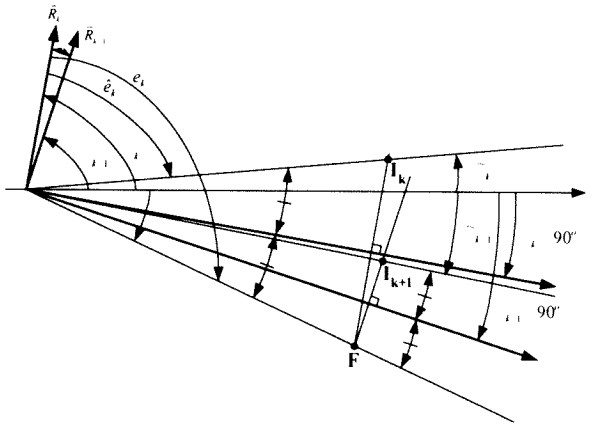


Fig. 5 – Geometria do problema de orientação do robot para o farol F.

Na figura 5 o robot, num dado instante t_k, está orientado na direcção R_k com um ângulo β_k relativamente a um eixo de referência genérico (linha horizontal). A recta que une os dois microfones, perpendicular à orientação do robot, faz então um ângulo de β_k-90° com esse eixo. As expressões (12) permitem obter os erros angulares relativo ao farol F (e_k) e relativo à imagem I_k ($\hat{e}_k$), para esse instante de amostragem. Dada a simetria da imagem e do farol relativa à linha dos microfones, os dois erros são complementares, i.e. $e_k + \hat{e}_k = \pm 180°$.

$$\begin{aligned} e_k &= \alpha - \beta_k. \\ \hat{e}_k &= \tilde{a}_k - \beta_k = [2(\beta_k - 90) - \alpha] - \beta_k = \pm 180 - e_k \end{aligned} \qquad (12)$$

O erro angular efectivamente medido pelo robot está compreendido entre –90° e 90°, de modo que entre o farol real e a imagem, o robot detecta o que estiver à sua frente. Na situação descrita na figura 5 o farol F está atrás do robot e a imagem I_k à frente pelo que o erro medido é o relativo à imagem $\hat{e}_k$.

Utilizando um algorítmo de controlo que minimize o erro medido pelo robot, este efectuará um movimento angular correctivo ε_k na direcção do erro $\hat{e}_k$ (ou e_k se o farol estiver à frente). Assim, no instante seguinte t_{k+1}, o robot encontrar-se-á orientado segundo R_{k+1}, com um ângulo $\beta_{k+1} = \beta_k + \varepsilon_k$ Os erros relativos ao farol real e_{k+1} e à imagem $\hat{e}_{k+1}$ são dados pela expressões (13). A nova posição da imagem é dada por I_{k+1}.

$$\begin{aligned} e_{k+1} &= \alpha - \beta_{k+1} = \alpha - \beta_k - \varepsilon_k = e_k - \varepsilon_k \\ \hat{e}_{k+1} &= \tilde{a}_{k+1} - \beta_{k+1} = \pm 180 - e_{k+1} = \hat{e}_k + \varepsilon_k \end{aligned} \qquad (13)$$

Tendo em atenção que e_k , $\hat{e}_k$ e ε_k têm sempre o mesmo sinal, facilmente se vê pelas expressões (13) que o erro relativo ao farol real diminui, enquanto o erro relativo à imagem aumenta. De facto, enquanto e_k tende para 0, $\hat{e}_k$ tende para 180°. O movimento de orientação para o farol real faz com que a respectiva imagem se afaste para a traseira do robot. É interessante, ainda, reparar que quando o robot se orienta para imagem virtual do farol o erro aumenta em módulo. Contudo, assim que o farol real passa para a frente, i.e. a partir do ponto onde $e_k=\hat{e}_k=\pm 90°$, o robot começa a orientar-se por este e o erro diminui até 0.

Numa situação limite em que o robot se encontre perfeitamente alinhado com o farol mas em direcção oposta (i.e. $\hat{e}_k=0$ e $e_k=180°$), é possível que o algorítmo de orientação não funcione e que o robot siga a imagem do farol, afastando-se deste. No entanto, ao menor desvio deste alinhamento o robot orientar-se-à rapidamente para o farol real pelo que a situação referida dificilmente ocorrerá na prática.

V. OUTRAS QUESTÕES PRÁTICAS

Para além das questões relacionadas com cada método específico de detecção de uma fonte sonora, bem como com a indefinição frente-trás exposta na secção anterior, existem outras questões relacionadas com a construção prática do robot que merecem ser referidas. Em particular, abordaremos o compromisso entre resolução e aproximação de *far field* bem como outras fontes de erro e limitações inerentes ao funcionamento e construção do robot *Goodears*.

No que diz respeito ao primeiro aspecto, a resolução obtida na orientação do robot para o farol acústico é limitada pelo facto de haver um número discreto de ângulos que podem ser medidos. Este número está directamente relacionado com a distância entre os microfones. À medida que esta aumenta, também aumenta o atraso temporal entre os respectivos sinais. Quanto maior fôr este atraso mais amostras temos para representar esta diferença de tempo. Cada amostra tirada neste período representa um ângulo discreto. Para pequenas distâncias, ficamos com um número insuficiente de amostras para representar os ângulos, logo, com uma resolução muito pobre. Por outro lado, se aumentarmos muito a separação dos microfones, aumenta a resolução mas também aumenta a distância mínima da fonte sonora para que possamos usar a aproximação de *far-field*. Esta aproximação corresponde a desprezar a curvatura das ondas sonoras, considerando que estas se propagam como uma onda plana, e é usada no cálculo da posição angular da fonte sonora a partir da ITD. Assim, a determinação da distância entre os microfones deve ser efectuada com base neste compromisso entre resolução e distância mínima do farol acústico.

Para além da limitação na resolução, qualquer dos métodos descritos neste artigo pode ainda ser negativamente influenciado por outros factores. Por exemplo, a existência de reverberações pode levar a consideráveis distorções no sinal sonoro original correspondendo à co-existência de várias fontes idênticas mas com posições distintas.

Por outro lado, a precisão também é limitada pelo facto da velocidade do som aparecer como constante na fórmula

de estimação do ângulo a partir da ITD, quando esta varia ligeiramente com factores ambientais.

Uma importante fonte de ruído está associada ao próprio funcionamento dos motores eléctricos de locomoção do robot. De facto, estes motores não só geram ruído eléctrico como também sonoro, dificultando a determinação da posição do farol acústico. Para evitar este problema, optou-se por fazer a aquisição de som e a locomoção alternamente, sem sobreposição. Se, por um lado, se resolveram as questões de ruído gerado pelos motores, por outro o movimento obtido não é suave mas sim constituído por uma sequência de pequenas deslocações. Os tempos de imobilização para aquisição de som são, contudo, curtos, inferiores a 1 segundo, resultando num movimento quase contínuo.

VI. CONCLUSÃO

Este artigo descreve sucintamente o funcionamento do robot autónomo *Goodears*, sendo dada ênfase à sua capacidade de orientação para um farol acústico. Esta capacidade é conseguida à custa da detecção da posição angular de uma fonte sonora utilizando apenas dois microfones e informação de fase. Testaram-se dois métodos para efectuar a detecção angular da fonte sonora: o primeiro baseado em *matched filters* e utilizando um som pré-definido, um *chirp* de frequências (secção II); o segundo baseado na função de correlação cruzada reagindo a qualquer som com intensidade suficiente (secção III).

O desempenho obtido com o primeiro método foi considerado muito bom uma vez que o robot conseguia seguir o farol acústico eficientemente, mesmo na presença de ruído e com intensidade imperceptível ao ouvido humano. Este bom desempenho fica a dever-se à exploração eficiente do conhecimento apriorístico do sinal emitido.

Com o segundo método, o desempenho obtido foi também positivo mas inferior ao do método anterior. De facto, o desconhecimento do som emitido aumenta substancialmente a complexidade do problema. Para alguns sons mais pobres em conteúdo espectral, o robot perdia-se frequentemente. Contudo, para sons com maior largura de banda, tipo palmas, pancadas secas com um objecto duro, etc., o robot conseguia normalmente encontrar a fonte sonora.

Um dos resultados colaterais mais interessantes foi a observação de que a indefinição frente-trás inerente à utilização de apenas dois microfones não afecta a capacidade de orientação do robot. Este facto é explicado neste artigo na secção IV.

Finalmente, o robot *Goodears* é completamente autónomo, funcionando a partir de um conjunto de pilhas NiCad pequenas, tipo AA. A autonomia obtida depende das movimentações efectuadas pelo robot mas é próxima dos 15 minutos.

BIBLIOGRAFIA

[1] James H. McClellan, Ronald W. Shafer e, Mark A. Yoder: "*DSP First A Multimedia Approach*", Prentice-Hall, 1998

[2] Paul A. Lynn: "An introduction to the Analysis and Processing of Signals", 3ª ed. MACMILLAN PRESS LTD, 1989

[3] Paul A. Lynn: "Introductory Digital Signal Processing", 2ª ed., Wiley, 1998

[4] Roland E. Best: "Phase-Locked Loops", 3ª ed., McGraw-Hill, 1997

Extracção de dados sobre o formato dos lábios nos sons da Língua Portuguesa

Alexandra Grilo, Pedro Duarte, António Teixeira, Armando Pinho

Resumo - **Este artigo aborda o problema da extracção automática da forma dos lábios correspondente a sons característicos da Língua Portuguesa. Com essa finalidade, foi desenvolvido um sistema de aquisição e análise de imagem baseado em algoritmos genéticos, os quais, conjuntamente com outras técnicas mais tradicionais, proporcionam a segmentação da zona da boca. Com base nessa segmentação são calculados alguns parâmetros relevantes para o problema abordado.**

Abstract – **This paper addresses the problem of automatic extraction of lips format parameters for Portuguese sounds. For that purpose, a system for image acquisition and analysis was developed. The system, based in genetic algorithms and more traditional techniques, calculates several relevant parameters for the lips, using the segmented mouth region.**

I. INTRODUÇÃO

Assiste-se cada vez mais ao desenvolvimento de sistemas de síntese utilizando além da voz uma face sintética. As duas modalidades associadas permitem uma maior imunidade ao ruído e maior inteligibilidade. Com o desenvolvimento de sistemas de telecomunicações como o vídeo telefone, pode encarar-se a possibilidade de dotar as máquinas com este tipo de interface, usando "visual speech" [5].

Para isso, é necessário adquirir conhecimento sobre a configuração dos lábios para os vários sons da língua portuguesa. Numa fase posterior, estes conhecimentos podem ser utilizados na construção de um modelo tridimensional dos lábios e finalmente da cara [7].

O objectivo deste trabalho é a implementação de um sistema para a obtenção, de forma automática, de dados sobre a configuração dos lábios durante a locução dos diversos fonemas da língua portuguesa, com especial incidência nos sons vocálicos [9]. Em termos técnicos este projecto envolveu a aquisição de vídeo e posterior aplicação de algoritmos de processamento de imagem para a extracção de contornos [6][10].

II. AQUISIÇÃO

No início deste trabalho, foi proposto implementar um sistema de aquisição sincronizada de áudio e vídeo onde eram utilizadas duas câmaras para obter uma imagem lateral conjuntamente com a frontal. No entanto, tal não se veio a verificar por tal sistema ser demasiado complexo e dispendioso.

Outra das soluções possíveis seria o uso de duas câmaras em simultâneo [1]. No entanto esta solução tinha o inconveniente de não se conseguir o sincronismo automático, isto é, as imagens adquiridas por ambas as câmaras necessitavam de um sinal para se efectuar o sincronismo.

A solução por nós utilizada foi a mais simples e menos dispendiosa, pois utilizava apenas uma câmara para adquirir a imagem frontal. A câmara utilizada foi uma Quick Cam da Connectix que adquiria imagens e vídeos com uma resolução de 320x240 e uma *frame rate* da ordem das 10 *frames* por segundo.

III. SEGMENTAÇÃO

Geralmente, o primeiro passo a dar na análise de uma imagem passa pela segmentação desta. Tal tarefa significa dividir a imagem nos seus diferentes conteúdos. No nosso caso específico, o conteúdo da imagem a segmentar reside na zona interna da boca. Neste trabalho apresentamos dois métodos distintos embora com o mesmo objectivo: retirar de uma imagem facial a informação referente à zona interna da boca quando é pronunciado um som pelo indivíduo filmado.

O primeiro método aqui apresentado, o qual se baseia em algoritmos genéticos, aborda o tema da detecção de contornos de uma forma bastante diferente das dos métodos convencionais, tais como os filtros de Sobel, Prewitt e Frei-Chen [6]. Ao contrário destes métodos, o algoritmo genético tenta detectar os contornos sem processar por completo a imagem, baseando-se na forma como os cromossomas interagem entre si.

A. Segmentação baseada em algoritmos genéticos

No processamento digital de imagem, características tais como contornos, linhas e curvas podem ser detectados utilizando para tal alguns operadores matemáticos entre os quais gradientes, passagens por zero e filtros. A detecção dessas características pode facilitar bastante a compreensão da imagem em questão. No entanto, estas abordagens convencionais exigem o processamento de toda a imagem, processo esse que se pode tornar moroso.

O algoritmo aqui proposto utiliza agentes autónomos que se podem auto-reproduzir, mover e morrer durante a interacção com a imagem digital [8]. Desta forma pretende-se obter um método descentralizado baseado em comportamentos bem definidos, não havendo por isso necessidade de proceder à análise de toda a imagem.

1. Agentes autónomos

Os agentes autónomos operam num ambiente rectangular correspondente à imagem digitalizada. Na imagem digitalizada, cada grelha de oito pixels conectados representa uma localização possível para o agente habitar, quer definitivamente, quer temporariamente (Fig. 1).

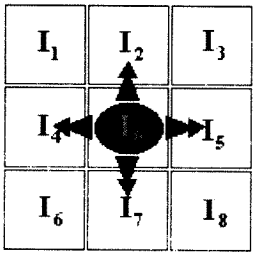


Figura 1 - Cada pixel no ambiente 2D é conectado a oito pixels vizinhos (k=1). Um agente autónomo pode-se auto-reproduzir ou mover em cada um dos oito pixels vizinhos.

A região vizinha de um agente num pixel p é uma região circular centrada no pixel p com raio k. Os pixels que se situam nesta região são chamados pixels vizinhos ao agente.

Um agente autónomo escolhe o comportamento a adoptar após o estudo da concentração de certos níveis de cinzento na sua região vizinha. Se a concentração estiver dentro de uma determinada gama, o agente activará o seu mecanismo de auto-reprodução. Tal concentração é designada por estímulo local.

O estímulo local que selecciona e activa o comportamento do agente é calculado através da distribuição da densidade de todos os pixels da região vizinha cujo níveis de cinzento se localizem perto da intensidade do pixel onde o agente se localiza. Mais especificamente, a distribuição de densidade é definida por

$$D^{K}_{I(i,j)} = \sum_{s=-k}^{k}\sum_{t=-k}^{k}\left\{\frac{1}{\left\|I(i+s,j+t)-I(i,j)\right\|} < \delta\right\}$$

onde k representa o raio da região de vizinhança centrada no ponto de agente (i,j), s e t indices de um pixel pertencente à vizinhança, I(i,j) o nível de cinzento em (i,j) e δ um limiar pré-definido [8].

Durante a avaliação da região vizinha, cada agente apresentará um determinado comportamento. Esse comportamento é accionado pelo estímulo externo presente no ambiente que o rodeia. Quando um agente detecta um pixel de contorno, é colocado um marcador nesse pixel. Sendo λ=[u,v] uma gama aceitável, onde u ≤ v, o agente coloca um marcador no pixel p se a avaliação da distribuição da densidade em p cair dentro da gama especificada por λ.

De acordo com a distribuição da densidade de um dado agente numa dada região o agente adoptará um dos seguintes comportamentos:

Difusão: Sendo λ =[u,v] um intervalo aceitável para a contagem de pixels usando (1), o agente mover-se-á para as localizações adjacentes cada vez que a distribuição de densidade cair fora do intervalo λ, i.e., $D^{k}_{I(i,j)} \notin \lambda$. A direcção da difusão será escolhida aleatoriamente na primeira vez que o agente se deslocar. A partir do momento em que uma direcção é escolhida para um agente, esse agente deslocar-se-á sempre nessa direcção (Figura 2).

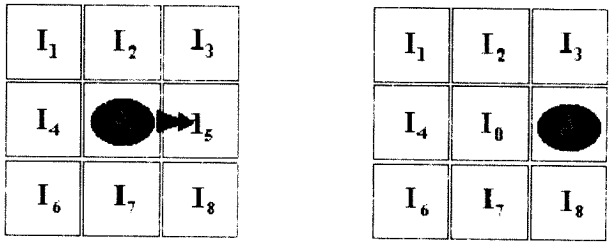


Figura 2 - Caso $D^{k}_{I(i,j)} \notin \lambda$ então o agente deslocar-se-á numa direcção determinada no primeiro movimento.

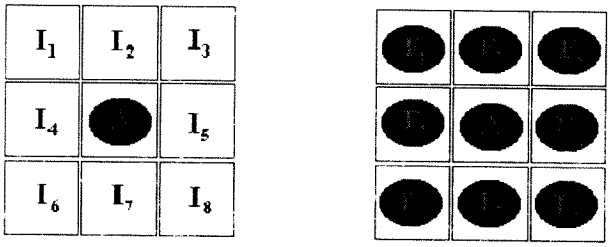


Figura 3 - Caso $D^{k}_{I(i,j)} \in \lambda$ então o agente produzirá agentes filho na sua zona de vizinhança.

Auto-reprodução: Se um agente detecta um pixel de contorno p ($D^{k}_{I(i,j)} \in \lambda$), produzirá um número finito de agentes filho dentro da zona de vizinhança de raio k. Neste trabalho foi por nós considerado que sempre que um agente detecta um pixel de contorno seriam produzidos por esse agente oito agentes filho (Fig. 3).

Morte: Quando um agente chega ao fim do seu tempo de vida, não pode proceder a mais nenhuma operação e é apagado do ambiente. Assim a detecção do contorno continuará a ser efectuada pelos seus descendentes, que por sua vez irão dar lugar a uma nova prole caso detectem o contorno. Caso o agente se desloque para uma zona onde não haja contorno, esse agente morrerá sem dar origem a uma nova geração.

2. Algoritmo

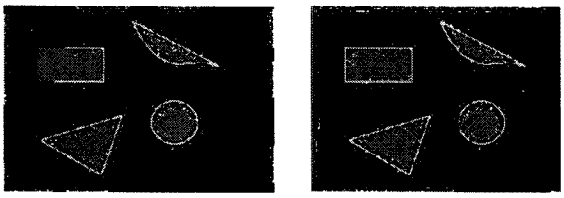
De seguida é apresentado o algoritmo completo utilizado para a detecção dos contornos dos lábios.

```
Distribuir aleatoriamente n agentes pela imagem,
inicializando o seu tempo de vida {φ_n^0}.
While ((NumAgentesVivos ≠ 0) and
      ((NumOperaçõesEfectuadas ≤ VAL_MAX))
   For  i = 0  to NumAgentesVivos
      If  ( D^k_{I(i,j)}(φ_i) ∈ λ)  then
         Gerar 8 agentes filhos na vizinhança.
         NumAgentesVivos=NumAgentesFilho + 8.
      Else
         If ( Primeiro movimento do agente )
               Calcular  direcção do movimento
         End If
         Mover agente φ_i
      End If
      If ( IdadeDoAgente = MAX_IDADE ) then
         Matar agente φ_i
         NumAgentesViVos--
      Else
         IdadeDoAgente++
      End If
   Next i
End While
```

3. Resultados obtidos

Utilizando o algoritmo atrás descrito, começámos por aplicá-lo numa imagem de teste bastante simples com vista a analisar quais as suas verdadeiras capacidades na detecção de contornos, por mais simples que estes fossem. A evolução do algoritmo ao longo das diferentes operações está descrita nas imagens apresentadas nas Figuras 4 e 5.

Os resultados foram obtidos utilizando 500 agentes iniciais, δ = 20 e λ = [1,7]. As linhas mais claras representam os agentes que se encontram sobre pontos de



n=25 n=70

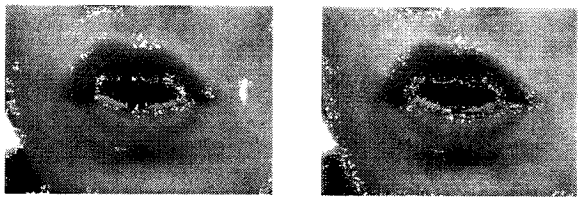
Figura 4 - Duas etapas na detecção dos contornos utilizando o algoritmo genético.

contorno, enquanto que os pontos claros dispersos pela imagem representam agentes que não estão mortos, mas que ainda não encontraram pontos de contorno.

Como se pode ver na Figura 4, verifica-se que os contornos presentes na imagem de teste são correctamente detectados (n=70). Os agentes, ao encontrarem pontos de contorno, marcam esses pontos e começam a formar colónias nessa zona, colónias essas que evoluem em torno das zonas de contorno. Os agentes que não encontram zonas óptimas de estabelecimento acabam por morrer, desaparecendo da imagem de teste.

Embora os resultados obtidos sejam satisfatórios, é importante salientar que a imagem de teste utilizada é bastante contrastada, facilitando a detecção dos contornos.

Como o objectivo deste projecto é detectar os contornos internos dos lábios, procedemos de seguida à detecção dos contornos em imagens faciais. Os resultados obtidos estão expostos na Figura 5.



n=8 n=25

Figura 5 – Aplicação do algoritmo genético às imagens faciais.

B. Segmentação baseada na Análise do Histograma

Ao contrário do algoritmo apresentado atrás, o algoritmo baseado no histograma vai isolar determinadas características da imagem, com vista a detectar os parâmetros dos lábios. Uma diferença importante entre os dois algoritmos reside no facto do algoritmo do histograma necessitar de processar toda a imagem.

1. Cálculo do histograma de uma imagem

O histograma de uma imagem digital a preto e branco é uma função discreta

$$p(r_k) = \frac{n_k}{n}$$

onde r_k representa o nível de cinzento k, n_k representa o número de pixels da imagem com esse nível de cinzento e n o número total de pixels na imagem.

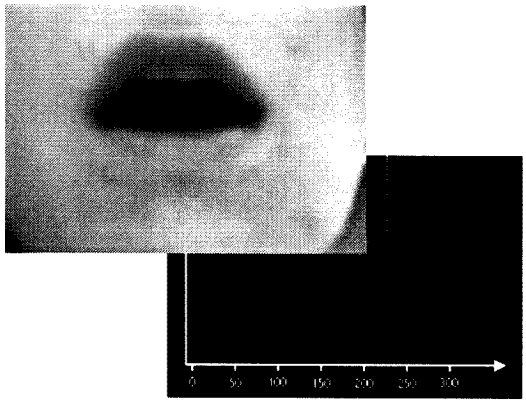


Figura 6 - Imagem típica utilizada e respectivo histograma.

Desta forma, $p(r_k)$ dá-nos uma estimativa da probabilidade de ocorrência do nível de cinzento r_k. Um gráfico desta função para todos os valores de k fornece uma descrição global do conteúdo da imagem.

Dependendo das características da imagem, o histograma pode apresentar várias formas.

Calculando o histograma para o nosso tipo de imagem, obtivemos os resultados que se apresentam na Figura 6.

Pela observação do histograma, constata-se que há um pico no intervalo de 0 a 70 que nos representa os níveis de cinzento mais escuros. Na imagem, os níveis mais escuros representam o interior da boca. Utilizando estes dados, tentámos isolar a zona interna da boca, usando para este fim o algoritmo a seguir apresentado.

2. Algoritmo utilizado

Utilizando os conhecimentos adquiridos pela análise do histograma, desenvolvemos um algoritmo capaz de segmentar o interior da boca, rejeitando tudo o resto.

De seguida é apresentado o algoritmo completo utilizado para a detecção dos contornos dos lábios :

```
For  x = 0 to DimX
 For y = 0 to DimY
   If ValMinCinzento<Imagem(x,y)< ValMaxCinzento
   Then Marcar ponto como PontoInternoDaBoca
 Next y
Next x
```

3. Resultados obtidos

Aplicando o algoritmo acima descrito sobre imagens faciais, os resultados obtidos foram semelhantes aos apresentados na Figura 7.

Figura 7 - Aplicação do algoritmo do histograma.

De modo a só se detectar o interior da boca, foi necessário isolar parte da imagem, visto que se não houvesse isolamento, também as fossas nasais seriam detectadas.

IV. CÁLCULO DOS PARÂMETROS DOS LÁBIOS

A. O problema da escala da imagem

Um dos primeiros problemas que se nos depara quando tentamos calcular os parâmetros dos lábios prende-se com o facto de durante a aquisição das imagens o indivíduo filmado poder estar mais afastado ou mais perto da câmara. Para tal é necessário ter um método que nos permita calcular a escala das distâncias.

O método por nós utilizado consiste em medir a largura da boca do indivíduo em repouso (em pixels) e, tendo como base a distância real em centímetros, calcular um factor que permita saber a quantos pixels corresponde um milímetro. Este factor é chamado de factor de escala e será mencionado posteriormente aquando da análise da aplicação ***Magic Lips***.

A largura da boca é calculada utilizando para tal o método do histograma visto anteriormente. No entanto, o limite de cinzento máximo aqui utilizado passa de 70 para 100. Este cálculo deve ser efectuado antes de iniciar qualquer processamento.

B. Contorno do algoritmo genético

Após a informação fornecida pelo algoritmo genético, torna-se necessário processar a imagem por forma a obter desta a informação acerca da altura, largura e área da zona interna da boca. A informação processada pelo algoritmo genético é passada através de uma matriz do tamanho da imagem processada, matriz esta que contém os pontos de contorno.

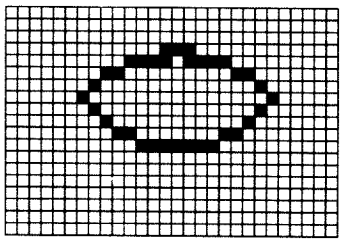


Figura 8 - Forma matricial como o algoritmo genético representa o contorno.

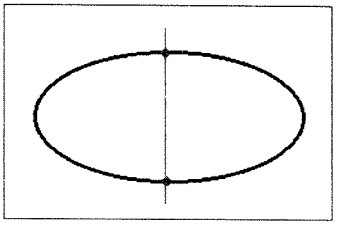


Figura 9 - Cálculo da altura da boca através de pontos extremos do contorno.

Para calcular a altura da boca é necessário calcular as coordenadas dos pontos de contorno que se afastam o mais possível na vertical. Considerando que a boca se encontra centrada na imagem, e considerando o ponto central da imagem, ao deslocar esse ponto na vertical (tanto para cima como para baixo), esse ponto irá encontrar os pontos extremos do contorno na vertical. A altura da boca será o módulo da diferença entre os pontos encontrados.

V. A APLICAÇÃO MAGIC LIPS

Foi desenvolvida, usando Visual Basic, uma aplicação integrando as facilidades de manipulação de filmes e os algoritmos de extracção de parâmetros dos lábios.

Na Figura 10 apresenta-se a interface da aplicação, denominada ***Magic Lips***.

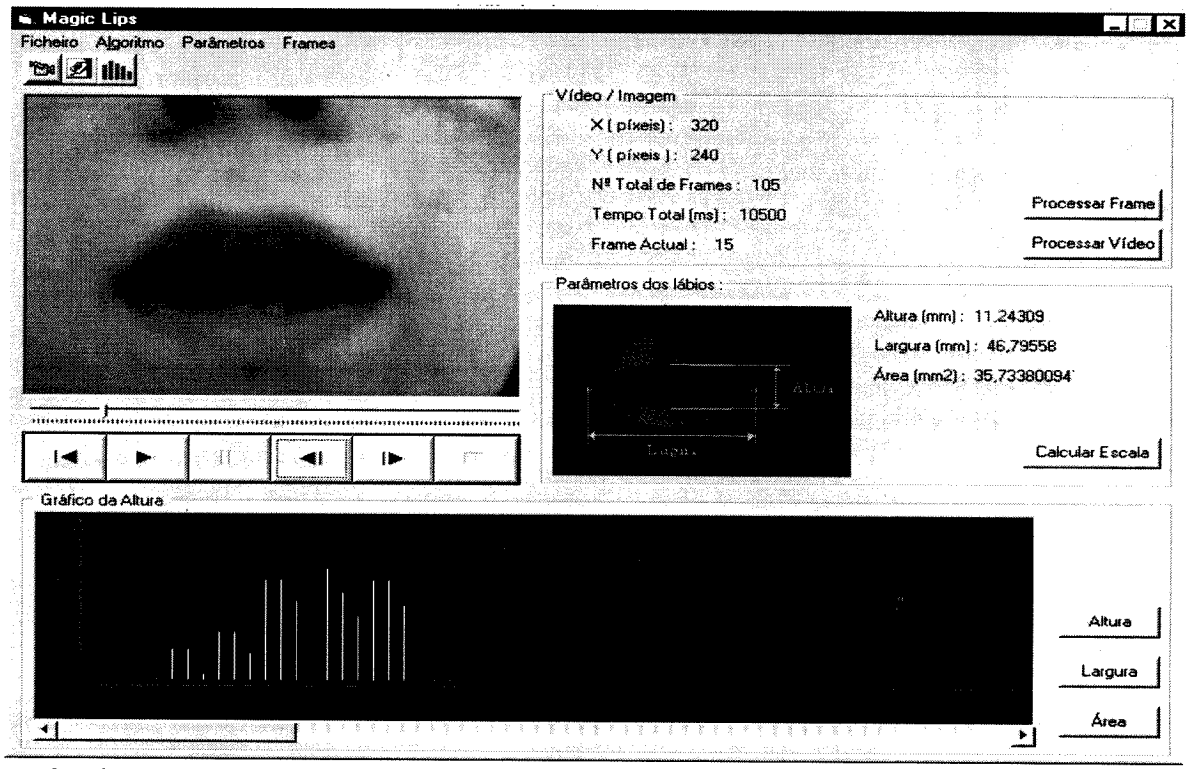


Figura 10 - Interface da aplicação ***Magic Lips***.

A interface da aplicação é composta por diferentes áreas, cada qual com uma função específica:

- menus para: operações com ficheiros, escolha do algoritmo, ajuste dos parâmetros dos algoritmos e selecção da parte de um vídeo que o utilizador pretende analisar. Existem também botões para acesso rápido a algumas das opções anteriores.
- zona de visualização das imagens. Esta zona permite manipular as imagens. Para tal existe uma zona de botões destinada a efectuar operações sobre o vídeo.
- zona de visualização dos parâmetros de vídeo/imagem. Permite o processamento de uma *frame* ou de um vídeo.
- A zona de parâmetros dos lábios. Após o processamento de cada *frame*, os parâmetros são disponibilizados nesta zona.
- A zona de visualização dos gráficos. Permite visualizar a variação dos parâmetros dos lábios ao longo do tempo. Três botões permitem comutar entre os gráficos da altura, da largura e da área.

VI. APLICAÇÃO AO ESTUDO DE VOGAIS

A. Estudo de vogais isoladas

O objectivo deste trabalho reside em extrair dados sobre o formato dos lábios nos sons da língua portuguesa, incidindo nas vogais. Analisámos vídeos onde um sujeito produzia os sons referentes a cinco vogais da língua portuguesa ([a], [E], [i], [ ], [u])[1]. A vogal [E] corresponde ao som da palavra **pé** e a vogal [ ] ao som utilizado na palavra **pó**. As outras três aparecem em **pá, pi, pu**lo.

Os valores obtidos estão expostos nas tabelas seguintes:

Método do histograma

Vogal	Altura (mm)	Largura (mm)	Área (mm^2)
[a]	7.98	38.76	286.67
[E]	4.49	34.23	150.64
[i]	2.46	30.65	87.25
[ ]	7.65	34.87	240.56
[u]	3.73	21.09	62.59

Algoritmo genético

Vogal	Altura (mm)	Largura (mm)	Área (mm^2)
[a]	10.54	49.02	405.92
[E]	7.02	49.67	273.66
[i]	4.65	53.92	196.94
[ ]	10.77	55.28	467.77
[u]	1.87	57.94	85.04

Os vídeos utilizados foram comuns aos dois métodos, no entanto existem discrepâncias entre alguns valores

obtidos, nomeadamente nas larguras (com óbvias repercussões nos valores das áreas). Esta ocorrência deu-se devido às condições de luz verificadas durante a filmagem dos vídeos. A câmara utilizada era muito sensível à quantidade de luz, sendo muito difícil obter imagens com boa qualidade. Desta forma resolvemos utilizar apenas os valores obtidos com o método do histograma, uma vez que este método veio a revelar-se menos sensível à qualidade das imagens.

Os resultados obtidos estão qualitativamente de acordo com a descrição, em termos do posicionamento dos lábios, que é geralmente feita das vogais do Português [2]. As vogais [a] e [ ] são as que apresentam maior abertura (altura) e maiores áreas de abertura dos lábios. A vogal [u] devido à configuração arredondada dos lábios e consequente protrusão (movimento para a frente) apresenta uma área reduzida e também, pelo menos segundo o método do histograma, largura da abertura dos lábios inferior às outras vogais.

B. Estudo de vogais integradas em palavras

O estudo efectuado na alínea anterior visava as vogais isoladas, não havendo desta forma ligação entre estas e outros sons da língua portuguesa. Neste trabalho procurou-se identificar os parâmetros dessas mesmas vogais num contexto ligeiramente diferente e verificar se existem diferenças significativas. Para tal utilizámos a palavra *pato,* palavra esta que tem a uma expressão clara das vogais [a] e [u]. A figura seguinte mostra as imagens retiradas do vídeo e referem-se ao momento em que as vogais são pronunciadas. Para tal foi utilizado o algoritmo do histograma, visto este detectar com exactidão os pontos extremos das áreas em questão.

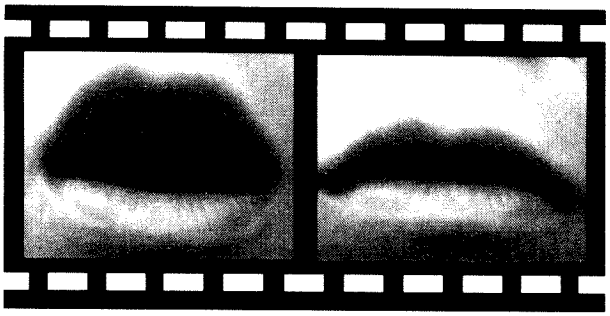


Figura 4 - Imagens relativas às vogais analisadas no vídeo da palavra *pato*, à esquerda o [a], à direita o [u].

Os resultados obtidos, semelhantes aos obtidos para estas duas vogais quando proferidas isoladamente, são apresentados na tabela seguinte.

Vogal	Altura	Largura	Área
[a]	9.721	39.40	369.22
[u]	1.790	23.79	56.41

VII. CONCLUSÕES

A segmentação baseada em algoritmos genéticos revelou adaptar-se bem às características das imagens em questão. É preciso, no entanto, mais trabalho para resolver o problema da determinação da largura de abertura dos lábios. As grandes vantagens deste algoritmo são, principalmente, não ser necessário processar toda a imagem e poder utilizar-se informação das *frames* anteriores. Estas características são muito interessantes para a aplicação pretendida. Refira-se que este algoritmo também permite a obtenção de informação do contorno exterior dos lábios e mesmo do rosto, informação essa necessária para sistemas de *visual speech.*

O algoritmo do histograma, embora conceptualmente bastante mais simples do que o algoritmo genético, revelou-se bastante rápido e eficiente, tendo produzido resultados muito bons.

Pudemos constatar que a extracção de parâmetros descrevendo o formato dos lábios não é uma tarefa fácil. No entanto, sendo uma área relativamente recente, foi gratificante concluir este trabalho com alguns dos objectivos atingidos. Por outro lado, embora o material utilizado fosse satisfatório, acreditamos que a utilização de imagens de melhor qualidade deverá contribuir para uma melhoria dos resultados.

Como trabalho futuro, identificamos as seguintes questões que poderão ser abordadas ou aprofundadas: resolução dos problemas detectados no algoritmo genético; utilização do algoritmo genético na obtenção de outros parâmetros da face; efectuar um estudo exaustivo de todas as vogais e consoantes existentes na língua portuguesa e utilização dos dados no desenvolvimento de modelo tridimensional dos lábios e/ou da face.

REFERÊNCIAS

[1] A. Andrade e M. C. Viana, "Fonética", *em Introdução à Linguística Geral e Portuguesa*, I. H. Faria et al., Caminho, 1996.

[2] J. M. Barbosa, *Introdução ao Estudo da Fonologia e Morfologia do Português*, Almedina, 1994.

[3] D. Beasley, D. R. Bull e R. R. Martin, "An Overview of Genetic Algorithms", *University Computing*, 1993.

[4] F. Lavaghetto, S.Lepsoy, C. Braccini e S. Curinga , "Lip Motion Modeling and Speech Driven Estimation", Proc. ICASSP, 1997.

[5] B. Le Goff, T. Guiard Marigny e C. Benoît, "Analysis-Synthesis and Intelligibility of a Talking Face", *Em Progress in Speech Synthesis*, J. van Santen, et. al. (Editores),, Springer Verlag, 1997.

[6] R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company, 1992.

[7] T. Guiard-Marigny, Ali Adjoudani e C. Benoît , "3D Models of the Lips and Jaws for Visual Synthesis", *Em Progress in Speech Synthesis*, J. P. H. Van Santen, et. Al (Editors), 1997.

[8] J. Liu, Y. Y. Tang e Y. C. Cao, "An Evolutionary Autonomous Agents Approach to Image Feature Extraction" , *IEEE Trans. on Evolutionary Computation*, Vol.1, Nº 2, Julho 1997.

[9] M. R. D. Martins. *Ouvir Falar Introdução à Fonética do Português*. Caminho, 1988.

[10] W. K. Pratt, *Digital image processing*, Wiley-Interscience, 1991.

[11] D. Whitley, *A Genetic Algorithm Tutorial*, Colorado State University, 1993.

Sistema de Controlo em Tempo Real da Qualidade da Água para Consumo Humano

Osvaldo Rocha Pacheco, Luís Pedro Antunes, Rui Ferreira*
*DECQ - Serviços Municipalizados de Aveiro
*DET - Universidade de Aveiro, ** DECQ - Serviços Municipalizados de Aveiro

Resumo **- Neste artigo apresentamos um sistema, que permitiu dotar a Divisão de Estudos e Controlo de Qualidade dos Serviços Municipalizados de Aveiro de uma rede de monitorização de parâmetros físicos e químicos que caracterizam sumariamente a qualidade da água captada para produção de água para consumo humano, de forma a complementar o sistema de controlo de qualidade em vigor. Pretende-se assim, com este sistema, controlar locais sensíveis da rede hídrica e parâmetros críticos nas captações de água. A informação recolhida pelo sistema é disponibilizada através da *Internet* e dos meios de comunicação social. Usando esta informação, pretende-se contribuir para a educação ambiental do cidadão, sensibilizando-o e estimulando-o para uma maior participação na protecção do ambiente.**

O sistema a apresentar é composto por uma rede de estações remotas responsáveis pela aquisição, análise e transmissão em contínuo dos parâmetros físicos e químicos de caracterização da qualidade da água, para uma estação central de recolha e processamento de dados e de gestão da qualidade da Água. Na estação central podem-se distinguir quatro módulos distintos: Módulo de Monitorização e Controlo em tempo real; Módulo de Gestão da Qualidade; Módulo de Comunicação e Módulo de Configuração do Sistema. Neste trabalho serão ainda apresentadas as motivações de suporte das decisões tomadas quanto aos pontos de instalação das estações remotas e as áreas onde pensamos que este projecto poderá proporcionar intervenção de relevo.

I. INTRODUÇÃO

A gestão racional dos recursos hídricos obriga a uma abordagem multidisciplinar por forma a garantir a sua protecção, conservação, valorização e monitorização. Assim, a prevenção e minimização dos riscos de poluição e de degradação dos meios hídricos, pode e deve ser feita recorrendo às melhores práticas de gestão e a técnicas ambientalmente adequadas [1]. Com este projecto pretendemos produzir algumas ferramentas de suporte para a obtenção dos objectivos enunciados. O sistema consiste na instalação de uma rede de estações remotas de análise física e química em contínuo de parâmetros de qualidade da água, colocados estrategicamente de modo a transmitir a uma estação central e posteriormente à população, informação que caracteriza sumariamente a qualidade da água captada para produção de água para consumo humano, de forma a complementar o sistema de controlo de qualidade em vigor e permitir em tempo real, a detecção e a monitorização de eventuais anomalias. A estação central tem por funções fazer a recolha dos dados, disponibilizar ferramentas de gestão da qualidade, accionar automaticamente alarmes e preparar a informação para difusão pública, via Internet e/ou pela comunicação social. A implementação deste sistema decorreu do projecto SIMOQUA (Sistema de Monitorização da Qualidade do Ambiente), foi realizado pelos Serviços Municipalizados de Aveiro e pela Universidade de Aveiro em consórcio e foi parcialmente financiada pelo programa Aveiro - Cidade Digital [2].

II. ENQUADRAMENTO

O sistema público de abastecimento de água do Município de Aveiro, tem uma extensão aproximada de 650 Km, abastecendo cerca de 70 000 habitantes e representando uma taxa de cobertura de 98%. Este sistema é constituído por 6 subsistemas, dos quais cinco se encontram interligados e é abastecido por 20 origens de água, das quais 17 são furos subterrâneos (que captam no aquífero Cretácico), uma é um aproveitamento de duas minas e as últimas derivam do fornecimento de água através da Associação de Municípios do Carvoeiro (AMC) que possui a sua captação no rio Vouga.

CAPTAÇÃO	CAUDAL MÉDIO DIÁRIO m3/dia
AMC	10 166
VALE DAS MAIAS	2 160
JK10 (furo do Viso)	2 328
TOTAL	**14 654**

Tabela 1 - Caudal médio diário por captação

Pode-se dizer que três das captações são as que mais significativamente contribuem para a produção de água para consumo dos Aveirenses, pois a sua produtividade média (tabela 1), permite abastecer o Concelho durante

todo o ano, com excepção de alturas pontuais de elevado consumo (meses de Verão) ou avarias no sistema de distribuição. Assim, naturalmente estas origens, por serem as de maior impacto na qualidade da água distribuída, foram as escolhidas para este projecto.

Considerando a dimensão e complexidade do sistema verifica-se que, para cumprimento da legislação (Decreto Lei n.º 74/90 de 7 de Março e agora o Decreto Lei n.º 236/98 de 1 de Agosto), é gerado anualmente um grande manancial de resultados analíticos que genericamente não são do conhecimento da população abastecida. Utilizando o longo historial analítico existente e conhecendo os riscos ambientais associados a cada uma das captações escolhidas, foram seleccionados os parâmetros a integrar no controlo de qualidade em tempo real, de acordo com as tecnologias disponíveis actualmente. Assim, seleccionaram-se respectivamente os seguintes locais e parâmetros:

AMC (reservatório do Silval, onde se inicia a distribuição em baixa) : ***PH, Cloro residual, Nitratos e Condutividade.***

Esta água apresenta-se com PH ligeiramente básico, consequência do tratamento efectuado no sistema em "alta" pouco mineralizada, com alcalinidade média, pouco dura e apresentando alguns nitratos.

VALE DAS MAIAS (no reservatório da Cidade onde se inicia a distribuição em baixa): ***PH, Cloro residual, Ferro e Nitratos.***

As principais características desta água são o PH ligeiramente ácido, (embora corrigido na captação, através da adição de leite de cal) com algum ferro em excesso, apresentando alguns nitratos embora com teores normalmente inferiores ao VMR (valor máximo recomendado) definido no decreto-lei 236/98 de 1 de Agosto.

JK10 (na própria captação): ***Ferro, Turvação, Condutividade, Cloro residual e Cloretos.***

Esta água possui características alcalinas e teores de Ferro da ordem do VMA (valor máximo admissível) ou superior. Refira-se que aos restantes parâmetros indicados no decreto-lei n.º 236/98 de 1 de Agosto estão de acordo com os respectivos valores máximos recomendáveis e as suas tendências de evolução não levantam preocupações de maior.

III. OBJECTIVOS

No âmbito deste projecto, foram equacionados os seguintes objectivos:

- Desenvolver mecanismos de controlo e de gestão em tempo real da qualidade da água para consumo humano.
- Ter acesso e dar a conhecer à população de Aveiro e entidades interessadas, utilizando as Tecnologias da Informação e da Comunicação, informação contínua e actualizada da qualidade da água das captações de água para consumo humano.
- Determinar locais sensíveis da rede hídrica superficial e de parâmetros críticos nas captações de água, que possam influenciar o uso destas para produção de água para consumo humano.
- Contribuir para a educação ambiental do cidadão, usando a informação disponibilizada pelo sistema.

IV. DESCRIÇÃO DO SISTEMA

O projecto compreende a integração de um sistema de supervisão desenvolvido no Departamento de Electrónica e Telecomunicações com equipamentos de análise física e química de parâmetros da qualidade da água comerciais (figura 1). Os parâmetros são recolhidos em estações remotas que compreendem a instrumentação necessária desenvolvida para o efeito. O sistema de supervisão é baseado num modelo desenvolvido e testado para a gestão de uma rede de estações meteorológicas (projecto SiMoVA) [3]). A informação disponibilizada pelas estações remotas é guardada numa base de dados que serve de suporte à estação central. Esta, incorpora as funcionalidades de um sistema de supervisão e controlo tipo SCADA e um módulo para a análise e gestão da qualidade da água.

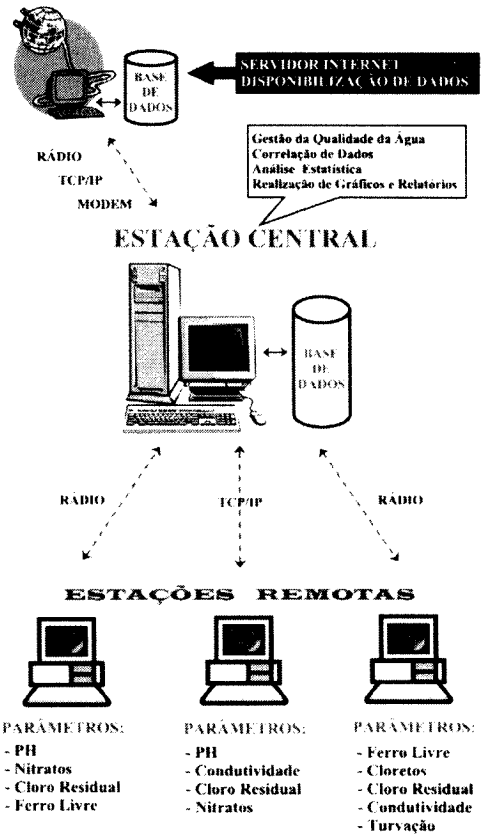


Figura 1 - Arquitectura geral do sistema.

Estações Remotas

Como já foi referido, o sistema é constituído por três estações remotas e uma estação central. Em cada uma das estações remotas são recolhidos os parâmetros físicos e químicos que caracterizam sumariamente a qualidade da

água, à frequência programada, sendo de seguida enviados para a estação central, onde se efectuará todo o processamento dos dados recebidos. Os sistemas analíticos são baseados na técnica FIA (*Flow Injection Analysis*), análise por injecção em fluxo, pelo que são geralmente dotados de uma enorme capacidade de adaptação a diversos propósitos analíticos. Os ritmos analíticos conseguidos, devido à técnica utilizada, são habitualmente elevados, sempre superiores aos conseguidos pelos métodos discretos e muitas vezes maiores que os conseguidos por outras propostas de automatização [4].

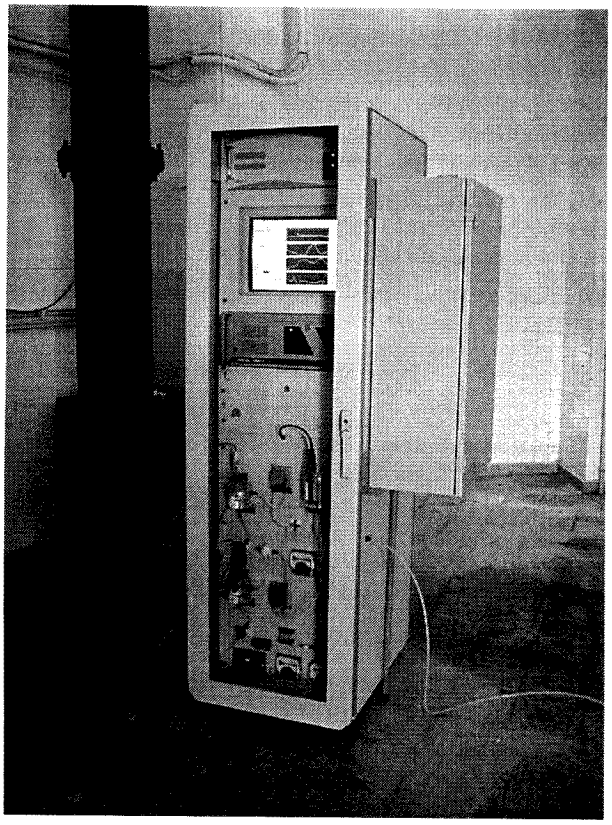


Figura 2 – Vista da estação remota do Silval.

Para concentrar a informação proveniente de cada um dos analisadores, foram implementadas duas soluções. Assim, em função das condições de instalação, pode-se recorrer a um controlador lógico programável (*PLC*) ou a um PC industrial (figura 2). A interligação de cada um dos analisadores com o concentrador é efectuada em corrente (4-20mA), RS232C ou por RS485. O software presente neste concentrador recolhe os dados de cada analisador, disponibiliza-os num formato que seja susceptível de ser transmitido e envia-os para a estação central com uma periodicidade programável pelo utilizador, local ou remotamente, limitada superiormente pelo tempo necessário para a realização de uma análise. A estação remota permite ainda o controlo da calibração dos analisadores que possam ser calibrados automaticamente, bem como a abertura e fecho das válvulas dos analisadores. Este software é desenvolvido em linguagem '*Ladder*', caso se esteja a utilizar a solução baseada no *PLC*, ou em *LABVIEW 5.01*, caso se esteja a utilizar o PC industrial. Neste último caso, para além de controlar todo o sistema, este software permite visualizar a variação temporal dos parâmetros em análise nas últimas 24 horas, bem como a verificação das principais variáveis de funcionamento.

Estação Central

Na estação central encontra-se um servidor a correr Windows NT, baseado numa plataforma Intel. O software presente neste servidor, desenvolvido em LABVIEW 5.01, controla o processo de recolha, processamento, armazenamento e disponibilização da informação recebida. Na estação central podem-se distinguir quatro módulos distintos.

Módulo de Configuração do Sistema

O Módulo de Configuração do Sistema, permite activar e configurar as diferentes estações remotas, quer quanto ao número e tipo de instrumentação usada, quer quanto aos alarmes que lhes estão associados. Neste módulo é ainda possível a definição do tipo de comunicação a estabelecer, a configuração da interligação com o servidor *Internet* para a divulgação da informação. Finalmente, também é possível definir e gerir os tipos diferentes de utilizadores e as respectivas permissões associadas (figura 3).

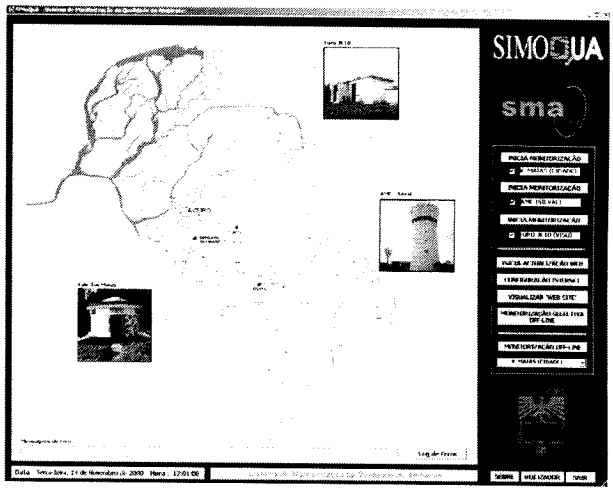


Figura 3 – Interface do módulo de sistema.

Módulo de Monitorização e Controlo em tempo real

Este módulo basicamente disponibiliza as funcionalidades de um software *SCADA*, permitindo a visualização dos últimos parâmetros recolhidos bem como dos alarmes que lhes estão associados (figura 4). Uma situação de alarme pode ser difundida localmente ou remotamente por correio electrónico. Alarmes que indiciam situações de alguma gravidade implicam o envio de um correio electrónico aos gestores da Divisão de Controle de Qualidade. Do ponto de vista do controlo do sistema, neste módulo é possível a desactivação de uma

estação ou de um qualquer analisador ou forçar numa estação remota, a repetição de uma dada análise. Os dados recolhidos são armazenados numa base de dados, que os disponibiliza ao módulo de gestão da qualidade. Uma versão reduzida deste módulo está presente nas estações remotas e permite a visualização local dos parâmetros.

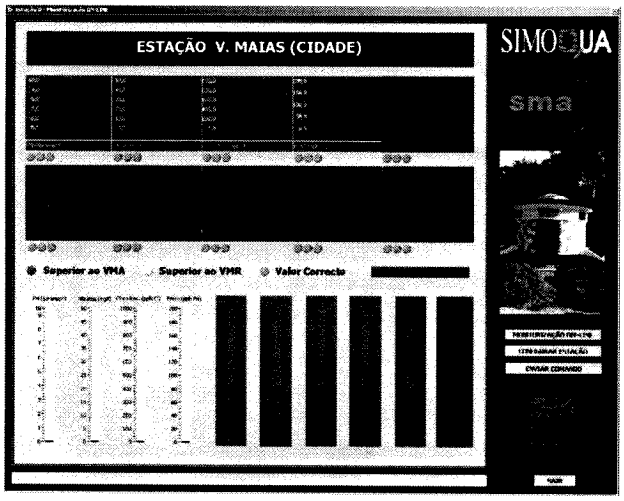


Figura 4 – Visualizador disponível para cada estação remota instalada para a visualização em tempo real dos parâmetros recolhidos.

Módulo de Gestão da Qualidade

O Módulo de Gestão de Qualidade disponibiliza funcionalidades que se pode agrupar da seguinte forma: visualização e análise de dados; realização de cálculos e análise estatística; construção de gráficos e relatórios automaticamente. Estas funcionalidades são suportadas pela base de dados que é actualizada pelo módulo de monitorização e controlo em tempo real.

A visualização e análise de dados permite a construção de gráficos seleccionando o intervalo de tempo em análise e os parâmetros independentemente da estação remota de recolha. Assim, definindo os intervalos temporais, seleccionando os dados para visualização é disponibilizado um sem número de configurações, quaisquer que sejam as origens (estações e/ou parâmetros) dos dados (fig. 5).

Este módulo disponibiliza também ferramentas para a realização de um conjunto significativo de cálculos, tais como médias flutuantes, anuais, mensais, diárias e horárias, bem como a determinação de máximos e mínimos, num intervalo escolhido pelo utilizador. A inserção de novas ferramentas de análise, devido à estrutura do *software,* podem ser facilmente incorporadas.

A construção de relatórios, partindo de relatórios tipo, a construção de gráficos para impressão e a exportação de dados para, por exemplo, o *Microsoft Excel* são outras das facilidades disponíveis no módulo de gestão da qualidade.

A informação armazenada pela Estação Central, depois de tratada é enviada para uma base de dados associada a um servidor *Internet*, garantindo a disponibilização da informação necessária para a construção do *site* para consulta pública. O cidadão pode assim analisar a água que nesse instante está a ser usada para produção de água para consumo humano, em função da sua origem ou do seu local de consumo, bem como, nestas duas perspectivas, pode analisar o histórico armazenado.

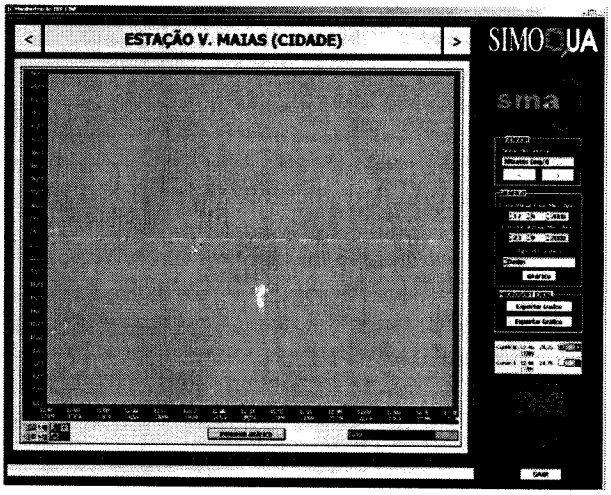


Figura 5 – Visualizador de parâmetro em função da estação seleccionada.

Módulo de Comunicações

O Módulo de Comunicações assegura a comunicação da estação central com as estações remotas e com o servidor Internet. Foram implementados mecanismos de detecção, correção e recuperação de erros. O tipo de comunicação depende da interface com que dotamos o sistema e que fundamentalmente é uma de duas: RS232C com ou sem modem ou TCP/IP. Neste módulo é também possível configurar os diferentes parâmetros da comunicação para as diferentes estações remotas.

Para a comunicação entre as estações remotas e a estação central, foram equacionadas e testadas várias soluções, que passavam pela utilização do GSM, linhas RDIS, linha analógica, rádio Trunking ou TCP/IP. Duas das estações remotas estão ligadas por rádio Trunking. A terceira estação por estar dentro do mesmo perímetrro privado da estação central está interligada por TCP/IP. A estação central está ligada ao servidor *Internet* também por TCP/IP. A aquisição de rádios emissores - receptores e 'radio-modems' para cada estação (central e remotas) apresentam um custo elevado, ao que acresce uma mensalidade de baixo valor. Em compensação, as comunicações não apresentam nenhum custo adicional, sendo somente limitadas em tempo (cerca de 1,5 minutos). As grandes desvantagens desta solução são a baixa largura de banda disponível (1200bps) e a existência de interferências nas comunicações quando as condições climatéricas são adversas. Note-se que qualquer outro tipo de ligação pode a qualquer momento ser usado, pois o software de interface está presente na aplicação sendo apenas necessário a sua configuração.

Servidor Internet

A informação disponibilizada na Internet (**www.simoqua.pt**) serve três propósitos: informar a população da qualidade da água fornecida para consumo humano; dinamizar a sensibilização e formação para uma actuação correcta ao nível ambiental em particular na utilização e gestão individual da água e divulgar informação institucional ligada ao projecto. O utilizador pode visualizar os últimos dados recolhidos numa qualquer das estações remotas (fig. 6) ou analisar a evolução dos parâmetros ao longo do tempo recorrendo à informação guardada na base de dados (fig. 7). Assim, o utilizador pode definir o intervalo de tempo em análise, seleccionar o parâmetro ou um seu derivado (média hora, dia, etc), escolher um qualquer ponto de captação ou realizar o seu pedido em função da água que é fornecida na sua residência.

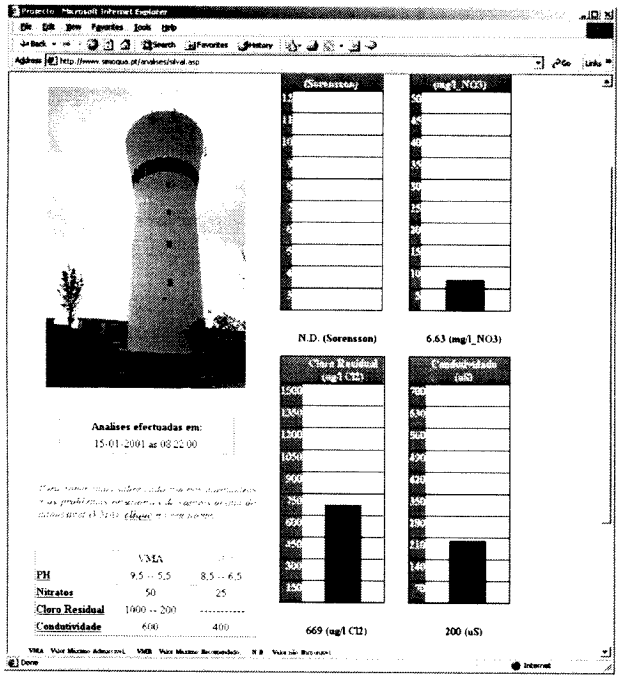


Figura 6 – Página WEB construída dinâmicamente apresentando as análises mais recentes da estação remota do Silval.

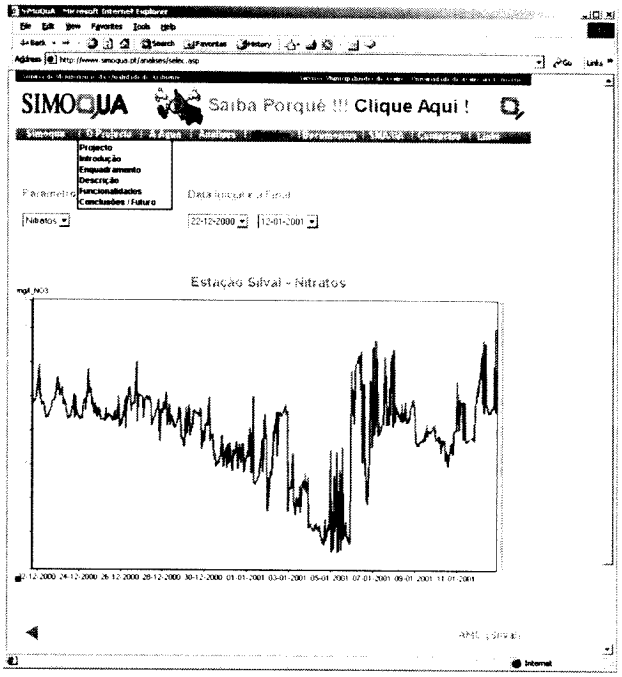


Figura 7 – Gráfico apresentando a evolução dos Nitratos no Silval para um período de tempo definido pelo utilizador.

V. CONCLUSÕES E TRABALHO FUTURO

O sistema de controle em tempo real dos principais parâmetros caracterizadores da qualidade da água trás vantagens significativas para o funcionamento do sistema de abastecimento público de água, dado que:

- Permite ter acesso e dar a conhecer à população de Aveiro e entidades interessadas, informação contínua e actualizada da qualidade da água das captações de água para consumo humano;
- Permite conhecer mais cedo os eventuais problemas ou contaminações da água nas origens, originando uma intervenção precoce para solucionar as causas destas situações anómalas e/ou ainda contactar as autoridades competentes para a tomada de medidas que visem a protecção dessas mesmas origens de água;
- Permite a tomada de medidas de optimização dos sistemas de tratamento de água existentes no sistema de abastecimento de água;
- Contribui para a melhoria da qualidade do serviço prestado, e ainda com a disponibilização desta informação à população, poder-se-á aumentar a sua sensibilização e formação para uma actuação correcta ao nível ambiental em particular na utilização e gestão individual da água.

A implementação deste projecto trouxe já significativas alterações nas metodologias usadas pela Divisão de Controlo de Qualidade da Água. A existência de informação em tempo real e o elevado número de amostras realizadas levou a que os responsáveis tenham uma melhor percepção dos problemas existentes e tenham tomado medidas para os minimizar. Esta realidade é o melhor resultado obtido com a implementação deste sistema pois dele resulta a produção de água de melhor

qualidade ao consumidor. Pensamos também que o projecto apresenta um carácter inovador e estratégico, uma vez que não existe no país uma rede de monitorização a esta escala e com estes objectivos, nomeadamente por disponibilizar ao cidadão a informação recolhida.

O consórcio formado no âmbito deste projecto, aproveitando as sinergias geradas no decorrer do Projecto SIMOQUA, partiu já para novos desafios, potenciando as valências das duas partes nas diferentes áreas. Assim dois novos projectos estão já em desenvolvimento:

O Sistema de Informação de Apoio ao Controlo da Qualidade da Água que permitirá a sistematização e a gestão de toda a informação que resulta dos procedimentos definidos por lei no controlo da qualidade da água; O Sistema de Apoio à Decisão no Controlo da Qualidade da Água que permitirá apoiar os técnicos nas decisões a tomar no âmbito da gestão da qualidade da água.

Atendendo ao conjunto de actividades ligadas à fruição da Ria, que decorrem em Aveiro, como por exemplo: turismo, pesca, actividades desportivas aquáticas (remo, vela), lazer, aquacultura e salicultura, pensamos ainda, que seria de todo interesse, alargar o âmbito deste projecto, tornando possível a análise da qualidade da água superficial dos pontos representativos dos canais urbanos da Ria de Aveiro. A gestão desta informação permitiria para além da melhoria em termos de qualidade de vida, a classificação dos recursos hídricos, tal como actualmente acontece com as praias de Bandeira Azul.

REFERÊNCIAS

[1] Plano Nacional da Água - *Site* do Instituto da Água (www.inag.pt). 1999.

[2] Dossier de candidatura do Projecto SIMOQUA ao Programa Aveiro - Cidade Digital. Rui Ferreira, Osvaldo Pacheco. 1998.

[3] L. Coimbra, R. Matos, J. P. E. Oliveira, O. R. Pacheco. Sistema de Monitorização e Actuação de Aplicação Ambiental. Anais da Engenharia Electrotécnica. Ano IV, nº 7. Maio de 1999.

[4] J. Catita. Analisadores Automáticos Baseados em Técnicas de Fluxo, para Controlo em Tempo Real de Parâmetros Caracterizadores da Qualidade da Água para Consumo Humano. Relatório Técnico da Paralab. Outubro. 2000.

Título: Síntese articulatória das vogais nasais do Português Europeu

Title: *Articulatory synthesis of European Portuguese Nasal Vowels*

Autor/*Author*: António Joaquim da Silva Teixeira

Orientador/*Advisor*: Francisco A. C. Vaz

Data Apresentação/*Acceptance Date*: 18/07/2000

Palavras Chave: Síntese articulatória, nasais, Português, dinâmica, interacção, testes perceptuais, qualidade

Key Words: *Articulatory synthesis, nasals, Portuguese, dynamics, interaction, perceptual tests, quality*

Doutoramento/*Ph.D.*

Resumo

A principal motivação para este trabalho foi a obtenção de síntese com qualidade próxima da natural. Um dos caminhos mais promissores para atingir esse objectivo é a utilização de síntese articulatória, técnica baseada na modelação directa do processo de produção humano. A síntese articulatória permite realizar simulações e, funcionando como um falante versátil, obter estímulos para a realização de testes perceptuais, sendo útil em estudos indispensáveis na obtenção de novos conhecimentos sobre os mecanismos de produção e percepção.

O objectivo principal deste trabalho foi o de estudar e aplicar as técnicas da síntese articulatória ao estudo e síntese do Português Europeu. Em vez de dispersar a nossa atenção por todos os tipos de sons da língua portuguesa, optamos por nos dedicar exclusivamente ao estudo das vogais nasais do Português, sons com especial relevância na língua de Camões, e em que outras técnicas de síntese têm limitações.

Foi desenvolvido um sintetizador articulatório especialmente vocacionado para a síntese de sons nasais, utilizando técnicas já conhecidas e introduzindo um novo modelo acústico para o tracto nasal e uma fonte glotal interactiva.

O sintetizador articulatório foi utilizado em estudos, tendo por objectivo melhorar o conhecimento acerca dos sons nasais, abordando dois assuntos: a interacção entre a fonte de excitação glotal e as cavidades acima da laringe, e o efeito da dinâmica dos articuladores.

Os resultados das experiências efectuadas mostram que a variação do velo, e mesmo de outros articuladores, influencia a percepção de nasalidade. Incluindo a forma como os articuladores variam, nos vários contextos em que as vogais nasais do Português aparecem, melhora a qualidade. Com base nos resultados obtidos, consideramos que as vogais nasais, do Português Europeu, devem ser vistas como ditongos, iniciando-se com radiação pelos lábios dominante e acabando em radiação nasal dominante, sendo a transição entre as duas configurações gradual.

As alterações nas características da onda de excitação glotal, devidas ao acoplamento do tracto nasal, são muito mais visíveis no domínio do tempo do que no domínio da frequência e este efeito não tem a mesma relevância para todas as vogais. Os resultados de testes perceptuais mostram que as alterações não são de fácil identificação, existindo, no entanto, uma tendência para o efeito ser mais facilmente detectável para a vogal elevada [ɪ̃], produzida com uma cavidade bucal de área bastante reduzida.

No seu estado actual, o sintetizador e os conhecimentos que obtivemos permitem produzir vogais nasais e palavras contendo sons nasais com qualidade entre o razoável e bom (níveis 3 a 4 na escala MOS).

Abstract

Main motivation for this work was the attainment of synthesis with natural quality. One of the promising paths to reach this objective is the use of articulatory synthesis, technique based on direct modellation of the human speech production process. Articulatory synthesis can be used in simulations, and as a versatile informant to produce stimuli for perceptual tests, indispensable in speech production and perception studies.

The main objective of this work was the application of articulatory synthesis techniques to the study and synthesis of European Portuguese. Instead of dispersing our attention with all the sound types of the Portuguese language, we decided to concentrate on the study of the Portuguese nasal vowels, sounds with special relevance in the language of Camões, and where other synthesis techniques have shown limitations.

An articulatory synthesizer, especially tailored for the synthesis of nasal sounds, was developed, using available techniques and introducing a new comprehensive nasal tract acoustic model and an interactive glottal source model.

The synthesizer was used in studies having as objective the improvement of the knowledge concerning Portuguese nasal vowels. Studies addressed two different subjects: source tract acoustic interaction, and the effect of the articulators' dynamics.

Experiments results showed that the variation of the velum and other articulators position, during nasal vowel production, influences perception of nasality. Including, in the synthesis process, information about velum and other articulators dynamics improves the quality. On the basis of the results obtained, for several contexts, we propose that European Portuguese nasal vowels can be seen as diphthongs, initiating themselves with dominant oral radiation and finishing with dominant nasal radiation, being the transition gradual.

Alterations in the glottal flux due to the extra coupling of the nasal tract are much more visible in time than in the frequency domain and the effect does not have the same relevance for all the vowels. Perceptual tests showed that the alterations are not easily perceived, existing a tendency of the effects being more detectable for the high vowels, like [ɪ̃], having a small area of the vocal tract regions between velum and the lips.

In its present state, the synthesizer, using the results from our studies, is capable of producing nasal vowels, and words with nasal sounds, with quality between Fair and Good (levels 3 to 4 in a MOS quality test).

Título: Reconstrução de Sinal e Codificação
Title: *Signal Reconstruction and Coding*
Autor/*Author*: José Manuel Neto Vieira
Orientador/*Advisor*: Paulo Jorge S. G. Ferreira
Data Apresentação/*Acceptance Date*: 11/02/2000
Palavras Chave: Reconstrução de sinal, correcção de erros, códigos de correcção de erros, Combinatória.
Key Words: *Signal reconstruction, Error correction, Error correction codes, Combinatory.*
Doutoramento/*Ph.D.*

Resumo

Considere-se um sinal passa-baixo com N amostras, t das quais foram posteriormente modificadas. Será possível detectar quais as amostras modificadas, e recuperar o seu valor original?A quase totalidade das técnicas conhecidas sobre reconstrução de sinal não resolve este problema, sendo necessário conhecer a posição das amostras erradas para que seja possível obter a amplitude correcta. No entanto, os códigos de correcção de erros do tipo BCH, entre outros, conseguem resolver este problema, ou seja, determinar a posição das amostras erradas e, numa segunda fase, corrigir a sua amplitude. Estes códigos são normalmente aplicados a sinais digitais recorrendo a "software" com aritmética num corpo finito necessitando, por esse motivo, de processadores dedicados para realizar as operações de codificação e descodificação de forma eficiente. No entanto, e como veremos ao longo deste trabalho, é possível utilizar estas técnicas no corpo dos números complexos, levantando-se no entanto uma série de questões novas como, por exemplo, a da estabilidade da reconstrução. As técnicas de reconstrução de sinal e os códigos de correcção de erros são usualmente encarados como disciplinas distintas com as suas técnicas próprias não existindo aparentemente qualquer relação entre as duas. No entanto, estas dificuldades resultam em grande medida da diferente aritmética, notação e linguagem utilizadas. Esta dissertação utiliza em simultâneo estas duas disciplinas, transportando técnicas e conceitos de uma área para outra, numa tentativa de enriquecer o conhecimento e compreensão de ambas.Neste trabalho estudamos várias técnicas para determinar a posição das amostras erradas num sinal discreto limitado em banda e de duração finita, sendo descrito como reconstrução não linear. Comparamos igualmente algumas técnicas de determinação da amplitude das amostras erradas, oriundas dos códigos de correcção de erros, com as que são utilizadas nos algoritmos de reconstrução de sinal. O problema da estabilidade da determinação da posição das amostras erradas é investigado, mostrando-se por exemplo que os efeitos da amplitude e posição dos erros na estabilidade podem ser separados.Um dos objectivos deste trabalho foi o de encontrar técnicas que permitissem o projecto de códigos de correcção de erros no corpo dos números complexos. A chave para a solução deste problema reside na combinatória dos padrões dos erros e da sua influência na estabilidade dos algoritmos de reconstrução.

Abstract

Consider a low-pass signal with N samples where the amplitude of t samples was modified. Is it possible to find a method to detect which samples violate the low-pass condition and reconstruct their original amplitude? Most of the known techniques of signal reconstruction don't solve this problem, being necessary to know the error positions in order to find their correct amplitudes. However, the BCH type error correction codes, for example, can solve this problem in two steps, first, they found the error positions, then, they found the error amplitudes. Those codes are usually implemented on a finite arithmetic field with dedicated processors, which implement the coding and decoding tasks on an efficient way. Nevertheless, and as we will see during this work, it is possible to use those techniques on the complex field, arising some new problems as the numerically stability of the reconstruction process. The signal reconstruction techniques and the error correction codes are usually faced as distinct disciplines, with their own techniques and results, and apparently with few aspects in common. These difficulties are the result of differences in the arithmetic notation and language used. This thesis deals with both disciplines transporting techniques and concepts from one area to another, trying to enrich the knowledge and comprehension of both. In this work we have studied several techniques to find the error positions in a band-limited and time-limited signal. Those techniques are described as non-linear signal reconstruction. We also compare some techniques for error amplitude correction imported from error correction codes with some signal reconstruction techniques. The stability of the problem of finding the error positions was also studied. We have achieved some interesting results as for example the separation of the influence of the error amplitude from the error position on the stability. One of the goals of this work was to find some techniques to design error correction codes on the complex field. The key to this problem is the error pattern combinatory and their influence on the reconstruction stability.

Título: Modelos, Métodos e Ferramentas para Implementação de Unidades de Controlo Virtuais.
***Title**: Models, Methods and Tools for Implementation of Virtual Control Units.*
Autor/*Author*: Arnaldo Silva Rodrigues de Oliveira
Orientador/*Advisor*: Valery Anatolevitch Sklyarov
Data Apresentação/*Acceptance Date*: 20/07/00
Palavras Chave: Unidades de Controlo, Dispositivos Lógicos Programáveis, Sistemas Reconfiguráveis, Síntese Lógica.
***Key Words**: Control Units, Field Programmable Logic Devices, Reconfigurable Systems, Logic Synthesis.*
Mestrado/*M.S.*

Resumo

As unidades de controlo são um dos tipos de circuitos digitais mais importantes, estando presentes numa grande diversidade de aplicações, desde simples controladores de semáforos até sistemas complexos de processamento de dados. A sua função é estabelecer a sequência de operações realizadas pelo sistema a que pertencem. Dependendo da aplicação, podem ser utilizadas isoladamente ou em conjunto com outros componentes, tais como unidades de execução, sensores e actuadores. Como um caso particular de circuitos sequenciais, são normalmente descritas por modelos orientados ao estado, dos quais a máquina de estados finitos é o exemplo mais conhecido. No entanto, com a crescente complexidade dos sistemas e consequentemente das suas unidades de controlo, este modelo deixou de ser adequado para realizar a sua especificação, uma vez que não suporta a descrição explícita de hierarquia e concorrência. Nesta dissertação são abordados alguns dos modelos e linguagens mais apropriadas para este fim, em particular a máquina de estados finitos hierárquica e/ou paralela, os esquemas de grafos hierárquicos e/ou paralelos e os *Statecharts*. O aparecimento na última década de dispositivos lógicos de elevada capacidade e programáveis pelo utilizador foi responsável por importantes alterações no projecto de sistemas digitais, principalmente ao nível do tempo, custo e flexibilidade de projecto. Além disso, os dispositivos programáveis dinamicamente, deram origem a uma nova classe de circuitos: os sistemas reconfiguráveis. Estes podem ser usados para construir sistemas computacionais modificáveis, pelo que permitem combinar as vantagens de uma solução programável com o elevado desempenho de uma implementação em hardware. Os sistemas reconfiguráveis podem também ser utilizados em aplicações onde a quantidade de recursos de hardware necessários para uma implementação integral seja elevada e onde nem todos os sub-sistemas sejam necessários em simultâneo, sendo portanto possível e até desejável uma implementação parcial em conjunto com a sua reconfiguração dinâmica. Neste caso, devido à analogia com os sistemas de memória virtual, os sistemas reconfiguráveis são também designados por sistemas de hardware virtual e os respectivos circuitos de controlo por unidades de controlo virtuais. Como as unidades de controlo são específicas de cada projecto e normalmente bastante irregulares, torna-se necessário o estabelecimento de algumas restrições de forma a simplificar o seu projecto e reconfiguração. Nesta dissertação é proposta uma arquitectura de unidades de controlo virtuais baseada numa estrutura predefinida, parametrizável e optimizada para o dispositivo de implementação utilizado, a FPGA XC6200 da Xilinx. Esta arquitectura em conjunto com os modelos e os dispositivos lógicos programáveis utilizados permite construir unidades de controlo complexas, flexíveis, extensíveis e reutilizáveis. O processo de síntese de unidades de controlo é também abordado, com uma atenção especial para as técnicas mais apropriadas para as FPGAs. Finalmente, para suportar a reconfiguração dinâmica dos circuitos desenvolvidos, foi construída uma biblioteca de classes em C++ e um controlador (*device driver*) para a placa de desenvolvimento utilizada neste trabalho.

Abstract

Control units are one of the most important types of digital circuits. They are used in a great variety of applications, from simple traffic light controllers to complex data processing systems. Their function is to establish the sequence of operations accomplished by the system they belong to. Depending on the application, they can be used separately or together with other components, such as execution units, sensors or actuators. Because control units are a particular kind of sequential circuits, they are usually described by state-oriented models, the finite state machine being the best known example. However, with the increasing complexity of the systems and consequently of its control units, this model is becoming less adapted to perform its specification, because it does not support the explicit description of hierarchy and concurrency. This dissertation presents some of the models and languages better suited to this purpose, in particular the hierarchical and/or parallel finite state machine, the hierarchical and/or parallel graph schemes and the Statecharts formalisms. The appearance in the last decade of high capacity user programmable logic devices was responsible for important modifications in the way digital systems are designed, mainly in terms of time, cost and design flexibility. Besides, the availability of dynamically reconfigurable devices made possible the emergence of a new class of circuits: the reconfigurable systems. They can be used to build modifiable computational systems combining the advantages of a programmable solution with the high performance of a hardware implementation. The reconfigurable systems can also be used in applications where the amount of required hardware resources for an integral implementation is too high or at least bigger than desirable and where all the subsystems are not necessary simultaneously, making possible a partial implementation in conjunction with its dynamic reconfiguration. Due to the analogy with virtual memory systems, these systems can also be called virtual

hardware systems and their respective control circuits virtual control units. Because the control units are specific for each project and usually very irregular, it is necessary to impose some constraints in order to simplify their design and reconfiguration. In this dissertation an architecture for virtual control units based on a predefined, parameterizable and optimized template for a particular target device, the FPGA XC6200 from Xilinx, is proposed. However, the main ideas of this architecture can also be easily applied to other FPGA families. The use of this architecture together with the hierarchical and/or parallel models and the dynamically reconfigurable logic devices allows building complex, flexible, extensible and reusable control units. This contributes to a decrease in development time and permits system updates after the completion of the design and manufacturing cycles. The control unit synthesis process is also presented with special emphasis on the techniques better suited to FPGA implementation. Finally, to support the dynamic reconfiguration of the developed circuits, a C++ class library and a device driver for the development system used in this work were built.

Título: Especificação, Optimização e Teste de Algoritmos de Controlo Hierárquicos.
Title: *Specification, Optimization and Testing of Hierarchical Control Algorithms.*
Autor/*Author*: Andreia Barbosa de Melo
Orientador/*Advisor*: Valery Sklyarov
Data Apresentação/*Acceptance Date*: 06/01/00
Palavras Chave: Técnicas de Especificação e Optimização, Algoritmos de Controlo, Implementação e Teste em FPGAs.
Key Words: *Specification and Optimization Techniques, Control Algorithms, Implementation and Testing in FPGAs.*
Mestrado/*M.S.*

Resumo

Este trabalho incide sobre a especificação de algoritmos de controlo, a sua verificação, optimização, conversão em código VHDL sintetisável (*VHSIC Hardware Description Language*) e depuração.

O comportamento dos algoritmos de controlo é especificado utilizando Esquemas de Grafos Hierárquicos (*HGSs – Hierarchical Graph Schemes*). Esta é uma linguagem gráfica formal de especificação que se baseia na utilização de nodos ligados entre si de forma semelhante a um fluxograma. Este método de especificação suporta hierarquia e permite construir algoritmos utilizando uma notação que é independente do modelo de implementação do circuito.

No âmbito deste trabalho foi construída uma ferramenta de software, chamada *GraphBuilder*, que consiste num ambiente integrado para especificação, optimização e teste de algoritmos de controlo hierárquicos. A construção de hierarquias de esquemas de grafos é suportada pelo *GraphBuilder* cujas principais funções são a construção/edição de forma gráfica e a gestão das hierarquias. Nesta aplicação é possível verificar a especificação do algoritmo, assim como gerar algoritmos aleatoriamente, sob determinadas restrições, para efeitos de teste. O *GraphBuilder* integra a aplicação *Optimaker*, responsável pela optimização e a aplicação *SynGraph* que converte os esquemas de grafos em código VHDL sintetizável. A partir do momento em que possuímos uma descrição do circuito em VHDL delegamos os passos posteriores do projecto às ferramentas de software já existentes. Assim, todos os passos desde a síntese do código VHDL até à implementação do circuito em hardware serão assegurados por ferramentas externas que são normalmente dependentes da tecnologia usada para implementar o circuito.

Após a implementação do circuito em hardware torna-se necessário verificar se o seu funcionamento é o que realmente foi especificado e para tal, a aplicação *GraphBuilder* permite efectuar a depuração do algoritmo. Desta forma fecha-se o ciclo do projecto do circuito pois tornou-se possível efectuar a depuração utilizando o mesmo ambiente que foi usado na sua especificação. No caso de ser necessário efectuar modificações ao circuito, estas poderão ser feitas e testadas mais rapidamente pois do ponto de vista do utilizador está a utilizar-se apenas uma ferramenta de software no projecto do circuito porque o ambiente *GraphBuilder* encarrega-se de invocar todas as aplicações necessárias de forma transparente.

Finalmente, os algoritmos de controlo são implementados em circuitos reconfiguráveis. Neste trabalho utilizaram-se as FPGAs (*Field Programmable Gate Arrays*) das famílias XC6200 (reconfigurável dinâmica e parcialmente) e XC4000 (reconfigurável estaticamente) da Xilinx.

Abstract

This work is devoted to the specification of control algorithms, their verification, optimisation, conversion to VHDL (VHSIC Hardware Description Language) code and real-time debugging

The behaviour of control algorithms is described by HGSs (Hierarchical Graph Schemes), which is a formal graphical specification language similar to flowcharts. This method supports hierarchy and parallelism and allows building control algorithms using a notation that is independent of the circuit implementation model.

Within the scope of this work, a software tool, called *GraphBuilder*, was developed. It consists of an integrated environment for specification, experimentation and test of hierarchical specifications. These are supported by *GraphBuilder*, whose main functions are graphical construction, edition and hierarchy management. It also performs verification and generates hierarchical control algorithms randomly, with some constraints, for testing purposes.

GraphBuilder integrates the application *Optimaker*, responsible for optimisation, and *SynGraph*, which converts Graph Schemes to synthesisable VHDL code. After building the VHDL description, we delegate the following design steps to the commercially available software tools, which are dependent on the technology used to implement the circuit.

After the hardware implementation it is necessary to verify if the circuit behaves as specified. For that purpose *GraphBuilder* provides real-time debugging facilities. Consequently, this tool closes the design cycle since it allows for specifying and debugging the circuit using the same environment. This simplifies the task of modifying, improving and correcting the initial specification.

Finally, the control algorithms are implemented in reconfigurable circuits. In this work FPGAs (Field Programmable Gate Arrays) of the Xilinx XC6200 (partially dynamically reconfigurable) and XC4000 (statically reconfigurable) families were used.

ARTES GRÁFICAS
AG
Sérgio Cabaço